Scalable Evaluation of Rule-Based Recommender Systems: Algorithms and Benchmarks

Len Feremans and Bart Goethals

Department of Computer Science, University of Antwerp, Belgium
firstname.lastname@uantwerpen.be

Abstract. Recommender systems help users to identify the most relevant items from a huge collection of items. Rule-based recommenders offer efficient, interpretable, accurate, and trustworthy recommendations, addressing key challenges in recommender design. Using association rules having a single or multiple conditions, we build transparent white-box models, especially for long-tail items. Moreover, recent studies challenge the trade-off between interpretability and accuracy. However, aspects beyond accuracy and efficiency—such as popularity bias, coverage, diversity, and comprehensibility—have largely been overlooked in prior evaluations. Additionally, well-known higher-order rule-based recommender methods lack scalability. Finally, many methods have been proposed that vary in rule form, scoring measures, aggregation and inference strategies. We introduce RuleRec, a scalable toolkit offering six seminal rule-based recommenders. We extend Apriori, MSApriori and adaptive-support rule mining, thereby presenting novel algorithms based on the generalization of pairwise rule-mining using an inverted index. We find that the proposed algorithms are an order of magnitude more efficient. Finally, we empirically evaluate six rule-based recommender algorithms on six benchmark datasets, comparing their accuracy, efficiency, diversity, popularity bias, and comprehensibility. To our knowledge, this is the first work to provide an efficient open-source implementation and comparative evaluation over multiple rule-based recommenders.

1 Introduction

Recommender systems (RS) are important in the current age where end-users are overwhelmed with information from news websites, social media, e-commerce, music and video services and emerging applications such as health care. Challenges for RS are limited user preference information and implicit feedback, where there is a high amount of uncertainty attached to interpreting observed user behaviour [11]. Another challenge is the cold start scenario, where user-item interactions are unavailable because the user or item is new.

A perceived disadvantage of traditional rule-based models such as decision trees and association rules is their relatively low accuracy compared to state-of-the-art deep learning methods. In the seminal paper [22], Rudin argues that it is a myth that there is a trade-off between accuracy and interpretability. Goethals

et al. conclude that the trade-off exists but that for the majority of datasets the difference is rather small [9]. Ludewig et al. find that a simple rule-based method (SR) outperforms recent deep learning-based RS in 4 out of 8 sequential recommendation datasets [16]. More recent approaches leveraging attention mechanisms and large language models could achieve superior accuracy, but this comes at the cost of increased training time and reduced trustworthiness. Deep learning–based RS are only post hoc explainable, where post hoc explanations are uncertain and not always faithful to the black-box model [20].

Most authors who develop new RS methods focus on accuracy, but the increase in accuracy is often marginal, whereas computational resource requirements have increased substantially. More recently, different authors recognize RS quality requirements beyond accuracy, such as fairness, trustworthiness, and diversity. The trustworthiness of an RS is enhanced by: (i) providing explanations for recommendations; (ii) taking the causability of a decision into account; (iii) ensuring recommendations are robust against noise in the data and adversarial attacks; (iv) managing popularity bias; (v) increasing diversity; (vi) ensuring calibration and fair exposure; (vii) and comprehension of the model [2, 28, 25, 9].

In addition, in emerging and high-stakes domains, such as healthcare, education, or the legal domain, precise and interpretable recommendations are a requirement. Rule-based RS have several interesting properties, namely: (i) intrinsic interpretability; (ii) the ability to identify accurate patterns and rules locally; (iii) the capability to learn from both user interactions and content/context features; (iv) the potential for rule list optimization and ensembles to increase accuracy; (v) high efficiency; (vi) capture higher-order interactions; (vii) capture short-term and dynamic user preferences; (viii) and learn online directly from user feedback. In this paper, we compare different RS, and measure the diversity, popularity bias, and comprehensibility of rule-based RS, which is often not reported in original work. Finally, we tackle scalability issues that are common in traditional higher-order rule mining methods. We make the following contributions:

- We analyze the literature and categorize and compare a wide variety of rule-based recommendation algorithms proposed in the past decades.
- We introduce RuleRec, a new toolkit that contains efficient algorithms of a wide variety of rule-based recommenders. Technically, we extend Apriori-based rule mining algorithms to efficiently scale to large datasets.
- We perform an independent evaluation study using six publicly available benchmark datasets from different domains, including e-commerce, news, and music. We extend prior work by comparing precision, recall, coverage, popularity bias, diversity, comprehensibility, and runtime.

2 Background and definitions

First, we introduce some background terminology for recommender systems and rule mining.

2.1 Collaborative-filtering based recommender systems.

Recommender systems learn models from historical user interactions. Let U be the set of users and I the set of items. Each user (or session) $p \in U$ has a set of implicit interactions $p = \{s_1, s_2, \ldots, s_k\}$, where $s_i \in I$ denotes a positive action (e.g., a click). Interactions are typically logged online as tuples of user, item, and optionally time. All interactions can be represented as a sparse user–item matrix $R \in \mathbb{R}^{|U|\times|I|}$, since most users interact with only a few items in a large catalog. Depending on the dataset, interactions may represent product purchases, clicks on movies or songs, or news article views.

The goal of a recommender system is to suggest the most relevant items for each user. Content-based methods rely on user and item metadata (e.g., recommending other thrillers by John Grisham if the user liked one), while collaborative filtering leverages user–item interactions (e.g., users who liked "The Firm" also like X). In this work, we focus on collaborative filtering, although rule-mining methods can exploit metadata, interaction data, or both.

To compare approaches, we evaluate systems offline. A model $\mathcal{M}$ is trained on historical interactions and, during inference, produces top-n recommendations for each user. Users are split into training and test sets. For each test user $p \in U_{\text{test}}$, interactions are partitioned into observed p^{obs} and held-out p^{held} sets. The model generates a ranked list $\hat{p} = \hat{s}_1, \ldots, \hat{s}_n$ from p^{obs}, which is then compared with p^{held}. Evaluation uses recall@k, rank-aware measures such as NDCG@k, and secondary metrics addressing popularity bias and diversity.

2.2 Apriori

Agrawal et al. introduced Apriori, the first method for mining patterns and association rules in large databases [3]. Apriori uses breadth-first search to generate singletons, pairs, triples, etc., and discovers itemsets X that exceed a minimum support threshold θ, often set low (e.g., covering 1% of instances). Crucial to the efficiency of Apriori is the anti-monotonicity of support, i.e. for each set of items X the support, or the number of occurrences in a database, is smaller for any superset Z of X. Next, Apriori outputs association rules for each frequent itemset thereby partitioning the itemset into an antecedent and consequent. Given an itemset $X = \{A, B\}$, we generate a rule $A \rightarrow B$ where:

$$\mathit{conf}(A \rightarrow B) = P(B \mid A) = \frac{\mathit{support}(A \wedge B)}{\mathit{support}(A)} \text{ and } \mathit{support}(X) = \sum_{p \in U} 1_{EQ}(X \subseteq p).$$

Here, $1_{EQ}(\mathit{predicate})$ is an indicator function that returns 1 if the predicate is true, and 0 otherwise. Apriori also prunes rules with confidence below a minimum threshold, denoted as γ. In RS, the consequent typically is a single item. If the rule matches with the user history, the consequent of the rule is recommended to the user. We review RS that mine itemsets with different support thresholds [15], or mine the top-k confident rules directly [17, 14]. In Section 4, we discuss how to scale Apriori to large, sparse datasets.

3 Review of existing rule-based RS methods

In this section we review seminal rule-based recommender methods. Existing work mainly differs in the following aspects:

- Many RS restrict conditions to a single item, yielding "because you like X, we recommend Y" rules. Others RS adapt Apriori to enumerate higher-order conditions, yielding "because you like X_1 and X_2 and we recommend Y".
- Items are ranked using scores such as confidence, cosine similarity, or adjusted confidence. Rules may be chosen by their maximum score or by aggregating similarities across rules with the same consequent.
- Recommendation (or reasoning) strategies vary: some incorporate sequential patterns or history filtering, crucial in domains where time matters (e.g., news or music).
- Mining approaches differ, with various constraints and pruning strategies to limit the combinatorial search space.

An overview of the seminal rule-based recommender systems compared in this work is shown in Table 1. An example is shown in Fig. 1.

3.1 Recommendations based on pairwise association rules

Ludewig et al. propose Simple Association Rules (AR), which is based on pairwise association rules [16]. Let U denote the set of all users and p contains a time-ordered list of items $p = (s_1, \ldots, s_k)$. We compute the confidence for each item pair s_i, s_j using

$$conf(s_i \rightarrow s_j) = \frac{support(s_i \wedge s_j)}{support_{LEN}(s_i)} \quad \text{where}$$

	Inception	*Titanic*	*The Matrix*	*Avatar*
User 1		✓	✓	
User 2	✓		✓	✓
User 3		✓		✓
User 4	✓	✓	✓	
User 5	✓	✓	?	?

Fig. 1. Example of the user-item interaction matrix. For **User 5** different rule-based recommender systems yield the following Top-1 recommendation:
a) Pairwise rules: We find *Inception* $\rightarrow$ *The Matrix* ($conf = \frac{2}{2}$) and *Titanic* $\rightarrow$ *Avatar* ($conf = \frac{1}{3}$). So recommend *The Matrix*.
b) Higher-order rules: We find $\{$*Inception*, *Titanic*$\}$ $\rightarrow$ *The Matrix* ($conf = \frac{1}{1}$), so recommend *The Matrix*.
c) Similarity-based: Both *Inception* and *Titanic* have higher cosine similarity with *The Matrix* ($cos = \frac{2}{3} + \frac{2}{3}$) than *Avatar* ($cos = \frac{1}{2} + \frac{1}{\sqrt{3}\sqrt{2}}$), so recommend *The Matrix*.

$$support_{LEN}(s_i) = \sum_{p \in U} 1_{EQ}(s_i \in p) \cdot (|p| - 1).$$

Our definition of AR is equivalent to that of Ludewig et al. under the assumption that every item s_i occurs once in each session. The denominator is different from the traditional confidence definition which Ludewig motivates as a normalisation against the number of rules. However, we argue this formulation mainly imposes a severe penalty for longer user sessions. For top-n recommendations we compute:

$$score_{AR}(p, s_i) = conf(s_k \rightarrow s_i)$$

That is, we compute recommendations based on s_k, the last clicked item in the observed user session. We store the top-n rules with the highest confidence for each antecedent. Since we generate rules with rare items in the antecedent, this method yields high recall and coverage; however, low confidence and support values may reduce precision.

3.2 Recommendations based on pairwise sequential rules

The second pairwise method uses sequential rules (SR) and was proposed by Kamehkhosh et al. [12] and found to be competitive with deep learning methods in [16]. We consider the probability of a transition from an item A to B where A occurs before B. In contrast, Markov Chains model the transition from items occurring consecutively. The authors propose a weighting for sequential rule confidence, where the weight decays based on the number of items separating the interaction between A and B in a session. We compute the confidence for each sequential pair s_i, s_j using:

$$conf(s_i \rightarrow s_j) = \frac{support_{DUR}(s_i, s_j)}{support(s_i)} \quad \text{where}$$

$$support_{DUR}(s_i, s_j) = \sum_{p \in S} \frac{1_{EQ}(\langle s_i, s_j \rangle \prec p)}{duration(s_i, s_j)}$$

We use the notation $\langle s_i, s_j \rangle \prec p$ to denote that item s_i occurs before s_j in session p. The duration of two items s_i and s_j is defined by the number of items between s_i and s_j + 1. Intuitively, if s_i is always just before s_j, the denominator will be 1. During inference, they rank top-n recommendations for items based on the pairwise confidence with the last interacted item.

3.3 Item-based collaborative filtering

Deshpande et al. introduced item-based collaborative filtering (IBCF) that makes predictions based on the k items with the highest similarity, defined using cosine similarity [5]. The cosine similarity between two item vectors $\mathbf{x}$ and $\mathbf{y}$ is given by the following formula:

$$cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

where x_i and y_i are the components of the vectors $\mathbf{x}$ and $\mathbf{y}$, and n is the number of dimensions (in this case, users who have rated both items). Assuming binary implicit feedback, we rewrite the previous equation as [4]:

$$cos(s_i, s_j) = \frac{support(s_i \wedge s_j)}{\sqrt{support(s_i)} \cdot \sqrt{support(s_j)}} = conf(s_i \rightarrow s_j)^{\frac{1}{2}} \cdot conf(s_j \rightarrow s_i)^{\frac{1}{2}}. \tag{1}$$

To generate the top-n recommendations for a given user based on IBCF, we compute:

$$score_{IBCF}(p, s_j) = \sum_{s_i \in p} cos\_top\_k(s_i, s_j) \text{ where}$$

$$cos\_top\_k(s_i, s_j) = \begin{cases} cos(s_i, s_j) & \text{if } cos(s_i, s_j) \text{ ranked in top-}k \text{ for } s_i \\ 0 & \text{otherwise} \end{cases}.$$

The parameter k is introduced to increase efficiency, since we only need to store $k \cdot |I|$ rules. If k is sufficiently large (default set to 100), an increase in k will typically not change the order of the top-n items recommendations assuming $n \ll k$. We note that Deshpande also studied using confidence, scaling the support of s_j, and normalization based on the length of each session. IBCF is closely related to AR, because predictions are based on the association between item pairs. A key difference with AR is that for computing the score we aggregate the similarities for each consequent, and typically use the full user history.

3.4 Higher-order association rules using multiple support thresholds and window-based recommendations by Mobasher

Mobasher et al. leverages Apriori to discover higher-order itemsets that are frequent [18]. A potential advantage of higher-order association mining is the discovery of highly confident rules that have larger antecedents. A problem with Apriori, is that if we set the minimal support threshold too low, we generate too many patterns. On the other hand, if we set the threshold too high, we miss "nuggets", i.e., rare but strong rules. Hence, Mobasher adopts MSApriori thereby using multiple minimum support levels [15]. A pattern X is frequent subject to multiple support levels if:

$$support(X) \geq \min_{i \in X}(MIS(i)) \qquad \text{where}$$

$$MIS(i) = \begin{cases} support(i) \cdot \beta & \text{if } support(i) \geq \theta \\ \theta & \text{otherwise} \end{cases}. \tag{2}$$

Here θ is the global minimal support threshold and β controls the relative support. The intuition is that we want to limit ourselves to enumerating many higher-order patterns consisting of frequent (or popular) items. Additionally, we constrain the maximal size d of each itemset, usually set to 3 or 4. After mining frequent itemsets X, we generate all rules $X \setminus \{y\} \rightarrow \{y\}$ and discard rules below the

minimal confidence threshold γ. For generating recommendations, the authors propose an *all-kth-order* approach inspired by Markov chain models and makes prediction by matching rules of size $d, d-1, \ldots, 2$ until a recommendation is generated. Additionally, they only consider the last $d-1$ items in the user's history. The rationale behind this process is that higher-order rules, by taking a larger fraction of the (most recent) user's history into account, increase precision. Finally, the authors propose to index rules using a Frequent Itemset Graph before generating recommendations [18]. However, we find this does not scale to larger datasets with millions of users and items, and present a more efficient solution in Section 4.

3.5 Adaptive-support higher-order association rules by Lin

A second method that mines higher-order rules for recommendations was proposed by Lin [17, 14]. Again, the motivation is to discover "nuggets", i.e. rare but strong rules. A crucial difference with the previous method, is that we do not define a minimal support threshold, do not filter the user history, and do not prioritize longer rules. Instead, Lin proposes to discover between $[k_{min}, k_{max}]$ confident rules local to each possible consequent item. The algorithm directly finds confident rules for each consequent item instead of the common two-phase approach where we first mine frequent itemsets and then extract confident rules [17, 15]. The authors report that mining more than k_{max} (e.g. 100) rules for each item does not improve recommendation accuracy and avoids an unnecessarily large runtime. For making rule-based recommendations, we compute the top-n eligible rules with the highest confidence.

3.6 Higher-order rule-based recommendations using adjusted confidence and default rules by Rudin

More recently, Rudin proposed a rule-based recommender thereby combining ideas from statistical learning with association rule mining. The main novelty is that items are ranked on the adjusted confidence measure, which provides an improved estimate of the true conditional probability [23]. For mining association rules they propose to use an existing algorithm, such as Apriori or FP-Growth. For recommendations, all rules are ranked on adjusted confidence for each user, and as such, there is no threshold on the minimal confidence. Additionally, if fewer than n matching rules are found we fall back on rules with empty antecedents, effectively providing popularity-based recommendations. The adjusted confidence is defined as:

$$conf_{ADJ}(X \rightarrow s_j) = \begin{cases} \frac{support(X \wedge s_j)}{support(X)+K} & \text{if } X \neq \emptyset \\ \frac{support(s_j)}{|S|+K} & \text{otherwise} \end{cases}. \tag{3}$$

where K is a hyper-parameter. The intuition is that K bounds the support of both the left and right-hand side of the rule. They find that, if the confidence of two rules is similar, support should be used to break ties, which is what adjusted

Method	Rule form	Scoring	Mining strategy	Reasoning step
Simple association rules [16]	$x \rightarrow y$	Confidence weighted by session length	Enumerate all item pairs	History filtering and use rule with maximal confidence
Sequential rules [12]	$x \rightarrow y$	Confidence weighted by duration	Enumerate all sequential pairs	History filtering and use rule with maximal confidence
Item-based CF [5]	$x \rightarrow y$	Cosine similarity	Enumerate all similar pairs	Aggregate the similarity using all items in the history
Association rules with multiple support thresholds [18]	$x_1, \ldots x_d \rightarrow y$	Confidence prioritizing longer rules	Apriori with multiple minimum support levels	Window-based history filtering and use rule with maximal confidence
Adaptive-support association rules [14]	$x_1, \ldots x_d \rightarrow y$	Confidence	Apriori local to each consequent with adaptive support levels	Full history and use rule with maximal confidence
Sequential Event Prediction [23]	$x_1, \ldots x_d \rightarrow y$ $\{\} \rightarrow y$	Adjusted confidence	Apriori	Full history and use rule with maximal confidence or default rule

Table 1. An overview of seminal rule-based recommender systems in RuleRec.

confidence does. Experimentally, they report that accuracy is maximized for small values of K and the effect is higher than tuning the minimal support and confidence thresholds.

4 RuleRec: An efficient rule-based recommender library

We propose RuleRec, a lightweight library for pre-processing, rule mining, rule-based recommendation, and evaluation. A disadvantage of the original methods for higher-order rule mining is that they do not scale to sparse datasets with millions of users, items and interactions. We want to ensure higher-order mining is not dramatically less efficient than pairwise rule mining, when generating rules that are more precise by having multiple conditions in the antecedent. Hence, we developed scalable algorithms for rule mining in large, sparse datasets, leveraging techniques such as projection-based support counting, and reverse indexing. Additionally, we separate rule mining from inference, allowing for configurable strategies for selecting, ranking, and explaining recommendations.

4.1 Pairwise methods

To compute the similarity-matrix, or the confidence between all item pairs, the naïve algorithm has a complexity of $O(|I|^2)$. We propose a generic method for the implementation of AR, SR and IBCF. Where we compute and rank pairwise rules on confidence or similarity scores, using Eq. 1 for the latter [4]. Crucial for efficiency is that we depend on the number of entries in the user-item interaction matrix $R \in \mathbb{R}^{|U| \times |I|}$, typically having less than 1% non-zero entries. We leverage an inverted index, where items and users are indexed. By iterating

over the inverted indices, we compute only nonzero co-occurrence counts. This approach is embarrassingly parallel and highly effective for sparse datasets. This process is illustrated in the first three steps of Fig. 2. Finally, for generating recommendations and explanations, we propose a reusable module, RFER, which is discussed later.

4.2 Efficiently mining higher-order association rules in sparse datasets

In this section, we propose new algorithms to improve the efficiency of Apriori [3], MSApriori [15] and adaptive support mining [17, 14]. The key idea of our method is to compute all co-occurrences using an inverted index similar to pairwise recommenders, but generalized for higher-order itemsets. That is, we consider each frequent itemset of size k as a new virtual item, and maintain a list of all users who interacted with the itemset. The use of *projection*-based indexing has two advantages: candidate-free generation and fast support computation for higher-order itemsets. We remark that candidate-free enumeration is also used in FP-Growth [10, 26].

Extending Apriori. Similar to Apriori, we search for frequent itemsets using breadth-first search. That is, at each level we generate patterns of size k, and prune patterns that are not frequent. The proposed algorithm, Apriori++, is illustrated in Fig. 2. First, we compute an inverted index for pairs, and compute co-occurrences. For higher-order itemsets we generalize the process. Formally, given a pattern $X_k = \{s_i, \ldots, s_j\}$, we define a projection $B(X_k)$, as the set of users that

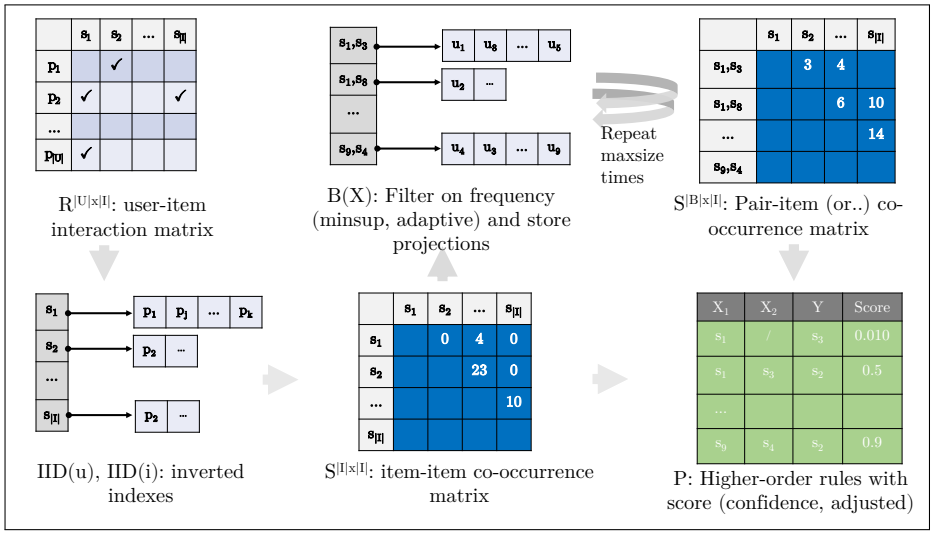

Fig. 2. Proposed algorithm for efficient pairwise and Apriori-based rule mining and recommendations.

interacted with all items in X_k. For any candidate superset $X_{k+1} = X_k \cup \{s_k\}$, we need to locate all users interacting with X_k and s_k. Instead of computing the support as

$$support(X_{k+1}) = | \cap_{s_i \in X_{k+1}} IID(s_i) |.$$

We compute:

$$support(X_{k+1}) = |B(X_k) \cap IID(s_i)|.$$

Hence, amortized, only a single intersection is necessary. Moreover, the size of the projection significantly decreases as pattern length increases. In addition, we probe the projection and the inverted index. That is, for any candidate superset $X_{k+1} = X_k \cup \{s_k\}$, it holds that:

$$s_k \in \bigcup_{u_i \in B(X_k)} IID(u_i).$$

The consequence is that any candidate superset X_{k+1} occurs at least once in a user session. We repeat the creation of projections for pattern of size k, computing co-occurrences for patterns of size $k+1$, and prune patterns of size $k+1$ on frequency. Finally, we collect all varying-length frequent itemsets and compute rules and their confidence, which is a trivial step. The worst-case complexity is similar to Apriori, but the use of projections significantly reduces the practical runtime. We make a list of all frequent patterns of size $1, 2, \ldots d$ level-wise. Assuming $d = 4$, this results in at most $|I|^4$ candidates. However, if we prune infrequent patterns at each level k, using the threshold θ, the number of candidates is much lower. In Section 5, we compare the effectiveness of the proposed method against prior work [7].

Extending MSApriori. The rationale behind MSApriori, is we discover "nuggets", i.e. infrequent, but strong association rules. Extending MSApriori is trivial, since crucial steps such as computing projections, and the co-occurrence matrix level-wise are independent of the differences in any level-wise pruning strategy. Hence, we adopt the aforementioned algorithm, but prune itemsets of size k using Equation 2 instead of using a single threshold θ.

Extending Adaptive mining. The miner proposed by Lin is different in design, since the goal is to find the top-k rules with the highest confidence for each consequent item. The first key difference, is that the threshold on minimal support is adaptive, and we run Apriori with varying minimal support threshold, until the number of rules of the form $X \rightarrow s_t$ is in the configurable range $[k_{min}, k_{max}]$ [14]. However, in each iteration, we have to enumerate patterns and rules local to each item $s_t \in I$. Hence, the described optimisations are applicable, such as the use of an inverted index, co-occurrence matrices, projections and generalization to higher-order itemsets. For brevity, we discuss a novel algorithm for adaptive-support mining in the Supplementary material, where we adapt co-occurrence and projection-based computation from Apriori++ but constrained to itemsets of the form $X \cup \{s_t\}$.

4.3 RFER: An efficient and general approach for generating recommendations

Several authors do not discuss any strategies to efficiently calculate rule-based recommendations [5, 14] or propose complex and inefficient solutions [18]. We propose a general component to Retrieve, Filter, Explain and Rank rules (RFER). RFER assumes a large catalog of rules as input, and the history of selected users. We perform the following steps:

- First, the user history is preprocessed by selecting the full history or the last item(s)
- Next, eligible rules are retrieved from a potentially large rule catalog using a two-step process.
- Next, we create recommendations by ranking rules with the highest confidence measure, size, or aggregating rules have the same consequent.
- Finally, we link each recommendation, with the rule(s) causing the recommendation to enable explanations and analysis of the comprehensibility.

By logically separating rule mining and rule-based reasoning, we enable more variation than originally proposed. That is, history filtering, default rules, and various ranking and aggregation strategy for inference in rule-based recommendations, can be combined independently from any rule mining method. We observe that even if the number of rules is large, the number of eligible rules for a single user u is likely very small. Hence, we construct a simple rule index by mapping antecedents to rules using a hash map. For retrieval, we simply probe the hash map and filter eligible rules in a second phase. Previous work, suggested using a Frequent Itemset Graph [18]. However, storing and traversing the complete graph, i.e. keeping all pairwise and higher-order rules and relations between sub- and supersets is memory-intensive and computationally inefficient.

4.4 Higher-order methods

We now discuss the implementation of the three higher-order rule-based recommender from Table 1. For MOB, we use MSApriori++ and RFER for making recommendations. During inference, we perform history filtering using a sliding window, and prioritize longer rule, by ranking on size, and only secondary on confidence. For LIN, we use the novel, efficient algorithm of Adaptive mining for mining rules and RFER for inference, where we rank rules on confidence and use rule support to break ties. Finally, for RUDIN we use Apriori++ to mine rules. After mining, we compute the adjusted confidence for each rule using Equation 3. We generate recommendations using RFER, prioritize rules on adjusted confidence, and use default, popularity-based recommendations if fewer than top-n recommendations are generated.

5 Experiments

In this section, we aim to answer the following research questions: **RQ1**: How do the rule-based RS compare on accuracy? **RQ2**: How do the rule-based RS

compare on runtime? What is the runtime of Apriori++ compared to Apriori? **RQ3**: How do the rule-based RS compare on diversity and popularity bias? **RQ4**: How do the rule-based RS compare on comprehensibility?

5.1 Experimental setup

In our experiments, we evaluated rule-based RS on six real-world datasets from the e-commerce, news, and music domains. All datasets and the RuleRec library are publicly available[1].

Datasets. We evaluated six implicit-feedback datasets from music (30Music, AOTM), e-commerce (Retailrocket, RCS15), and news (CLEF, EBNeRD). Metadata was not used. We applied standard preprocessing: removing re-consumed items and filtering users/items with fewer than five interactions. On average, the datasets contain 610k users, 47k items, and 4.8M interactions. The news datasets have smaller item catalogs upto 5k, while the largest dataset, RCS15, includes 1.2M users, 34k items, and 10M interactions.

Evaluation protocol. We created a random split of disjoint users using 80% for training and 20% for evaluation following the strong generalization principle. For evaluation, we hide the last 20% of interactions of test users, and generate the top-20 recommendations.

Evaluation metrics. We report NDCG, recall, F1, and precision. We also report user coverage, i.e. the percentage of users with at least one recommendation. We measure the following secondary measures: intra-list list diversity [28], item coverage [2], and the average percentage of long-tail item recommendations (APLT) [1]. For comprehensibility, we report the number of rules. Additionally, we propose to report crucial statistics related to rule form and popularity bias, including the percentage of rules with 2 conditions, the percentage of rules with a popular item as consequent, and the percentage of rules between long-tail items. Here, popular items are defined as those items cumulatively covering 80% of all user-item interactions. A summary of well-known evaluation measures is presented in Table 3.

Parameter selection. We selected parameters after exploring the effect using grid search. Similar to [18, 14], we also report a precision-recall trade-off, having maximal recall with lower confidence and support thresholds, and maximal precision (and F1) using a higher confidence thresholds. In Table 2 we present the parameter setting used in our experiments, tuned for recall (and diversity), and for prediction (and F1). We set *normalize* to True for IBCF, and the adjusted confidence parameter K to 0.001 for RUDIN resulting in optimal recall for both methods. For higher-order methods we set d to 3, resulting in rules consisting of 1 or 2 conditions.

[1] RuleRec is implemented in Java and Python, open-source and available at `https://bitbucket.org/len_feremans/rule-based-recommenders/`

5.2 RQ1: How do the rule-based RS compare on accuracy?

First, we compare all methods on NDCG and recall. For brevity, we report average results in Table 4. Detailed results are available in the supplementary material. Our first observation is that SR, on average, is the most accurate method, performing best on 3 out of 6 datasets for recall and 4 out of 6 for NDCG. This aligns with its known strong performance in sequential recommendation [16]. Second, since AR and SR base predictions on only the last interaction, this confirms that history filtering is a simple yet effective strategy for improving ranking. Third, there is no universal best method: IBCF (cosine similarity) performs best on AOTM, while LIN achieves the highest recall on two datasets. Among higher-order methods, LIN outperforms RUDIN and especially MOB, which has the lowest recall. We attribute this to LIN's pruning strategy, which ensures at least k_{min} rules per item, yielding more pairwise rules. Finally, in terms of precision and F1, AR is the clear winner, while RUDIN performs worst. MOB achieves a relatively high F1 score despite its lower recall.

5.3 RQ2: How do the rule-based RS compare on runtime?

In Table 4, we also report the average runtime of each method. Our implementation of pairwise methods, as well as higher-order methods MOB and LIN,

	AR	**SR**	**IBCF**	**MOB**	**LIN**	**RUDIN**
Optimal recall	$k = 20$	$k = 20$	$k = 200$	$\theta = 2$, $\beta = 0.1$	$k_{min} = 20$, $k_{max} = 200$	$\theta = 10$
	$\gamma = 0.0$	$\gamma = 0.0$	$\gamma = 0.0$	$\gamma = 0.001$	$\gamma = 0.001$	$\gamma = 0.001$
Optimal F1	$k = 20$	$k = 20$	$k = 50$	$\theta = 5$, $\beta = 0.1$	$k_{min} = 5$, $k_{max} = 50$	$\theta = 10$
	$\gamma = 0.25$	$\gamma = 0.25$	$\gamma = 0.25$	$\gamma = 0.25$	$\gamma = 0.25$	$\gamma = 0.25$

Table 2. Parameter settings for optimal recall (and NDCG, diversity and coverage) and F1 (and precision and comprehensibility) after grid search.

Metric	**Description**
Hitrate, NDCG, MRR [21]	Measure the relevance, rank-aware or otherwise, of top-n recommendations.
Precision, recall, F1, AUROC [21]	Measure the precision, recall, and the precision-recall trade-off when generating recommendations while varying the confidence threshold.
Item coverage [2]	Number of distinct items appearing in any top-k recommendation, also referred to as aggregate diversity.
Popularity bias [1]	Average popularity of recommended items and percentage of long-tail item recommendations.
Diversity [28]	Average dissimilarity of all pairs of items in recommendation lists.
Calibration, fairness [25]	Difference in distribution of an attribute between training and recommended items. More recently studied as a fairness constraint.
Model comprehension [9]	Number of rules, length of antecedent, rule structure etc. important towards comprehension.

Table 3. Overview of evaluation metrics

completes rule mining and recommendation within 10 minutes. Given that the largest dataset, RCS15, contains over 1 million users and 10 million interactions, and that we generate recommendations for 250k users, training and inference require less than 2 ms per user. For RUDIN, which uses Apriori, we set θ to 10 to reduce runtime, yet it still exceeded one hour on three datasets. In contrast, MOB and LIN gain efficiency from multiple or adaptive support thresholds.

We also compared Apriori++ with Apriori from the SPMF Library [7]. With a high support threshold $\theta = 93$ and $d = 3$, SPMF took over 300 seconds to find 867 itemsets on the 30Music dataset. At $\theta = 5$, it ran out of memory. Apriori++, in contrast, completed in under 1 second for $\theta = 93$ (300× faster), and within 12 seconds for $\theta = 5$, discovering 1.9 million itemsets. Since a large number of rules is often required for high recall and coverage, we conclude that classic implementations of Apriori (and related methods like MSApriori and FPGrowth) do not scale well on sparse recommender datasets with millions of users and items.

5.4 RQ3: How do the rule-based RS compare on diversity and popularity bias?

In Table 5 we report the results for diversity, coverage and popularity bias. Concerning diversity, IBCF and RUDIN perform best both scoring 0.88 on average, and MOB performs poorly, which is possibly caused by the window-based inference strategy. Concerning item coverage and average percentage of long-tail recommendation, we find that LIN and SR outperform other methods, where IBCF, MOB, and RUDIN exhibit greater popularity bias, as they recommend more popular items. For user coverage, most methods, generate recommendations for all users on most datasets, except MOB. Since MOB decreases diversity, increases popularity bias, and has low item and user coverage, this method performs poorly. We conclude that, only AR, SR and LIN perform well on all secondary evaluation measures.

		AR	**SR**	**IBCF**	**MOB**	**LIN**	**RUDIN**
NDCG@20	*Average*	0.220	**0.246**	0.167	0.146	0.189	0.157
	Wins	1	**4**	1	0	0	0
Recall@20	*Average*	0.408	**0.427**	0.366	0.266	0.391	0.328
	Wins	0	**3**	1	0	2	0
Precision@20	*Average*	**0.059**	0.044	0.029	0.040	0.041	0.018
	Wins	**4**	1	1	0	0	0
F1@20	*Average*	**0.188**	0.088	0.109	0.163	0.108	0.067
	Wins	**4**	1	0	0	1	0
Runtime	*Average*	20.1s	**19.2**s	55.5s	35.7s	183.0s	3151.7s
	Wins	1	**3**	0	0	0	2

Table 4. Average results of rule-based RS on NDCG, recall, precision, F1 and runtime on six datasets.

5.5 RQ4: How do the rule-based RS compare on comprehensibility?

In Table 6, we report statistics on model comprehension and popularity bias. Pairwise methods AR and SR yield the fewest rules, typically $k \cdot |I|$. For IBCF, k is fixed at 200, producing a much larger rule set, reduced comprehensibility, and more complex explanations. Higher-order methods generate fewer rules than IBCF. MOB and RUDIN vary greatly, as their rule counts are highly sensitive to pruning thresholds; for instance, RUDIN has the largest rule set on three datasets and the smallest on three others. For news datasets with smaller $|I|$, all methods yield fewer rules—except RUDIN. From Table 6, we also observe that LIN discovers more pairwise rules than MOB and RUDIN. RUDIN mainly produces rules with popular consequents, whereas MOB and LIN generate more with long-tail antecedents and consequents. These findings suggest that postprocessing and rule-set optimization may be promising directions for future work. Although the total number of rules is high, domain experts can still filter and review those relevant to specific items. To better capture model complexity, we recommend Sankey plots [24]. These visualize the flow between user recommendations, eligible rules, and relevant items, offering insight into model behavior—something which is not possible with black-box models. Further examples are provided in the supplementary material.

6 Related work

Wu et al. developed a framework based on FP-Growth and distributed processing for scalable rule-based recommendations [26]. Their work complements ours but

		AR	**SR**	**IBCF**	**MOB**	**LIN**	**RUDIN**
Diversity	*Average*	0.845	0.862	0.881	0.579	0.869	**0.883**
	Wins	0	0	4	0	0	2
Item coverage	*Average*	69.2%	77.9%	52.5%	47.8%	**81.1%**	33.1%
	Wins	0	1	1	0	4	0
APLT	*Average*	14.4%	**18.6%**	7.9%	13.9%	17.2%	8.1%
	Wins	0	4	1	1	0	0
User coverage	*Average*	99.0%	98.6%	99.9%	80.6%	99.9%	**100%**

Table 5. Average results of rule-based RS on diversity, item coverage, average percentage long-tail item recommendation, and user coverage on six datasets.

		AR	**SR**	**IBCF**	**MOB**	**LIN**	**RUDIN**
Number of rules	*Average*	843.4k	**757.6**k	3067.9k	1990.4k	2841.7k	1920.3k
Rules of size 3	*Average*	0%	0%	0%	49%	8%	**67%**
Rules pop. cons.	*Average*	64%	61%	68%	53%	37%	**89%**
Long-tail rules	*Average*	31%	32%	27%	**39%**	39%	7%

Table 6. Average results of rule-based RS on comprehensibility, rule size, and rule popularity bias on six datasets.

differs from our work in two key aspects: they focus on distributed processing with load-balancing and partitioning for parallel speed-up, and they propose a single rule-mining algorithm with alternative confidence measures. Several authors have empirically studied higher-order rule-based recommenders. Deshpande et al. examined similarity measures, normalization strategies, and parameter effects such as k [5]. They report IBCF often outperforms Apriori; however, we find Lin's adaptive-support method surpasses IBCF. More recent work extends the above methods: Osadchiy et al. [19] use pairwise association rules with aggregated confidence and support; Forsati et al. [6] mine higher-order rules ranked by frequency and dwell time; Yap and Gedikli personalize rule selection based on similar sessions [27, 8]; and Rudin [23] learns weights for all pairwise associations to optimize a target objective. Finally, SR has been extended with higher-order sequential rules weighted by duration for news recommendation [13]. Most studies still evaluate mainly accuracy on datasets from a single domain.

7 Conclusion

We analyzed the literature to categorize and compare a broad range of rule-based recommendation algorithms. Our study revealed substantial differences in rule structure, confidence measures, pruning strategies, mining constraints, and inference. Notably, existing higher-order rule mining methods often fail to scale on large, sparse datasets. Moreover, prior research has focused primarily on accuracy, neglecting critical factors like efficiency, diversity, popularity bias, and comprehension. To address these gaps, we implemented and evaluated a diverse set of representative algorithms in RuleRec—an open-source, lightweight, and scalable library featuring modules for preprocessing, evaluation, visualization, and various rule-based recommenders. We evaluated all methods on six benchmark datasets, yielding insights into trade-offs in accuracy, comprehensibility, diversity, and popularity bias. Pairwise and sequential association rule methods with history filtering show strong NDCG performance. Higher-order approaches, such as Lin's adaptive-support technique, provide clear benefits in recall and item coverage. Our proposed Apriori-based rule mining algorithms achieve up to 300× speed-up over a well-known baseline. All recommender systems complete within minutes on large, sparse datasets. We extend rule mining through level-wise generation, adaptive or multiple threshold pruning, and projection-based data structures. Our modular design further decouples rule mining from inference, offering greater flexibility and efficiency than prior systems. In domains demanding transparency, fairness, and efficiency — such as healthcare, education, or public services — rule-based recommenders provide a compelling alternative. A limitation of our study is the lack of detailed analysis on how hyper-parameters and dataset characteristics (e.g., sparsity and popularity bias) affect effectiveness. For future work, we recommend advancing rule mining, exploring multi-objective optimization, and conducting independent evaluation studies.

Acknowledgements. L.F. is funded on Research Fund Flanders grant 12B0V24N.

Bibliography

[1] Abdollahpouri, H., Burke, R., Mobasher, B.: Managing popularity bias in recommender systems with personalized re-ranking. In: The Thirty-Second International FLAIRS Conference (2019)

[2] Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Transactions on Knowledge and Data Engineering **24**(5), 896–911 (2011)

[3] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB). vol. 1215, pp. 487–499. Santiago, Chile (1994)

[4] Aiolli, F.: Efficient top-n recommendation for very large scale binary rated datasets. In: Proceedings of the 7th ACM Conference on Recommender Systems. pp. 273–280 (2013)

[5] Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems **22**(1), 143–177 (2004)

[6] Forsati, R., Meybodi, M.R., Neiat, A.G.: Web page personalization based on weighted association rules. In: 2009 International Conference on Electronic Computer Technology. pp. 130–135 (2009)

[7] Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 36–40 (2016)

[8] Gedikli, F., Jannach, D.: Neighborhood-restricted mining and weighted application of association rules for recommenders. In: International Conference on Web Information Systems Engineering. pp. 157–165 (2010)

[9] Goethals, S., Martens, D., Evgeniou, T.: The non-linear nature of the cost of comprehensibility. Journal of Big Data **9**(1), 30 (2022)

[10] Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Mining and Knowledge Discovery **8**(1), 53–87 (2004)

[11] Jannach, D., Quadrana, M., Cremonesi, P.: Session-based recommender systems. In: Recommender Systems Handbook, pp. 301–334. Springer (2022)

[12] Kamehkhosh, I., Jannach, D., Ludewig, M.: A comparison of frequent pattern techniques and a deep learning method for session-based recommendation. In: RecTemp@RecSys. pp. 50–56 (2017)

[13] Karimi, M., Feremans, L., Cule, B., Goethals, B.: Session-based news recommendation using cohesive patterns. In: 2024 IEEE International Conference on Big Data (BigData). pp. 440–447 (2024)

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

[15] Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 337–341 (1999)
[16] Ludewig, M., Jannach, D.: Evaluation of session-based recommendation algorithms. User Modeling and User-Adapted Interaction **28**(4), 331–390 (2018)
[17] Ma, B., Liu, B., Hsu, Y.: Integrating classification and association rule mining. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (1998)
[18] Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: Effective personalization based on association rule discovery from web usage data. In: Proceedings of the 3rd International Workshop on Web Information and Data Management. pp. 9–15 (2001)
[19] Osadchiy, T., Poliakov, I., Olivier, P., Rowland, M., Foster, E.: Recommender system based on pairwise association rules. Expert Systems with Applications **115**, 535–542 (2019)
[20] Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2016)
[21] Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems Handbook, pp. 1–35. Springer (2010)
[22] Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)
[23] Rudin, C., Letham, B., Salleb-Aouissi, A., Kogan, E., Madigan, D.: Sequential event prediction with association rules. In: Proceedings of the 24th Annual Conference on Learning Theory. pp. 615–634 (2011)
[24] Schmidt, M.: The sankey diagram in energy and material flow management: Part ii: Methodology and current applications. Journal of Industrial Ecology **12**(2), 173–185 (2008)
[25] Steck, H.: Calibrated recommendations. In: Proceedings of the 12th ACM Conference on Recommender Systems. pp. 154–162 (2018)
[26] Wu, Z., Li, C., Cao, J., Ge, Y.: On scalability of association-rule-based recommendation: A unified distributed-computing framework. ACM Transactions on the Web **14**(3), 1–21 (2020)
[27] Yap, G.E., Li, X.L., Yu, P.S.: Effective next-items recommendation via personalized sequential pattern mining. In: International Conference on Database Systems for Advanced Applications. pp. 48–64 (2012)
[28] Zhang, M., Hurley, N.: Avoiding monotony: Improving the diversity of recommendation lists. In: Proceedings of the 2008 ACM Conference on Recommender Systems. pp. 123–130 (2008)

Supplementary material

A Algorithm for adaptive rule mining

The main algorithm for efficiently mining rule using adaptive support [14, 15], is shown in Algorithm 1. The main procedure MineRulesAdaptive runs MineRulesAdaptiveLocal for each target item, with varying support levels until between $[k_{min}, k_{max}]$ are discovered. In MineRulesAdaptiveLocal we adopt Apriori++ by computing a co-occurrence matrix and projections at each level for growing rules. The main difference with Apriori++, is that level-wise co-occurrence matrices and projections are local to each consequent item. We remark, that the recommender of Lin, that uses this algorithm for rule mining, produces more pairwise, and long-tail rules than RUDIN or MOB. This has a positive effect on both accuracy, i.e. having the highest recall on two datasets, and reduces popularity bias, i.e. having the highest coverage, and second highest APLT.

Algorithm 1: Adaptive support rule mining for sparse datasets

Input : User-item matrix $R^{|U|\times|I}$, number of rules $[k_{min}, k_{max}]$, maximum pattern size d, confidence threshold γ
Output: Set of rules $X \rightarrow y$

procedure MineRulesAdaptive($R, k_{min}, k_{max}, \gamma$)
  $IID_i, IID_u \leftarrow$ compute inverted index;
  $R \leftarrow \{\}$;
  foreach $s_t \in I$ **do**
    $\theta_c \leftarrow support(s_t)$;
    $R_t \leftarrow$ MineRulesAdaptiveLocal($s_t, \theta_c, \ldots$);
    while $|R_t|$ *not within* $[k_{min}, k_{max}]$ **do**
      if *number of rules* $> k_{max}$ **then**
        Increase θ_c;
      else if *number of rules* $< k_{min}$ **then**
        Decrease θ_c;
    $R \leftarrow R \cup R_t$
  return R
procedure MineRulesAdaptiveLocal($s_t, \theta_c, \ldots$)
  $R_t \leftarrow \{\}$;
  for l ***in*** $1, \ldots, d-1$ **do**
    $S_l \leftarrow$ compute co-occurrences using IID or P_{l-1};
    $X_l \leftarrow$ find frequent patterns $X_l = (s_1, \ldots s_l, s_t)ifsupport(X_l) \geq \theta_c$;
    $R_{t,l} \leftarrow$ find rules $r = s_1, \ldots s_l \rightarrow s_t$ if $conf(r) \geq \gamma$;
    $R_t \leftarrow R_t \cup R_{t,l}$;
    $P_l \leftarrow$ store projections for itemsets X_l;
  return R_t

B Experimental results on each dataset

In Table 7 we show the results for each dataset concerning accuracy, in Table 8 on runtime, in Table 9 on diversity, and in Table 10 on comprehension.

	Dataset	**AR**	**SR**	**IBCF**	**MOB**	**LIN**	**RUDIN**
NDCG@20	30music	0.295	**0.343**	0.205	0.190	0.260	0.131
	EB-NeRD	0.265	**0.310**	0.160	0.178	0.191	0.190
	aotm	0.016	0.018	**0.030**	0.004	0.017	0.009
	clef	0.308	**0.380**	0.270	0.217	0.288	0.317
	rcs15	0.278	**0.289**	0.186	0.186	0.226	0.237
	retailrocket	**0.158**	0.139	0.150	0.103	0.152	0.061
	Average	0.220	**0.246**	0.167	0.146	0.189	0.157
Recall@20	30music	0.416	0.413	0.370	0.356	**0.417**	0.191
	EB-NeRD	0.562	**0.610**	0.411	0.345	0.458	0.465
	aotm	0.034	0.035	**0.062**	0.006	0.037	0.019
	clef	0.653	**0.740**	0.647	0.447	0.670	0.723
	rcs15	0.500	**0.516**	0.399	0.289	0.459	0.473
	retailrocket	0.283	0.249	0.305	0.152	**0.306**	0.094
	Average	0.408	**0.427**	0.366	0.266	0.391	0.328
Precision@20	30music	0.093	**0.197**	0.099	0.034	0.060	0.043
	EB-NeRD	**0.057**	0.016	0.011	0.046	0.034	0.013
	aotm	0.002	0.002	**0.003**	0.000	0.003	0.001
	clef	**0.073**	0.024	0.021	0.060	0.056	0.012
	rcs15	**0.083**	0.011	0.031	0.069	0.054	0.017
	retailrocket	**0.048**	0.016	0.008	0.029	0.039	0.026
	Average	**0.059**	0.044	0.029	0.040	0.041	0.018
F1@20	30music	0.291	**0.371**	0.337	0.300	0.111	0.234
	EB-NeRD	**0.231**	0.032	0.061	0.207	0.132	0.024
	aotm	0.015	0.008	0.010	0.002	**0.047**	0.002
	clef	**0.160**	0.049	0.099	0.158	0.099	0.022
	rcs15	**0.253**	0.022	0.122	0.224	0.121	0.032
	retailrocket	**0.176**	0.047	0.028	0.088	0.139	0.088
	Average	**0.188**	0.088	0.109	0.163	0.108	0.067

Table 7. Results of rule-based RS on NDCG, recall, precision and F1 on six datasets.

	Dataset	AR	SR	IBCF	MOB	LIN	RUDIN
Runtime	30music	20.3s	**19.2**s	66.9s	108.8s	121.3s	19.7s
	EB-NeRD	19.5s	**19.0s**	53.6s	19.1s	177.3s	7653.8s
	aotm	11.2s	10.2s	37.2s	9.4s	71.2s	**5.9s**
	clef	**23.9s**	24.6s	56.4s	30.4s	107.5s	6701.7s
	rcs15	36.0s	**34.8**s	97.9s	38.1s	581.1s	4524.9s
	retailrocket	9.5s	7.7s	20.6s	8.2s	39.4s	**4.4**s
	Average	20.1s	**19.2**s	55.5s	35.7s	183.0s	3151.7s

Table 8. Results of rule-based RS on runtime for six datasets.

	Dataset	AR	SR	IBCF	MOB	LIN	RUDIN
Diversity	30music	0.789	0.850	**0.862**	0.531	0.820	0.851
	EB-NeRD	0.829	0.844	0.866	0.652	**0.870**	**0.872**
	aotm	0.927	0.936	**0.940**	0.437	0.924	0.931
	clef	0.802	0.820	**0.831**	0.668	0.827	0.829
	rcs15	0.862	0.873	**0.888**	0.606	0.884	0.883
	retailrocket	0.858	0.847	0.900	0.579	0.887	**0.883**
	Average	0.845	0.862	0.881	0.579	0.869	**0.887**
Item coverage	30music	66.7%	77.4%	44.3%	53.8%	**78.2%**	11.1%
	EB-NeRD	62.7%	70.2%	**91.5%**	35.8%	78.5%	45.7%
	aotm	65.0%	**84.5%**	22.0%	20.1%	78.0%	1.3%
	clef	87.3%	90.7%	73.4%	85.5%	**96.0%**	89.4%
	rcs15	75.4%	81.7%	34.6%	66.2%	**89.8%**	48.9%
	retailrocket	58.2%	62.7%	49.3%	25.3%	**66.4%**	2.5%
	Average	69.2%	77.9%	52.5%	47.8%	**81.1%**	33.1%
APLT	30music	16.9%	**21.0%**	4.7%	18.2%	19.8%	0.0%
	EB-NeRD	8.5%	8.9%	**20.4%**	8.2%	15.9%	15.3%
	aotm	13.5%	**18.4%**	0.7%	8.9%	18.2%	0.0%
	clef	16.5%	**26.6%**	11.0%	22.4%	17.9%	20.2%
	rcs15	16.4%	19.8%	4.1%	**21.4%**	17.9%	13.0%
	retailrocket	14.8%	**16.9%**	6.7%	4.1%	13.4%	0.0%
	Average	14.4%	**18.6%**	7.9%	13.9%	17.2%	8.1%
User coverage	30music	99.7%	99.5%	**100.0%**	90.1%	**100.0%**	**100.0%**
	EB-NeRD	**100.0%**	100.0%	**100.0%**	91.0%	**100.0%**	**100.0%**
	aotm	100.0%	99.8%	**100.0%**	36.3%	**100.0%**	**100.0%**
	clef	100.0%	100.0%	**100.0%**	96.6%	**100.0%**	**100.0%**
	rcs15	100.0%	100.0%	**100.0%**	95.6%	**100.0%**	**100.0%**
	retailrocket	94.1%	92.5%	99.1%	74.2%	99.1%	**100.0%**
	Average	99.0%	98.6%	99.9%	80.6%	99.9%	**100.0%**

Table 9. Results of rule-based RS on diversity, coverage and popularity bias for six datasets.

	Dataset	AR	SR	IBCF	MOB	LIN	RUDIN
Number of rules	30music	2241.0k	2069.2k	8604.8k	10936.8k	8190.8k	**494.1**k
	EB-NeRD	91.6k	87.8k	379.1k	**7.1k**	112.4k	2088.4k
	aotm	1071.0k	1011.4k	4068.6k	275.5k	4378.0k	**4.0**k
	clef	26.5k	26.2k	127.5k	**17.6k**	26.0k	2706.2k
	rcs15	636.1k	597.3k	2744.2k	**385.2**k	1204.8k	6219.5k
	retailrocket	994.1k	753.6k	2482.9k	320.0k	3137.9k	**9.7**k
	average	843.4k	**757.6**k	3067.9k	1990.4k	2841.7k	1920.3k
Rules of size 3	30music	0%	0%	0%	**88%**	9%	78%
	EB-NeRD	0%	0%	0%	10%	1%	**84%**
	aotm	0%	0%	0%	**65%**	0%	36%
	clef	0%	0%	0%	17%	2%	**95%**
	rcs15	0%	0%	0%	54%	9%	**81%**
	retailrocket	0%	0%	0%	**57%**	24%	28%
	average	0%	0%	0%	49%	8%	**67%**
Rules with pop. consequent	30music	74%	72%	85%	67%	50%	**100%**
	EB-NeRD	44%	45%	27%	32%	6%	**87%**
	aotm	79%	77%	87%	61%	60%	**100%**
	clef	**68%**	59%	57%	62%	42%	60%
	rcs15	43%	39%	74%	15%	7%	**89%**
	retailrocket	73%	73%	80%	84%	54%	**100%**
	average	64%	61%	68%	53%	37%	**89%**
Long-tail rules	30music	18%	17%	11%	**24%**	18%	0%
	EB-NeRD	55%	55%	71%	66%	**85%**	7%
	aotm	14%	14%	10%	**24%**	10%	0%
	clef	27%	**32%**	32%	30%	30%	29%
	rcs15	55%	59%	25%	**83%**	70%	6%
	retailrocket	18%	15%	11%	8%	**22%**	0%
	average	31%	32%	27%	**39%**	39%	7%

Table 10. Results of rule-based RS on comprehensibility, and statistics related to rule size and popularity bias on six datasets.

C Sankey visualization of rule-based recommender systems

Before creating Sankey visualization we link recommendations with rules as follows:

- During training in RuleRec, we call `fit` of the selected algorithm, resulting in potentially millions of rules mined on U_{train}, stored with antecedent, consequent, support, confidence, and rule id.
- During inference, we call `predict`. For each user $u \in U_{test}$ with history p^{obs}, we generate top-n recommendations $\hat{p} = \hat{s}_1, \ldots, \hat{s}_n$, stored with user, item, and score. Additionally, RFER stores the rule id causing each recommendations[2].
- We join recommendations with rules, producing a table of explainable recommendations; optionally, item metadata (e.g., names) can also be included.

From this explainable table we construct a graph for the Sankey visualization:

- We analyze recommendations by rule size, presence of long-tail items[3], and relevance[4], as well as other characteristics of interest to domain experts.
- This information is encoded in a graph with labelled nodes and edges representing counts.
- In the visualization, recommendations are linked to rule groups of size 1 or 2, which connect to subgroups based on the popularity of the antecedent of consequent item ($L \to L$, $L \to P$, $P \to L$, $P \to P$). Within each subgroup, recommendations are further divided into relevant and non-relevant.

In Fig. 3 we show an example of the Sankey visualization of MOB on the 30Music dataset. In total 56k recommendations were generated, and 51k of those recommendations are from rules with a popular item in the antecedent and consequent (74% having a single condition and 26% having two conditions). Finally, the bulk of relevant recommendations (2k) results from these rules.

In Fig. 4 we show an example of the Sankey visualization of LIN on the same dataset. Compared to the MOB, We find much more pairwise rules (99% having a single condition), more rules between long-tail items (19.9% versus 1%), and that there is also a significant increase in the number of relevant recommendations (5k).

[2] For IBCF, this can be a list of rules since aggregation is used.
[3] Items with fewest occurrences, collectively covering 20% of interactions.
[4] A recommended item is in p^{held} for the user.

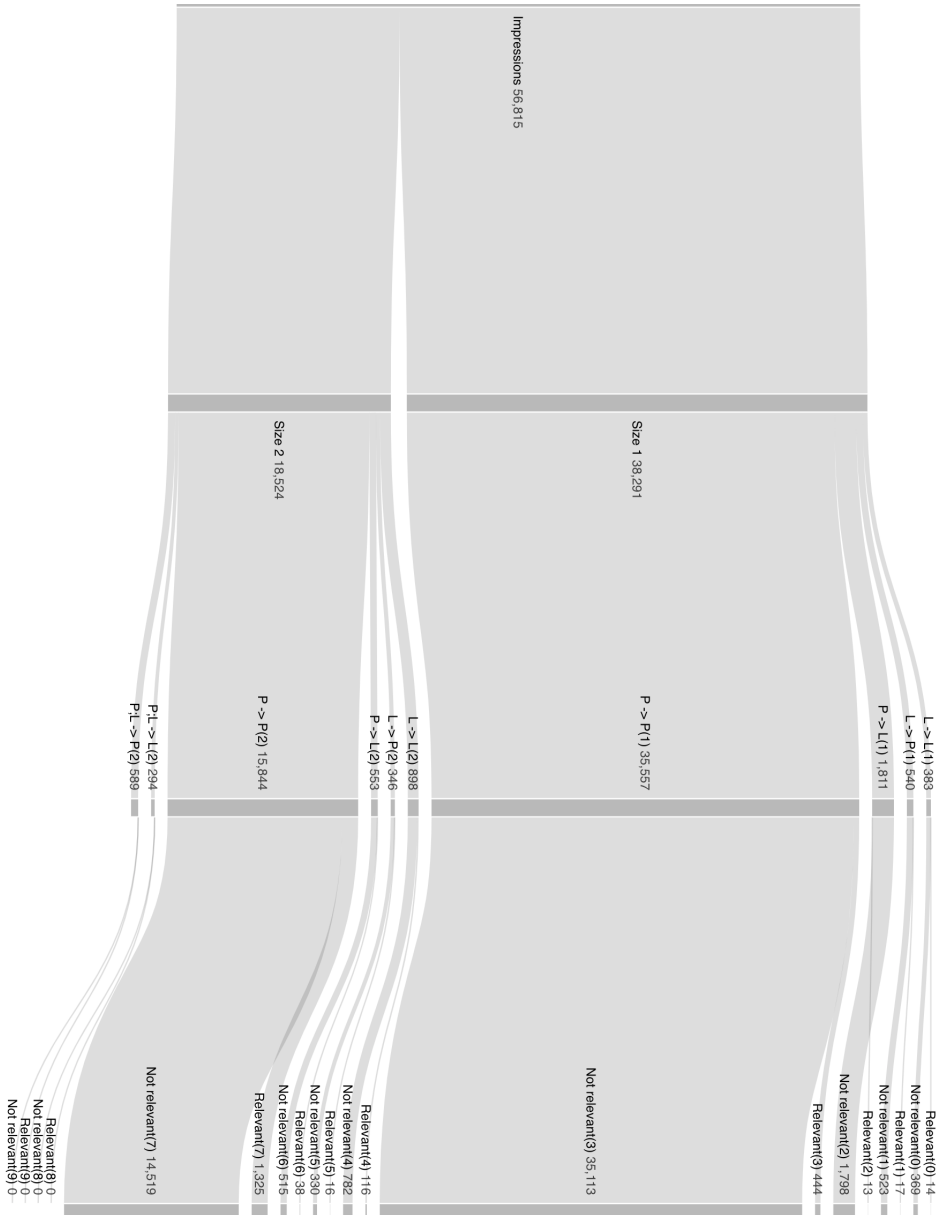

Fig. 3. Sankey diagram analyzing the recommendations generated by the Mobasher et al. method on the 30Music dataset. The flow shows the distribution of recommendations by rule type and their relevance.

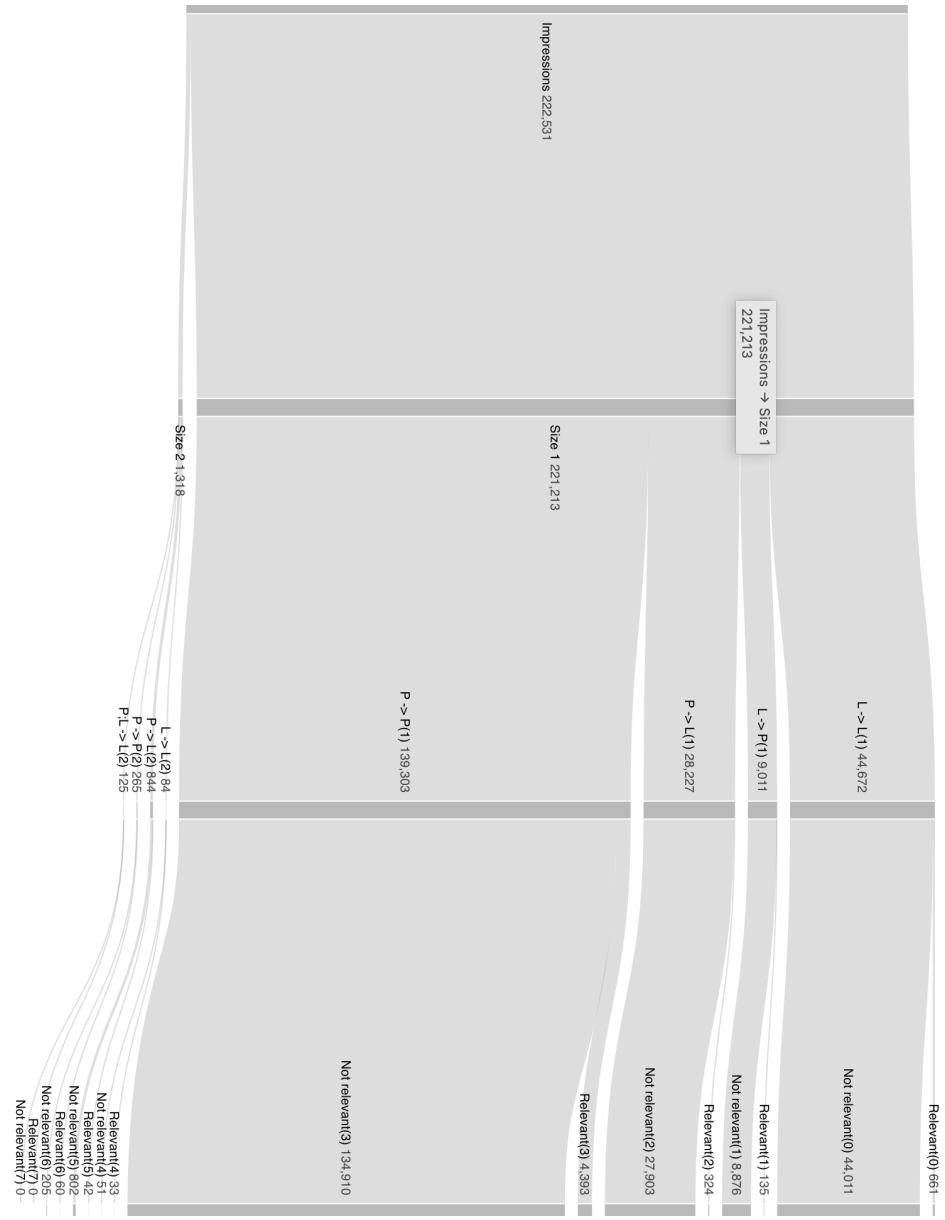

Fig. 4. Sankey diagram analyzing the recommendations generated by the Lin et al. method on the 30Music dataset. The flow shows the distribution of recommendations by rule type and their relevance. We find much more pairwise rules and rules between long-tail items ($L \rightarrow L$) compared to the method by Mobasher et al. on the same dataset.