algorithms

MDPI

Article

Boosting the Learning for Ranking Patterns

Nassim Belmecheri [1,2,*,†], Noureddine Aribi [1,†], Nadjib Lazaar [3,†], Yahia Lebbah [1,†] and Samir Loudni [4,†]

[1] Laboratoire d'Informatique et des Technologies de l'Information d'Oran, Université Oran1, Oran 31000, Algeria; aribi.noureddine@gmail.com (N.A.); ylebbah@gmail.com (Y.L.)
[2] Simula Research Laboratory, 0164 Oslo, Norway
[3] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, Université de Montpellier, CNRS, 34000 Montpellier, France; nadjib.Lazaar@lirmm.fr
[4] Laboratoire des Sciences du Numérique de Nantes, IMT Atlantique, 44307 Nantes, France; samir.loudni@imt-atlantique.fr
* Correspondence: belmecheri.nassim@edu.univ-oran1.dz or nassim@simula.no
† These authors contributed equally to this work.

Abstract: Pattern mining is a valuable tool for exploratory data analysis, but identifying relevant patterns for a specific user is challenging. Various interestingness measures have been developed to evaluate patterns, but they may not efficiently estimate user-specific functions. Learning user-specific functions by ranking patterns has been proposed, but this requires significant time and training samples. In this paper, we present a solution that formulates the problem of learning pattern ranking functions as a multi-criteria decision-making problem. Our approach uses an analytic hierarchy process (AHP) to elicit weights for different interestingness measures based on user preference. We also propose an active learning mode with a sensitivity-based heuristic to minimize user ranking queries while still providing high-quality results. Experiments show that our approach significantly reduces running time and returns precise pattern ranking while being robust to user mistakes, compared to state-of-the-art approaches.

Keywords: interactive data mining; machine learning; learning to rank; active learning; multi-criteria decision making; analytic hierarchy process

Citation: Belmecheri, N.; Aribi, N.; Lazaar, N.; Lebbah, Y.; Loudni, S. Boosting the Learning for Ranking Patterns. *Algorithms* **2023**, *16*, 218. https://doi.org/10.3390/a16050218

Academic Editor: Frank Werner

Received: 19 March 2023
Revised: 5 April 2023
Accepted: 18 April 2023
Published: 24 April 2023

1. Introduction

Data mining is a field that focuses on discovering relevant information from data and transforming it into useful knowledge. One of its significant sub-fields, pattern mining, involves searching and enumerating significant patterns in data. In recent years, there has been a notable shift in the pattern mining community from efficiency-based approaches to methods that can extract more meaningful patterns. Unfortunately, obtaining relevant results with traditional pattern mining methods can be challenging and time consuming. There are two main issues: first, a vast number of patterns are discovered, many of which are redundant. Second, the preferences and domain experts' background knowledge is not considered. The significance of incorporating user preferences and knowledge was initially highlighted by Silberschatz and Tuzhilin [1]. The fundamental concept is to represent the user's preferences using objective interestingness measures.

However, as noted in [2], using objective quality measures has limited practical utility since interestingness is subject to the particular user and task at hand. The interestingness of a pattern is determined solely based on the data.

In recent years, there has been a growing emphasis on user-centric, interactive, and anytime pattern mining [3–5]. This paradigm highlights the need to quickly present to the user some patterns that are likely to be interesting to them, and to obtain feedback that will influence subsequent iterations of the interactive mining process. An important aspect of this framework is the ability to learn user-specific functions for ranking patterns from feedback. Although the idea of learning user-specific ranking functions was first

investigated in [6], it has been extended in recent years by Boley et al. [4] and Dzyuba et al. [3,5] in the context of interactive pattern mining. Additionally, Xuguang Bao et al. [7] and Liang Chang et al. [8] adapted interactive data mining for extracting interesting spatial co-location patterns. These approaches leverage standard machine learning techniques to learn weighted vectors based on selected pattern features, such as items, transactions, and length.

In [3], a linear ranking function is learned using support vector machines (`RankingSVM`). In [4,5], the authors proposed using stochastic coordinate descent (SCD) [9] to learn a logistic function. While these methods enable the exploitation of user feedback to learn ranking functions, they also increase the computational cost of the learning task, particularly when the number of pattern–measure pairs used for ranking increases. Nonetheless, a crucial aspect of this framework is the ability to quickly present a set of patterns to the user and focus on what may be of interest using interestingness measures.

In this paper, we address the challenge of learning pattern ranking functions as a multi-criteria decision-making problem. To this end, we propose to use a weighted linear function to aggregate all the individual measures into a global ranking function. Linear functions have been widely adopted in learning to rank models because they are simple, interpretable, and efficient, yet can still effectively capture the underlying relationships between input features and relevance labels [10,11].

We propose a learning algorithm that is both fast and scalable, which enables us to maintain the expected weights of measures. Our approach leverages the analytical hierarchy process (AHP) [12] and a set of user-ranked patterns to learn the weighting vector of measures. Specifically, we seek to maximize the correlation between the unknown user's ranking function and the learned AHP-based ranking function. We previously introduced this approach in our work [13], where we presented the initial results of the passive version of the algorithm.

This study presents a new learning algorithm that can operate in both active and passive modes, specifically designed for pattern ranking. To improve performance in active mode, a sensitivity-based heuristic is proposed to select more interesting patterns. The algorithm is explained in detail using a running example. The approach was validated on large datasets using three different user simulations in both passive and active modes. The results indicate that the proposed approach is robust and can account for user mistakes and changes in ranking criteria in interactive learning scenarios. We conducted experiments to evaluate the effectiveness and efficiency of our approach using association rule mining as a case study. We have considered the following research questions:

1. Can we efficiently reveal user-specific preferences over all patterns from a sample of ranked patterns with a weighted linear function? What is the required input data to make the learning algorithm more effective?
2. How does our new learning method based on AHP compare to the state-of-the-art method Rank-SVM, for learning user-specific ranking functions?
3. To what extent is our sensitivity heuristic an appropriate choice for query selection in the context of active learning?
4. How robust is our approach and does it keep the learning consistent when the user makes mistakes?

Our experiments on various datasets demonstrated that our approach significantly reduces the running time while accurately ranking patterns compared to state-of-the-art methods.

The paper is organized as follows: In Section 2, we review the related work in pattern mining and ranking functions. In Section 3, we provide some necessary preliminaries for our approach. Section 4 formulates the problem we tackle and describes our proposed method. A running example is presented in Section 5 to illustrate our approach. In Section 6, we report and discuss the empirical results of our experiments. Finally, we conclude the paper in Section 7.

2. Related Work

Iterative/user-centric pattern mining has gained significant attention in recent years, where the user is involved in the mining process to improve the mining results by providing feedback. This approach is particularly useful in domains such as medical diagnosis or fraud detection, where the user's domain knowledge is essential.

To make this process more efficient and effective, researchers have been exploring methods to learn ranking functions that can quickly present patterns to the user based on their interests. This problem, known as object ranking, has been studied extensively in machine learning and data mining communities.

One common approach to object ranking is to learn a ranking function based on pair-wise comparisons. However, these methods can be time-consuming and expensive since they require a large number of pair-wise comparisons to learn a reliable ranking function. To address this challenge, researchers have explored alternative methods such as multi-criteria decision making. In our approach, we use the analytical hierarchy process (AHP) to learn the measures' weighting vector, which maximizes the correlation between the user's ranking function and the learned AHP-based ranking function. In [6], the authors proposed a user-centric approach to mining interesting patterns based on the user's feedback. They developed a log-linear model for itemsets that considers the user's prior knowledge about the items and their relationships. The model was trained using `RankingSVM` to learn a ranking function that assigns a score to each itemset. The authors also proposed a belief model that exploits belief probabilities assigned to transactions for more complex patterns. The authors demonstrated that their approach outperformed existing methods in terms of accuracy and efficiency for complex patterns where traditional methods may fail to capture the user's preferences.

Dzyuba et al. [3] extended the approach proposed by Xin et al. [6] and used `RankingSVM` for active preference learning to rank patterns. However, these methods suffer from computational inefficiencies, which limit their scalability on large datasets. In contrast, the AHP-based approach proposed in our work has a linear time complexity and can efficiently rank a large number of patterns.

In [14], the authors proposed a user-centric generic framework (PRIIME) to learn ranking functions. To achieve this, regression techniques based on neural networks were adopted. They assigned a score to each pattern based on user feedback and trained a regression model on labeled patterns and their corresponding scores. The model predicts the score of new patterns based on their features and user feedback. The proposed method outperformed existing methods in terms of accuracy and efficiency and can effectively mine interesting patterns and provide valuable insights to the user.

In their study, Lee and colleagues [15] investigated changes in the citation patterns of academics in response to the "evaluation environment" that emerged in academia with the rise of the World University Rankings (WURs). The authors analyzed papers published in two higher education journals across three periods: pre-WURs (1990–2003), WURs implementation (2004–2010), and adaptation to WURs (2011–2017). To compare and rank first-citation speeds across these periods, they used the non-parametric Kaplan–Meier method [16]. In [17], the authors investigated the factors that influence website rankings of search engine result pages, highlighting the top contributors to better rankings. The study is divided into two parts: a literature review and empirical research. The literature review identified 24 factors that affect website rankings, with the most common being backlinks, social media support, keywords in the title tag, website structure, size, loading time, domain age, and keyword density. In the empirical research, the authors analyzed the top 15 Google results for three search phrases, taking into account the factors identified in the literature review. The significance of each factor was measured using Spearman correlation. The findings revealed that the top factors contributing to higher rankings are the existence of an SSL certificate on the website, keywords in the URL, the number of backlinks, text length, and domain age. These results do not perfectly align with the

literature review, suggesting a potential gap between academic understanding of SEO factors and their real-world application.

The study by Zimmer et al. [18] investigated the impact of filter bubbles and echo chambers [19] on the spread of fake news, and analyzed the information behavior patterns of individuals who reacted to such news. The study aimed to answer two research questions: (1) whether machines facilitate the dissemination of fake news through the automatic creation of filter bubbles, and (2) whether echo chambers of fake news are artificially created. To accomplish this, the authors used a case study approach that involved both quantitative and qualitative content analysis of online comments and replies on a blog and Reddit, and an examination of social media ranking algorithms. The findings suggest that filter bubbles exist, but users' information behavior primarily feeds them, thus amplifying their behavioral patterns, while selective exposure to information may result in confirmation bias, other cognitive structures such as non-argumentative behavior, off-topic behavior, denial, moral outrage, meta-comments, insults, satire, and the creation of new rumors, also contribute to the various reactions to fake news. In [7], an efficient interactive approach was proposed for users to discover their preferred co-location patterns. The approach consists of a flexible interactive framework, an ontology-based similarity measure between co-location patterns, a pattern filtering model to express user preferences, and a pruning scheme to reduce the number of outputs.

The work proposed in [20] presents two significant contributions to the field of database management systems (DBMS). Firstly, the authors developed an ontology and methodology that describe essential features and aspects of the DBMS domain, including various paradigms, query languages, platforms, and specific contexts. This model is populated with significant individuals (actual DBMSs) by leveraging existing knowledge from DBpedia and Wikidata, which are free and open machine-processable knowledge bases. This approach improves the view of the DBMS domain and has the potential to enhance information integration and search capabilities by creating a specific knowledge graph. Secondly, the authors designed and developed two knowledge-based web applications that provide information about DBMSs according to the user's needs and preferences. These web systems, which the paper describes, serve as examples of using semantic web technologies and the proposed knowledge model for educational and pragmatic purposes.

In [8], the authors proposed IDMBS (interactive data mining based on support vector machine), an interactive mining system that uses SVM to discover user-preferred co-location patterns. The system includes a filtering algorithm to select patterns, which are then annotated by the user, and an SVM model to train on these patterns and discover additional user-preferred co-location patterns.

Our proposed approach tackles the problem of learning a user-specific function for ranking patterns as a multi-criteria decision-making problem using AHP. Moreover, our method remains fast when the number of patterns used for ranking increases. Our approach is novel in that it employs a multi-criteria decision-making methodology (AHP) that, to the best of our knowledge, has not been used before. Our method is capable of scaling when dealing with large datasets and is resilient to user mistakes.

Figure 1 depicts the overall methodology of our approach. Our approach operates in passive learning mode if a user ranking already exists (i.e., $S \neq \emptyset$), otherwise it runs in active mode by following the main steps. First, the sample patterns component selects a subset of patterns. Then, the pattern selection component selects the most sensitive pairs of patterns. Next, a query is generated and presented to the user to be ranked. Finally, the learning weights component computes the weights of the linear function by leveraging the user feedback. At the end of our learning process, the method provides the current weights of the linear function.

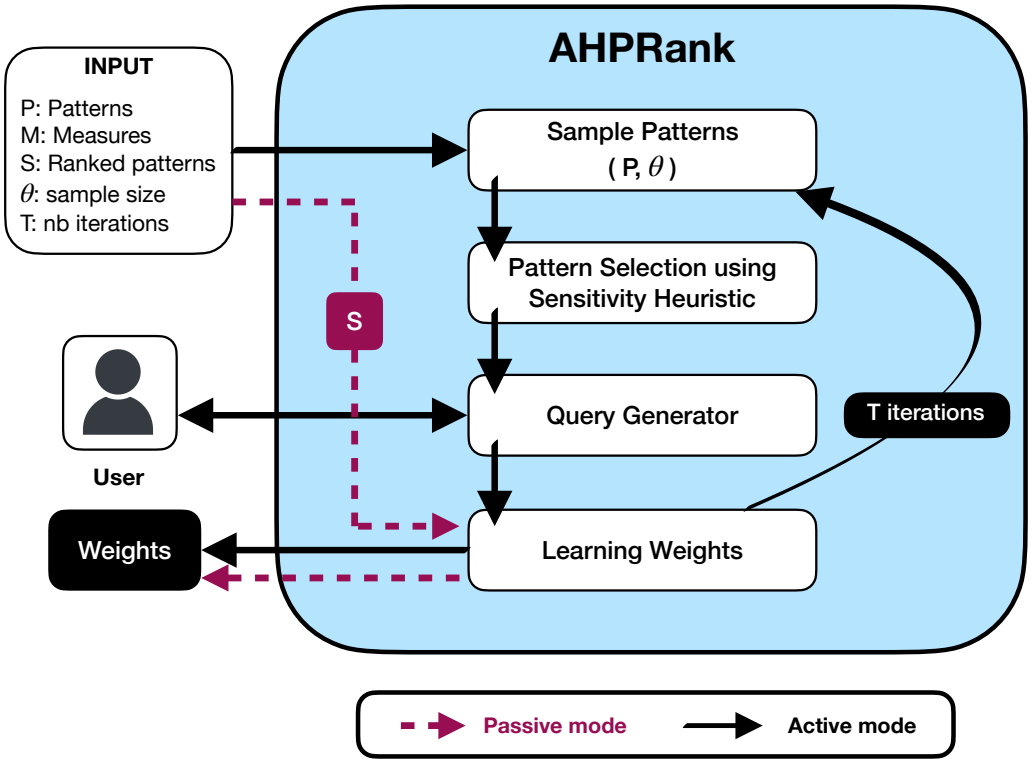

Figure 1. Methodology of the AHP-based learning to rank.

Most of the related work uses SVM-based algorithms to learn a user-specific function. According to [21], the worst-case time complexity of the SVM algorithm varies between k^2 and k^3 on the number of samples k; whereas our approach is linear on k (see Section 4.6).

3. Background

In this section, we provide the necessary background information about pattern mining, interestingness measures commonly used to evaluate pattern quality, and multi-criteria decision making using the AHP method.

3.1. Pattern Mining and Interestingness Measures

Data of all types can be represented as a triplet $(\mathcal{O}, \mathcal{A}, \mathcal{L})$, where $\mathcal{O}$ is a set of objects, $\mathcal{A}$ is a set of attributes, and $\mathcal{L}$ is the relational language used to express objects with respect to a set of attributes. This framework is inspired by the concepts of inductive databases [22,23]. The language $\mathcal{L}$ can be used to define various types of datasets. For instance, when $\mathcal{L}$ is a binary relation between attributes and objects (i.e., $\mathcal{L} \subseteq \mathcal{O} \times \mathcal{A}$), we obtain the classical itemset mining framework. When $\mathcal{L}$ expresses each object as a sequence of attributes, we obtain a sequential dataset. Finally, when $\mathcal{L}$ expresses each object as a graph where nodes and edges are labeled, we obtain a dataset of graphs.

The goal of data mining is to uncover meaningful patterns or regularities between objects within large datasets. To evaluate the quality of these patterns and determine the most relevant ones, interestingness measures are employed.

The field of data mining is actively researching how to measure the interestingness of discovered patterns with respect to user-specific preferences. Nine concepts have been introduced to assess the interestingness of patterns: conciseness, generality, reliability, peculiarity, diversity, novelty, surprisingness, utility, and applicability [24]. Some of these concepts are correlated, such as surprisingness and applicability, while others are conflicting, such as generality and pecularity, and some are independent, such as novelty and utility. Depending on the user's preferences, they may value certain concepts over others.

To satisfy these preferences, a wide range of measures and aggregation techniques have been proposed [25].

3.2. Analytical Hierarchy Process (AHP)

AHP is a popular multi-criteria decision-making method created by Saaty [12]. It involves breaking down the decision problem into a hierarchical structure and deriving criteria weights from the decision maker using pair-wise comparisons, rather than relying solely on numerical values.

After constructing a decision problem hierarchy using the AHP method, the important weights of the criteria can be determined through the following three main steps:

1. To estimate the relative importance of m criteria at a given level, the user is asked to make pair-wise comparisons between them. These comparisons are used to construct a pair-wise comparison matrix $\mathbf{A}$ of size $m \times m$, where a_{ij} represents the importance of criterion i relative to criterion j. If $a_{ij} > 1$, criterion i is considered more important than criterion j.
   To ensure consistency, the matrix is filled such that $a_{ij} = 1/a_{ji}$ and $a_{ii} = 1$. AHP uses a scale from 0 to 9 to quantify the preference degree, where 1 indicates indifference and 9 indicates absolute preference.
2. After constructing the pair-wise comparison matrix at a given level, various mathematical techniques proposed in the literature can be used to compute the weight vector $w = (w_1, \ldots, w_m)^T$. These techniques include the eigenvector method (EVM) [26,27] and distance-based minimization methods, such as the least squares method, logarithmic least squares method, weighted least squares method, logarithmic least absolute values method, and singular value decomposition [28–30]. The EVM method is the most commonly used and computes the weight vector w by solving the following characteristic equation:
$$\begin{cases} \mathbf{A} \cdot w = \lambda_{max} \cdot w \\ w^T \mathbf{1} = 1 \end{cases} \quad (1)$$
   where $\mathbf{A}$ is the pair-wise comparison matrix, λ_{max} is the highest eigenvalue of $\mathbf{A}$, and $\mathbf{1} = (1, \ldots, 1)^T$. The constraint $\sum_{i=1}^{m} w_i = 1$ is added to help the solving process and avoid infinite solutions. Note that, if $\mathbf{A}$ is positive then the highest eigenvalue is real (i.e., $\lambda_{max} \in \mathbb{R}$) [30]. Interestingly, Saaty [31] mentioned that the perfect (When the components of $\mathbf{A}$ are exactly obtained as the ratio between weights.) normalized eigenvector w satisfies the following relation:
$$a_{ij} = \frac{w_i}{w_j}, \forall i, j = 1 \ldots m. \quad (2)$$
   In this case, $\lambda_{max} = m$ only when the comparison matrix is perfectly consistent; otherwise, λ_{max} is greater than m. This equation explains why some researchers prefer distance-based methods, which aim to minimize the distance between $(a_{ij})_{m \times m}$ and $(wi/w_j)_{m \times m}$ directly. However, this leads to a non-linear and difficult-to-solve problem. Therefore, in this paper, we use the EVM method instead of distance-based methods.
3. Since AHP does not guarantee the transitivity of the user's preferences, the pair-wise comparison may result in inconsistencies. To detect these inconsistencies, a consistency check is used by calculating the consistency ratio (CR), defined as $CR = \frac{CI}{RI}$, where RI is a constant obtained from the random consistency index table of AHP. The consistency index of the EVM method is computed as follows:
$$CI = \frac{(\lambda_{max} - n)}{n - 1} \quad (3)$$
   The reliability of the computed weighting vector w (solution to Equation (1)) is assessed by checking whether the CR value is less than 0.1.

It is common for hierarchies to contain multiple levels of criteria, especially when there are more than nine criteria. To handle this, it is beneficial to establish a hierarchy based on existing dependencies between the criteria. This can help optimize the decision-making process and reduce the number of required pair-wise comparisons. For further information on this topic, we recommend referring to [30]. However, in this paper, we will not dive into this aspect. In Example 1, we illustrate the AHP process.

Example 1. *Consider a decision problem involving three criteria, where we aim to determine their relative importance in achieving an overall goal. To do so, we construct a pair-wise comparison matrix* $\mathbf{A}(a_{ij})_{3\times 3}$*, where each element* a_{ij} *represents the relative importance of criterion i compared to criterion j. The matrix is given as follows:*

$$\mathbf{A} = \begin{pmatrix} 1 & 1/2 & 1/4 \\ 2 & 1 & 1/2 \\ 4 & 2 & 1 \end{pmatrix}$$

To determine the weights of the criteria, we can employ the EVM method, which involves solving the equation $\mathbf{A}w = \lambda_{max}w$*. The resulting weight vector is given by* $w = (w_1, w_2, w_3) = (1/7, 2/7, 4/7)$*, where each component represents the weight of the corresponding criterion.*

4. AHP-Based Learning Approach for Ranking Patterns

We aim to develop a ranking function that combines a set of interestingness measures while also taking into account user-specific preferences.

4.1. Problem Statement: Learning Pattern Rankings

In this section, we demonstrate that learning a ranking function that incorporates a set of interestingness measures to approximate user-specific preferences can be formulated as a multi-criteria decision-making problem. In this context, the criteria correspond to the interestingness measures.

The objective of this task is to learn a ranking function for patterns based on a sample of ordered patterns. To accomplish this, we adopt the approach of ordered feedback, where users are asked to provide a total order over a small set of patterns based on their subjective interestingness. This approach is inspired by previous research [32].

Let $\triangleright$ be a binary preference relation between patterns. A ranking function based on $\triangleright$, denoted by $r_{\triangleright}(.)$, takes as input a subset of patterns $\mathcal{P} \subseteq \mathcal{L}$ and returns as output a permutation $S = \langle P_{r_1}, P_{r_2}, \ldots, P_{r_n} \rangle$ of $\mathcal{P}$, such that $|\mathcal{P}| = |S|$ and $\forall i, j \in [1, n], i < j$, $P_{r_i} \triangleright P_{r_j}$.

We aim to learn the user ranking function denoted by $r_{\triangleright_u}$, which is based on the user preference relation $\triangleright_u$. To achieve this, we use a set of interestingness measures $\mathcal{M} = \{M_1, \ldots M_m\}$ and their corresponding ranking functions $r_{\triangleright_{M_1}}, \ldots, r_{\triangleright_{M_m}}$. The rank of pattern $P_j \in \mathcal{P}$ with respect to the interestingness measure M_i is denoted by $rank_{M_i}(P_j)$. Here, $(P_k \triangleright_{M_i} P_l)$ means that $M_i(P_k) \geq M_i(P_l)$.

To frame the ranking problem as a learning problem, we need to find the weights w_i of a linear aggregation function over the measures from a set of ranked patterns that maximizes the correlation with the user ranking function $r_{\triangleright_u}$. Specifically, we aim to find the coefficients of the following function by solving a multi-criteria optimization problem:

$$f_w(P) = \sum_{i=1}^{m} w_i \,.\, M_i(P) \tag{4}$$

Our approach leverages user feedback to perform pair-wise comparisons between measures and estimate their relative importance based on individual measure scores. The pair-wise comparison results are then aggregated into a matrix, which is used by the analytic hierarchy process (AHP) to compute the weights w_i for each measure.

4.2. Comparing a Single Measure to a User Ranking

Let $S = \langle P_1, \ldots, P_n \rangle$ be a user ranking of patterns. Let $\mathcal{M} = \{M_1, \ldots, M_m\}$ be a set of interestingness measures for patterns. To evaluate how well a particular measure performs in terms of overall ranking accuracy compared to a user ranking, we use Kendall's W concordance coefficient [33]. For a measure M_i and a user ranking S, we define $K_i(S) \in [0, 1]$ as the Kendall's W coefficient between the ranking obtained using M_i and the user ranking S, where a value of 1 indicates perfect concordance and a value of 0 indicates no concordance.

$$K_i(S) = \frac{3\alpha}{|S|^3 - |S|}, \quad \alpha = \sum_{\ell=1}^{|S|} (R_\ell - \bar{R})^2, \tag{5}$$

where $R_\ell = rank_{M_i}(P_\ell) + rank_u(P_\ell)$ with $P_\ell \in S$, and $\bar{R} = \frac{1}{|S|} \sum_{\ell=1}^{|S|} R_\ell$.

4.3. Comparing Two Measures According to a User Ranking

Let M_i and M_j be two measures to be compared based on a given user ranking S. We can compute the gap $\Delta_{i,j}(S) = K_i(S) - K_j(S)$ to determine how much M_i is closer to the user ranking S than M_j. If $\Delta_{i,j}(S) \geq 0$, then M_i is closer, otherwise, the converse is true. When considering a set of user rankings $\mathcal{S} = \{S_1, \ldots, S_n\}$, comparing two measures M_i and M_j on $\mathcal{S}$ involves estimating the measure closest to $\mathcal{S}$ by averaging the gaps over all the user rankings in $\mathcal{S}$: $\Delta_{i,j} = \frac{1}{n} \sum_{S_k \in \mathcal{S}} \Delta_{i,j}(S_k)$.

4.4. Learning Weights Process

Algorithm 1 implements the learning process that computes a weight vector w (a weight for each measure) by exploiting the pair-wise comparisons of measures. It takes as inputs a vector of pair-wise comparisons Δ and its corresponding index l, a user ranking S, and a set of measures $\mathcal{M}$. The learning step is performed at line 2 where for each pair of measures, we first compute the Kendall W coefficient for each measure i with respect to the user ranking S ($K_i(S)$). Then, we compute the gap between the pairs.

Algorithm 1: `learnWeights`$(\langle \Delta, l \rangle, S, \mathcal{M})$.

In : Set of measures $\mathcal{M} = \{M_1, \ldots, M_m\}$; user ranking $S = \langle P_1, \ldots, P_n \rangle$;
In Out: Pair comparisons and index parameter $\langle \Delta, l \rangle$;
Out : Weight vector w;

1 $\mathbf{A}[i, j] \leftarrow 1, \forall i, j \in \{1, \ldots, m\}$;
2 **foreach** $M_i, M_j \in \mathcal{M} : i < j$ **do**
3 $\quad \Delta_{i,j} \leftarrow \frac{l}{l+1} \Delta_{i,j} + \frac{1}{l+1} (K_i(S) - K_j(S))$;
4 **end**
5 **foreach** $\Delta_{i,j} \in \Delta$ **do**
6 $\quad$ **if** $\Delta_{i,j} < 0$ **then**
7 $\quad\quad$ scale $\Delta_{i,j}$ to $(-9..-1)$; $\mathbf{A}[j, i] \leftarrow |\Delta_{i,j}|$; $\mathbf{A}[i, j] \leftarrow 1/|\Delta_{i,j}|$;
8 $\quad$ **else**
9 $\quad\quad$ scale $\Delta_{i,j}$ to $(1..9)$; $\mathbf{A}[i, j] \leftarrow \Delta_{i,j}$; $\mathbf{A}[j, i] \leftarrow 1/\Delta_{i,j}$;
10 **end**
11 $w \leftarrow solve \left(\begin{array}{l} \mathbf{A} \cdot w = \lambda_{max} \cdot w \\ \text{subject to } \sum_{j=1}^{m} w_j = 1, \quad w_j > 0, \forall j = 1 \ldots m. \end{array} \right)$
12 **return** w;

Algorithm 1 implements the learning process that computes a weight vector w (a weight for each measure) by exploiting the pair-wise comparisons of measures. It takes as inputs a vector of pair comparisons Δ and the corresponding index l, a user ranking S, and a set of measures $\mathcal{M}$. The function incrementally updates Δ by taking into account the previous l user rankings at line 3. At line 5, the AHP matrix is constructed by scaling the $\Delta_{i,j}$ comparisons to the AHP values between $1/9$ and 9. Once constructed, the criteria weight vector w is returned in line 12 by solving the minimization problem described in line 11, using the eigenvector method (EVM) [30].

After learning the weight vector w, we use it to compute the score of a given pattern P_i by applying a weighted aggregation function:

$$g_w(P_i) = \sum_{M_i \in \mathcal{M}} w_i \; scale(M_i(P)) \tag{6}$$

The *scale* function is used to normalize the interestingness values of the different measures to a $[0,1]$ scale so that they can be combined using the weighted aggregation function.

4.5. AHPRank Algorithm

We propose Algorithm 2 to implement the `AHPRank` function (Algorithm 1) to learn a weight vector in a passive/active mode. The algorithm takes as input a set of measures $\mathcal{M}$, a set of user-ranked patterns $\mathcal{S}$, and a triplet of parameters $\langle \mathcal{P}, \theta, T \rangle$. Here, $\mathcal{P}$ is a collection of patterns, θ is a sample parameter used in the query generator for active learning, and T is the maximum number of iterations used as a stopping criterion in the algorithm.

`AHPRank` initializes the vector of pair comparisons to zero in line 1. If `AHPRank` is called with a non-empty $\mathcal{S}$, it runs in passive mode by calling the `learnWeights` function on the given user-ranked patterns $\mathcal{S}$ (lines 3–6). This produces a weight vector w. Otherwise, it runs in active mode by submitting T queries to the user (lines 7–15). In the active mode, the user is asked to rank a subset of patterns proposed by a query generator. Our query generator is based on a heuristic that exploits the quality of the weights learned from the previous iterations. The active learning mode follows the following iterative steps:

1. As a first step, a subset of patterns is sampled from the global set of patterns $\mathcal{P}$ using a random sampling approach. At line 10, the function `SamplePatterns` is called, which returns a sample $\mathcal{P}'$ of θ patterns.
2. The `SensitivityBasedHeuristic` function is used to select a pair of patterns. This heuristic is designed to select a high-quality pair of patterns to accelerate the convergence of the learning process (see Algorithm 3 in Section 4.5 for details).
3. Interacting with the user obtains feedback through a ranking query (line 12).
4. AHP-based learning of the user ranking function is performed by calling the `learnWeights` function on the user-provided ranking query (line 13).

The proposed method is designed to handle large sets of patterns, which can be computationally expensive to process. To address this issue, we introduce two steps in the active learning mode. First, we randomly sample a subset of patterns from the global set $\mathcal{P}$, which reduces the complexity of the second step as well as introduces a diversity factor. Second, we use a sensitivity-based heuristic to select pairs of patterns that are of high-quality and diverse.

It is important to determine when to stop the learning process. Our method allows for a custom stopping criterion, but we also consider the case where a fixed number of iterations T is used. This simulates the scenario where the user can stop at any time based on their satisfaction with the learned ranking function.

Algorithm 2: `AHPRank`$(\mathcal{M}, \mathcal{S}, \langle \mathcal{P}, \theta, T \rangle)$.

In :Set of measures $\mathcal{M} = \{M_1, \ldots, M_m\}$;
Passive mode ($\mathcal{S} \neq \emptyset$): Set of user-ranked patterns $\mathcal{S} = \{S_1, \ldots, S_n\}$;
Active mode ($\mathcal{S} = \emptyset$): Collection of patterns $\mathcal{P}$; sample size θ ;
number of iterations T;

Output: w^t : weighting vector at the last iteration t;

foreach $\Delta_{i,j} \in \Delta$ **do** $\Delta_{i,j} \leftarrow 0$;
if $\mathcal{S} \neq \emptyset$ **then**
  foreach $S_k \in \mathcal{S}$ **do**
    $w^t \leftarrow$ `LearnWeights`$(\langle \Delta, k \rangle, S_k, \mathcal{M})$; `// Learn weights`
  end
end
else
  $w^0 \leftarrow \mathbf{1}$; `// Weight vector`
  for $t = 1, 2 \ldots T$ **do**
    $\mathcal{P}' \leftarrow$ `SamplePatterns`$(\mathcal{P}, \theta)$; `// Step 1`
    $Q_t \leftarrow$ `SensitivityBasedHeuristic`$(\mathcal{P}', w^{t-1}, \mathcal{M})$; `// Step 2`
    $S_t^* \leftarrow$ `AskRanking`(Q_t) ; `// Step 3`
    $w^t \leftarrow$ `LearnWeights`$(\langle \Delta, t \rangle, S_t^*, \mathcal{M})$; `// Step 4`
  end
end
return w^t

Algorithm 3: `Sensitivity` $-$ `basedheuristic`$(\mathcal{P}, w, \mathcal{M})$.

In :Collection of patterns $\mathcal{P}$; weight vector w; set of measures $\mathcal{M}$;

Output: *Pair* : Selected pair of pattern

$\mathcal{P}' \leftarrow$ `Sort`$(\mathcal{P}, w)$; `// sort` $\mathcal{P}$ `according to` g_w
Pair $\leftarrow \emptyset$;
$\sigma_{min} \leftarrow +\infty$;
for $i \in 1..|\mathcal{P}'| : P_i, P_{i+1} \in \mathcal{P}'$ **do**
  $score \leftarrow \sigma(P_i, P_{i+1})$;
  if $score < \sigma_{min}$ **then**
    $\sigma_{min} \leftarrow score$;
    Pair $\leftarrow \{P_i, P_{i+1}\}$;
  end
end
return *Pair*;

A Pattern Selection Heuristic for AHPRank

Active learning is a challenging task since each learning step needs to enhance the learned function. The selection of appropriate patterns for the user to rank is the key to incrementally improve the learning process. However, generating the ideal set of patterns is a computationally demanding task and is known to be NP-hard [34]. To address this issue, we propose a heuristic (Algorithm 3) based on sensitivity analysis for AHP models [35]. Our heuristic aims to select a pair of patterns that are close in terms of their overall score predicted by the learned function relative to the sum of the measure gaps:

$$\sigma(P_i, P_j) = \left| \frac{g_w(P_i) - g_w(P_j)}{\sum_{\ell=1}^{m} (M_\ell(P_i) - M_\ell(P_j))} \right|; i, j \in 1 \ldots k. \tag{7}$$

A successful aggregation should have the property that if two patterns are close in overall predicted score, they should also be close in their measure values. Thus, the algorithm will be enforced into learning the correct preferences of the patterns when having the lowest σ value since the selected pair is surely close in terms of g_w values and very distant on the values of the measures. Example 2 illustrates the selection process of the Algorithm 3.

Example 2. *Suppose we have two patterns P_1 and P_2 with four interestingness measures $\mathcal{M} = M_1, M_2, M_3, M_4$. Let $P_1 = \langle 0.1, 0.2, 0.3, 0.7 \rangle$, $P_2 = \langle 0.7, 0.3, 0.2, 0.1 \rangle$, and let the current learned weight vector be $w = \langle 0.25, 0.25, 0.25, 0.25 \rangle$. At this stage, $g_w(P_1) = 0.26$ and $g_w(P_2) = 0.26$. Although P_1 and P_2 have the same overall score, their measure values differ. This pair of patterns is an interesting candidate with $\sigma(P_1, P_2) = 0$, where retrieving the user's preference between them can improve the learning process and bring it closer to the user's ranking.*

4.6. Complexity Analysis

Proposition 1 (Time complexity). *Algorithm 2 has a time complexity of $O(n\ k)$ in passive learning and $O(n)$ in active learning when the number of interestingness measures is bounded.*

Proof. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a set of n user rankings, where k is the size of the largest ranking S_t and m is a bounded number of measures in $\mathcal{M}$. Once the pair-wise comparison matrix **A** is built, the preference vector of weights w can be computed using various mathematical techniques, including the eigenvector-based method (EVM) [30]. The worst-case time complexity of the EVM approach is $O(m^3)$ [36], which is of constant time complexity $O(1)$ since m is constant. The time complexity of computing the matrix at line 3 of the `learnWeights` function is $O(m\ n\ k)$ since Kendall's W is in $O(k)$ (see Definition (5)). The complexity of lines 2, 5, and 11 in `learnWeights` are, respectively, $O(n\ m^2 + m\ n\ k)$, $O(m^2)$, and $O(m^3)$. Notice that in AHP, it is demonstrated in [37] that the number of criteria is recommended to be no more than seven ± 2. Thus, assuming m is a small and fixed number of measures, we can set $m \leq 9$ and obtain an asymptotic quadratic complexity of $O(n\ k)$ when AHP is in passive mode. In other words, the approach takes $O(k)$ for each ranking in the given $\mathcal{S}$. When AHP is in active mode, where queries are pairs of patterns ($k = 2$), the time complexity is linear on the number of queries submitted to the user, i.e., $O(n)$. □

It is important to emphasize that in practice, the set $\mathcal{S}$ is typically reduced to a single large ranking in passive learning mode ($n = 1$ and a large k), and to a set of n queries consisting of pairs of patterns in active learning mode (large n and $k = 2$). As a result, `AHPRank` can run in linear time, with a complexity that is linear in k for passive mode and linear in n for active mode. This low complexity makes `AHPRank` a fast approach for learning a user ranking function, as supported by the experimental evaluation in Section 6.

5. Running Example

To demonstrate our approach, we utilize an initial set of patterns $\mathcal{P} = \{P_1, \ldots, P_{10}\}$, along with a user ranking $r_{\triangleright_u}$, and five interestingness measures $\mathcal{M} = \{M_1, \ldots, M_5\}$. The user ranking for $\mathcal{P}$, rankings given by M_i on $\mathcal{P}$, and the `AHPRank` results are presented in the columns of Table 1. It is important to note that none of the measures perfectly match the user ranking (column *user*).

Table 1. Running example with 10 patterns and 5 measures.

User	S	M_1–*Rank*		M_2–*Rank*		M_3–*Rank*		M_4–*Rank*		M_5–*Rank*		g_w–*Rank*	
1	P_7	0.95	**1**	0.48	8	0.79	2	0.30	9	0.80	**1**	0.72	**1**
2	P_3	0.75	3	0.72	1	0.78	3	0.70	**2**	0.61	**2**	0.68	**2**
3	P_6	0.80	2	0.49	7	0.50	9	0.65	4	0.60	**3**	0.61	**3**
4	P_1	0.47	9	0.47	9	0.76	5	0.56	6	0.59	**4**	0.54	**4**
5	P_8	0.56	6	0.65	4	0.63	8	0.69	3	0.40	**5**	0.53	**5**
6	P_{10}	0.57	5	0.50	**6**	0.80	1	0.4	8	0.02	10	0.34	8
7	P_5	0.62	4	0.62	5	0.66	6	0.57	5	0.27	**7**	0.48	**7**
8	P_2	0.48	**8**	0.66	3	0.65	7	0.1	10	0.05	9	0.33	9
9	P_4	0.50	7	0.68	2	0.77	4	0.50	7	0.35	6	0.50	6
10	P_9	0.02	**10**	0.1	**10**	0.05	**10**	0.8	1	0.25	8	0.18	**10**

The data presented in Table 1 is utilized by our algorithm to construct the AHP matrix and learn the weight vector w over the set of interestingness measures $\mathcal{M} = \{M_1, \ldots, M_5\}$. For simplicity, we refer to the AHP algorithm in the passive mode with $\mathcal{S} = \{S_a = \langle P_3, P_1, P_5, P_2, P_4 \rangle\}$. In this case, the user's ranking order is $P_3 \triangleright_u P_1 \triangleright_u P_5 \triangleright_u P_2 \triangleright_u P_4$. AHP learns the weights by invoking `learnWeights` on S_a and computing the correlation between the user rankings of S_a and the rankings of the measures M_i. The matrix in (8) shows the Δ values for each measure pair. It is worth noting that no two measures perfectly match the user ranking (column *user*) for the pattern set $\mathcal{P}$.

To compute $\Delta_{1,2}(S_a)$, we need to calculate Kendall's W $K_1(S_a)$ (resp., $K_2(S_a)$) between the ranking of M_1 (resp., M_2) and the user ranking S_a. The value of $\Delta_{1,2}(S_a)$ is given by $(K_1(S_a) - K_2(S_a))$, which in this case is $(0.25 - 0.25) = 0$. After scaling the values of Δ to $(-9 \cdots -1)$ for negatives and $(1 \ldots 9)$ for positives, we can observe from Equation (8) that measure M_5 is better than M_4 with a degree of 6. The same degree of preference is observed between M_5 and M_3, while M_3 and M_4 are indifferent.

$$\Delta = \begin{array}{c} \\ (M1) \\ (M2) \\ (M3) \\ (M4) \\ (M5) \end{array} \begin{array}{c} \begin{array}{ccccc} (M1) & (M2) & (M3) & (M4) & (M5) \end{array} \\ \begin{pmatrix} & 0 & 0.42 & 0.42 & -0.43 \\ & & 0.20 & 0.28 & -0.25 \\ & & & -0.08 & -0.55 \\ & & & & -0.60 \\ & & & & \end{pmatrix} \end{array} \xrightarrow{Scaling} \begin{array}{c} \\ (M1) \\ (M2) \\ (M3) \\ (M4) \\ (M5) \end{array} \begin{array}{c} \begin{array}{ccccc} (M1) & (M2) & (M3) & (M4) & (M5) \end{array} \\ \begin{pmatrix} & 1 & 4 & 4 & -4 \\ & & 2 & 3 & -3 \\ & & & 1 & -6 \\ & & & & -6 \\ & & & & \end{pmatrix} \end{array} \tag{8}$$

Afterwards, the `learnWeights` function in `AHPRank` computes the AHP matrix A using the scaled Δ matrix (i.e., the average correlation gap) as input:

$$\mathbf{A} = \begin{array}{c} \\ (M1) \\ (M2) \\ (M3) \\ (M4) \\ (M5) \end{array} \begin{array}{c} \begin{array}{ccccc} (M1) & (M2) & (M3) & (M4) & (M5) \end{array} \\ \begin{pmatrix} 1 & 1 & 4 & 4 & 1/4 \\ 1 & 1 & 2 & 3 & 1/3 \\ 1/4 & 1/2 & 1 & 1 & 1/6 \\ 1/4 & 1/3 & 1 & 1 & 1/6 \\ 4 & 3 & 6 & 6 & 1 \end{pmatrix} \end{array} \tag{9}$$

At the end, `AHPRank` computes the weighting vector w by solving a minimization problem. The learned weight vector $w = (w_{M_1}, w_{M_2}, w_{M_3}, w_{M_4}, w_{M_5})$ is $(0.24, 0.24, 0.065, 0.065, 0.39)$, reflecting the importance of each measure for achieving the goal of the user ranking function g_w.

Using the AHP interestingness measure g_w from Equation (6), we can rank all patterns in $\mathcal{P}$ provided in Table 1. `AHPRank` achieves an overall ranking accuracy of 91% in this example, correctly ranking the first five patterns, the seventh, and the tenth ones.

6. Experiments

In this section, we empirically evaluate the effectiveness of our proposed pattern ranking framework `AHPRank`. We first introduce our case study on association rules mining and the different oracles we used to simulate user-specific rankings. Next, we present the research questions we aimed to answer, the experimental protocol we followed, and the results we obtained.

6.1. Mining Associations Rules (ARs)

Our approach, `AHPRank`, is experimentally evaluated on association rule mining, which is one of the most important and well-studied tasks in data mining. Association rules are implications of the form $X \rightarrow Y$, where X and Y are itemsets such that $X \cap Y = \emptyset$ and $Y \neq \emptyset$. Here, X represents the body of the rule and Y represents its head. The frequency of an itemset X in a dataset, denoted by $freq(X)$, is the number of transactions of the dataset containing X. The frequency of a rule $X \rightarrow Y$ is the frequency of the itemset $X \cup Y$, that is, $freq(X \rightarrow Y) = freq(X \cup Y)$.

Various interestingness measures for association rules have been proposed, including support, confidence, interest factor, correlation, and entropy. Tan et al. [38] conducted a study on the usefulness of existing measures in different application types and identified seven independent groups of consistent measures having similar properties, as shown in Table 2.

Table 2. Independent subjective measure groups.

Groups	Measures
1	**Yules Q**, Yules Y, Odds Ratio
2	**Cosine**, Jaccard
3	**Laplace**, Support
4	**ϕ coefficient**, Collective Strength, Piatetsky-Shapiro's
5	**Goodman–Kruskal's**, Gini Index
6	**Interest factor**, added value, Klosgen K
7	**Certainty factor**, Mutual Information, Cohen's κ

To evaluate our approach, we selected one measure from each of the seven groups of consistent measures identified by Tan et al. [38]. This gave us a set of seven measures that are independent and have similar properties. These measures are highlighted in bold in Table 2.

6.2. User Feedback Emulators

Since it can be challenging to evaluate an interactive approach with limited user feedback, we simulated user feedback by using three different objective target ranking functions:

- RAND-EMU: The user-specific ranking is equivalent to a random weighted aggregation function. For each measure M_i, we generate a random weight $w_i \in [0, 1]$ such that $\sum_{i=1}^{m} w_i = 1$.
- LEX-EMU: The user-specific ranking follows a lexicographic order on the measures. We define a lexicographic order lex on $\mathcal{M}$ as $lex(\mathcal{M}) = \langle l_1, \ldots, l_m \rangle$ such that $\bigcup_{i=1}^{m} l_i = \mathcal{M}$. Given two patterns (P_1, P_2), P_1 is preferred to P_2 if $(l_i(P_1) > l_i(P_2))$ or $(l_i(P_1) = l_i(P_2) \wedge l_{i+1}(P_1) > l_{i+1}(P_2))$ for $i = 1$ to $m - 1$.
- CHI-EMU: χ^2 is a statistical measure that is a good candidate to emulate user feedback [39], while χ^2 can be a complex function to approximate with non-trivial correla-

tions, it is also a quality measure suggested in [5]. We use χ^2 as the target user-specific ranking function over ARs. For an association rule $X \rightarrow Y$, the χ^2 value is defined as:

$$\chi^2(X \rightarrow Y) = \frac{\left(freq(X \rightarrow Y) - \frac{freq(X)freq(Y)}{N}\right)^2}{\frac{freq(X)freq(Y)}{N}}, \tag{10}$$

where N is the number of transactions in the dataset.

6.3. Research Questions

Our evaluation seeks to address the following research questions:

- **RQ1**: Can we determine a user's preferences for a set of patterns based on a sample of ranked patterns? If so, how much data is needed and how long does the learning process take?
- **RQ2**: How does our proposed AHP-based learning method compare to the SVM-based baseline for automatically learning user-specific ranking functions?
- **RQ3**: How effective is `AHPRank` in an active learning context? Is the sensitivity heuristic a good choice for query selection?
- **RQ4**: How effective is `AHPRank` in an interactive data mining context?

6.4. Experimental Protocol

6.4.1. Implementation Settings

We implemented our `AHPRank` approach in Java with two modes: the passive mode, denoted as `AHPRank.0`, and the active mode, denoted as `AHPRank.1`. The code is publicly available on GitHub at `github.com/lirmm/AHPRank`. We compared our approach to the state-of-the-art SVM-based approach `RankingSVM` [5]. We use `RankingSVM.0` to denote the passive version and `RankingSVM.1` to denote the active version, following [3]. All experiments were conducted on an Intel Core *i*7 2.4 GHz with 16 GB of RAM, with a maximum duration of one hour.

Metrics

We use three metrics to evaluate the performance of our approach:

1. The Spearman's rank correlation coefficient ρ measures the accuracy of the learned ranking compared to the target ranking over n patterns. It is computed as follows:

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(rank_L(P_i) - rank_T(P_i))^2}{n(n^2-1)} \tag{11}$$

2. The recall metric R@k evaluates the ability of our approach to identify the top k most interesting patterns. It is computed as follows:

$$\text{R@}k = \frac{|\{rank_L(P_i) \leq k : i \in 1..n \wedge rank_T(P_i) \leq k\}|}{k} \tag{12}$$

3. CPU time in seconds is measured for both the passive and active modes. The waiting time between two queries is also recorded for the active mode.

6.5. Benchmark Datasets

We selected several datasets of realistic sizes from the FIMI repository (fimi.uantwerpen.be/data/, accessed on 19 March 2023). These datasets have different characteristics that represent various application domains. Table 3 reports the number of transactions ($|\mathcal{T}|$), the number of items ($|\mathcal{I}|$), the application domain, and the number of valid rules (`#Rules`) corresponding to the initial rules mined using a standard association rules algorithm without any knowledge about the user, for each dataset. The datasets are presented in ascending order of `#Rules`.

Table 3. Dataset Characteristics.

Dataset	$\|\mathcal{T}\|$	$\|\mathcal{I}\|$	Density (%)	Type of Data	#Rules
Hepatitis	137	68	50.00	Disease	0.5 M
Connect	67,557	129	33.33	Game steps	1 M
Mushroom	8124	119	18.75	Species of mushrooms	1.5 M
T40	100,000	1000	4.20	Synthetic dataset	2 M
Retail	88,162	16,470	0.06	Retail market basket data	2.5 M

T40:T40I10D100K.

6.6. Passive Learning Results

In this section, we address the first two research questions (**RQ1** and **RQ2**). To that end, we perform a five-fold cross-validation on each dataset, where we randomly select 20% of the rules as training data and use the remaining 80% for testing. This type of cross-validation allows us to evaluate the effectiveness of the approaches in learning from relatively small training sets. We report the results averaged over the five folds.

(A) Analyzing the different user feedback emulators. Table 4 presents the results of the correlation analysis between the user-specific ranking functions (RAND-EMU, LEX-EMU, and CHI-EMU) and the seven interestingness measures, as well as the virtual best measure (VBM), which returns the best rank correlation ρ provided by one of the seven measures. The analysis was performed on all datasets, and the results are reported in terms of Spearman's rank correlation coefficient ρ.

Table 4. Correlation results with user ranking for using the emulators.

Datasets	Measures							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	VBM
RAND-EMU								
Hepatitis	0.78	0.92	0.29	**0.97**	0.79	0.85	0.45	**0.97**
Connect	0.68	0.62	**0.95**	0.70	0.60	0.45	0.57	**0.95**
Mushroom	0.84	**0.97**	0.28	0.83	0.55	0.73	0.36	**0.97**
T40	0.71	**0.99**	0	0.76	**0.99**	**0.99**	0	**0.99**
Retail	0	**0.98**	0.43	0.71	0.84	**0.98**	0.51	**0.98**
LEX-EMU								
Hepatitis	0.79	0.64	0	0.68	**0.92**	0.63	0	**0.92**
Connect	0.21	0.47	**0.76**	0.26	0	0.23	0.50	**0.76**
Mushroom	0.21	**0.82**	0.38	0.77	0.78	0.37	0	**0.82**
T40	0.76	0.98	0	0.77	**0.99**	0.97	0	**0.99**
Retail	0.10	0.78	0.54	0.49	**0.84**	0.80	0.58	**0.84**
CHI-EMU								
Hepatitis	0.20	0.92	0.36	**0.96**	0.73	0.85	0.43	**0.96**
Connect	0.09	0.15	0	0.02	**0.29**	0.01	0	**0.29**
Mushroom	0.51	0.64	0	0.46	0.28	**0.88**	0	**0.88**
T40	0.71	0.98	0.17	0.64	0.98	**0.99**	0.29	**0.99**
Retail	0	**0.94**	0.53	0.60	0.83	0.57	0.61	**0.94**

(1): YulesY, (2): Cosine, (3): Laplace, (4): Leverage, (5): Lambda, (6): InterstFactor, (7): Certainty.

The results show that the correlation between a given measure and a user-specific ranking function can vary significantly depending on the dataset. For instance, the Lambda measure is highly correlated with CHI-EMU on the T40 dataset ($\rho = 98\%$), but it is weakly correlated on the Connect dataset ($\rho = 29\%$). Similarly, the correlation between RAND-EMU or LEX-EMU and the measures also varies significantly across the datasets. However, the VBM approach, which selects the best measure for each dataset, achieves a high level of accuracy (with a mean of 91%). These results suggest that a weighted aggregation of the selected measures could lead to a good trade-off between accuracy and robustness.

(B) Comparing `AHPRank.0` with `RankingSVM.0`. Table 5 presents the results of the k-folds cross-validation for `RankingSVM.0` and `AHPRank.0`, where we report the averaged rank correlation ρ, recall values (`R@10%` and `R@1%`), and CPU time in seconds averaged over the folds. Since `RankingSVM.0` can handle a training set of up to 100 K rules within an hour, we compared it with our approach with the `Hepatitis` dataset only. We observed that `RankingSVM.0` outperformed `AHPRank.0` in terms of ranking accuracy for all user feedback emulators, although `AHPRank.0` remained competitive with acceptable accuracy. The same trend was observed in terms of recall at the 10% and 1% top of the ranking (`R@10%` and `R@1%`). We noted that `AHPRank.0` achieved a high correlation with the user ranking functions on most datasets. However, on `Connect`, we observed a weak correlation with CHI-EMU, while a high correlation was observed with RAND-EMU and LEX-EMU. This can be attributed to the fact that RAND-EMU and LEX-EMU are linear functions expressed with the given seven interestingness measures, which explains their high accuracy, while CHI-EMU is a complex function that requires more extensive statistics to learn. Notably, the high accuracy of `RankingSVM.0` came at the expense of longer running times. For instance, it took more than 15 min for `RankingSVM.0` to learn from a training set of 100K rules, and exceeding 100K rules, `RankingSVM.0` required more than one hour, while `AHPRank.0` was able to handle 7.5 M rules in less than 4 min.

In what follows, the observations and the conclusions drawn from CHI-EMU remain true for RAND-EMU and LEX-EMU. For the sake of simplicity, we only report the results on the complex function CHI-EMU.

Table 5. Five-fold cross-validation results (passive learning).

RAND-EMU								
	`RankingSVM.0`				`AHPRank.0`			
	ρ	`R@10%`	`R@1%`	t(s)	ρ	`R@10%`	`R@1%`	t(s)
`Hepatitis`	0.94	0.74	0.77	851	0.93	0.79	0.71	9
`Connect`	-	-	-	`TO`	0.95	0.72	0.63	26
`Mushroom`	-	-	-	`TO`	0.89	0.69	0.66	32
`T40`	-	-	-	`TO`	0.99	0.93	0.58	63
`Retail`	-	-	-	`TO`	0.93	0.95	0.96	66
mean	-	-	-	-	0.94	0.82	0.71	39
LEX-EMU								
	`RankingSVM.0`				`AHPRank.0`			
	ρ	`R@10%`	`R@1%`	t(s)	ρ	`R@10%`	`R@1%`	t(s)
`Hepatitis`	0.99	0.99	0.99	902	0.92	0.86	0.81	13
`Connect`	-	-	-	`TO`	0.60	0.46	0.50	21
`Mushroom`	-	-	-	`TO`	0.86	0.56	0.65	32
`T40`	-	-	-	`TO`	0.99	0.93	0.46	71
`Retail`	-	-	-	`TO`	0.78	0.68	0.65	68
mean	-	-	-	-	0.83	0.70	0.61	41
CHI-EMU								
	`RankingSVM.0`				`AHPRank.0`			
	ρ	`R@10%`	`R@1%`	t(s)	ρ	`R@10%`	`R@1%`	t(s)
`Hepatitis`	0.99	0.97	0.92	979	0.94	0.90	0.77	10
`Connect`	-	-	-	`TO`	0.15	0.28	0.67	17
`Mushroom`	-	-	-	`TO`	0.98	0.91	0.93	28
`T40`	-	-	-	`TO`	0.99	0.99	0.96	84
`Retail`	-	-	-	`TO`	0.91	0.96	0.95	50
mean	-	-	-	-	0.79	0.81	0.86	38

(C) Impact of varying the size of the training data on learning. It is crucial to emphasize that our proposed AHPRank approach aims to achieve a fast learning process while providing high accuracy in ranking patterns. To further support our observation on the scalability of AHPRank.0 compared to RankingSVM.0, we present in Figures 2 and 3 a performance comparison by varying the size of the training data when learning the CHI-EMU function, as similar results were obtained for RAND-EMU and LEX-EMU functions. For each dataset, we randomly select *nb* rules, and we compare the two approaches for *nb* $\in$ 10, 100, 1K, 10K, 100K. The results are averaged over ten runs.

Regarding the Spearman correlation ρ, Figure 2 shows that the two approaches have a discrepancy of only 5% when the training data does not exceed 1K. However, the gap becomes significantly more substantial (exceeding 10%) and in favor of AHPRank.0 when the training data contains 10K and 100K rules.

In terms of recall at 10% and 1%, RankingSVM.0 outperforms AHPRank.0 with a gap of 23% and 15% at R@10% and R@1%, respectively, when the training data contains only 10 rules. However, as the size of the training data increases, the gap between the two approaches becomes narrower, with a difference of less than 5% for R@10%, and remaining relatively constant for R@1%.

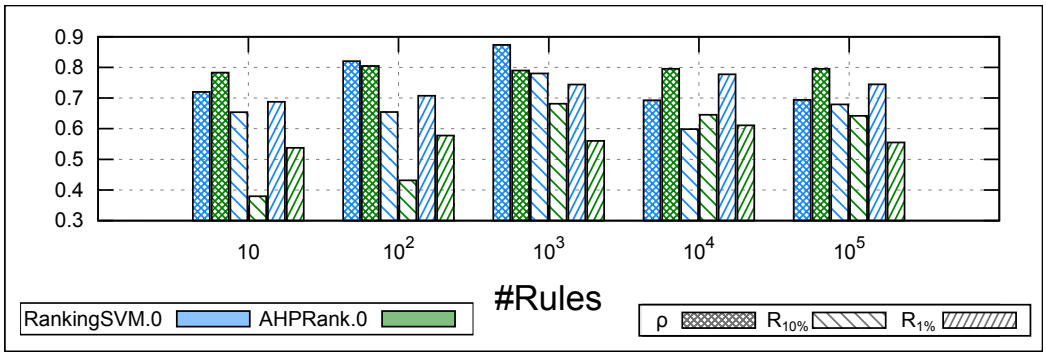

Figure 2. Learning accuracy comparison between RankingSVM.0 and AHPRank.0 learning CHI-EMU on different training data sizes.

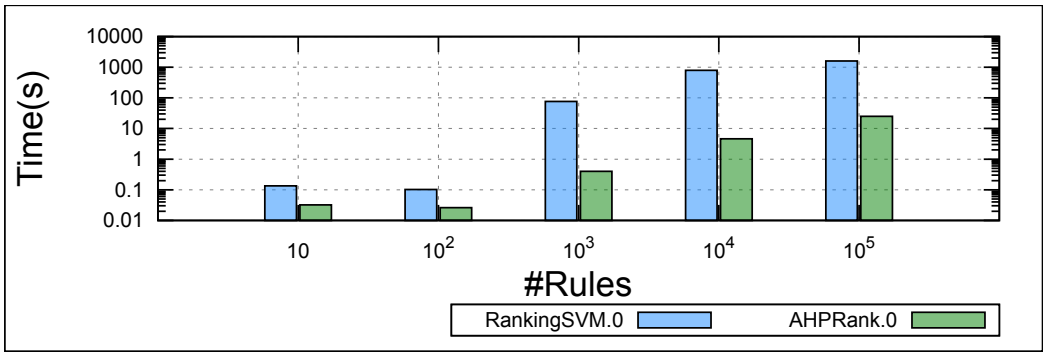

Figure 3. CPU time comparison between RankingSVM.0 and AHPRank.0 learning using the CHI-EMU emulator on different training data sizes averaged over all the datasets.

In terms of CPU time, Figure 3 demonstrates that RankingSVM.0 can process training data of up to 100 rules in less than a second, whereas it takes more than one minute for 1K rules, 13 min for 10K rules, and over 26 min for 100K rules. In contrast, AHPRank.0 is capable of handling training data ranging from 100 to 100K rules in a time span ranging from 0.03 to 24.86 s, while still maintaining a ranking accuracy comparable to that of RankingSVM.

6.7. Active Learning Results

In this section, we aim to address research questions **RQ2** and **RQ3**, which involve comparing the active learning versions of RankingSVM and AHPRank. To achieve this, we

employ a straightforward process that involves asking the user ranking queries on pairs of patterns (e.g., do you prefer P_i to P_j?).

(A) Evaluating the effectiveness of our sensitivity-based heuristic. In this section, we aim to evaluate the effectiveness of our sensitivity-based generator (SBG) (see Section 4.5) in improving the active learning process of `AHPRank.1`. To do so, we compare SBG to a random generator (RG), where RG randomly selects a pair of patterns from `#Rules` and submits them to both `RankingSVM.1` and `AHPRank.1`. We conduct our experiments with human-in-the-loop, setting the number of queries (i.e., iterations of Algorithm 2) to a maximum of 20 queries. We repeat the experiment 10 times and take the average result to account for the sampling step of our SBG and the randomness of the RG approach. After a few tests, we set the sample size $\mathcal{X}$ picked in line 10 of Algorithm 2 to $\theta = 10^3$, providing a good trade-off between time selection and the accuracy of the selected pair.

Figure 4 presents a scatter plot of 20 iterations comparing the performance of `RankingSVM.1` and `AHPRank.1` with both RG and SBG. We report the Spearman correlation ρ, recall at 10%, and recall at 1% for each iteration. The results show that the use of SBG outperforms RG, regardless of the learning algorithm, as can be observed from the scatter plot.

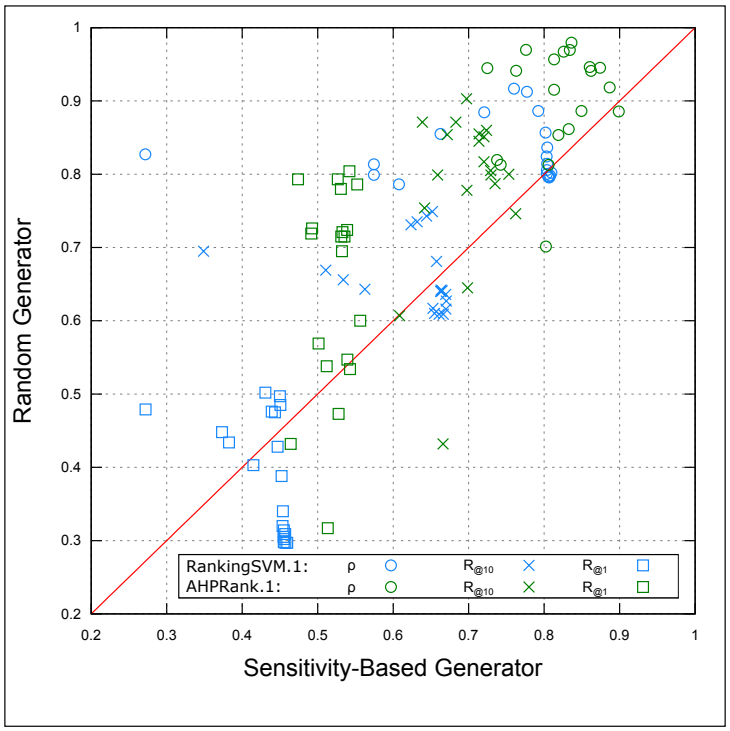

Figure 4. Qualitative comparison of the random vs. sensitivity-based query generator using the Spearman correlation ρ and the recall at 10% and 1% of each iteration.

To support our observation on the effectiveness of SBG, we conducted a statistical test using the Wilcoxon signed-rank test. We used a one-tailed alternative hypothesis with the null hypothesis that RG is more efficient than SBG: H0: The accuracy using RG $\geq$ the accuracy using SBG. The alternative hypothesis H1 states that SBG outperforms RG. With this statistical test, we concluded that the use of SBG is more efficient than RG (i.e., H1 is accepted).

Table 6 reports the p value, z-score, and confidence interval (CI) of each test. Except for the case of (`RankingSVM.1`, `R@10%`), we had strong evidence to reject the null hypothesis based on the p value column. The CI column clearly shows that the use of SBG is better than RG. Consequently, we will use the SBG heuristic in `RankingSVM.1` and `AHPRank.1` in the following.

Table 6. Wilcoxon signed-rank test (RG vs. SBG).

Approaches	Metrics	*p* Value	*z*-Score	CI
RankingSVM.1	ρ	0.00226	−2.8373	99%
	R@10%	0.12302	−1.1573	70%
	R@1%	0.06301	−1.5306	90%
AHPRank.1	ρ	0.00034	−3.3973	99.9%
	R@10%	0.00205	−2.8746	99%
	R@1%	0.00139	−2.9866	99%

(B) Comparing AHPRank.1 with RankingSVM.1. Table 7 presents a comparison between RankingSVM.1 and AHPRank.1 on five datasets, reporting the Spearman correlation ρ, recall at 10% and 1% stopping criterion T at 10, 50, and 100 queries, and the average latency time Time between two queries for RankingSVM.1 (never exceeding the latency bound of 0.1 s). The findings hold for the RAND-EMU, LEX-EMU, and CHI-EMU functions.

Table 7. Qualitative evaluation of RankingSVM.1 vs. AHPRank.1.

		(a)	(b)	(c)	(d)	(e)
10 Queries						
(1)	ρ	0.73	0.04	0.69	0.99	0.97
	R@10%	0.61	0.28	0.38	0.92	0.91
	R@1%	0.44	0.49	0.36	0.45	0.67
	Time	0	0	0	0	0
(2)	ρ	**0.78**	**0.10**	**0.93**	0.99	**0.99**
	R@10%	**0.66**	**0.36**	**0.80**	0.97	0.91
	R@1%	**0.51**	**0.50**	**0.71**	**0.81**	**0.93**
50 Queries						
(1)	ρ	0.94	**0.20**	0.71	0.99	0.98
	R@10%	0.77	0.10	0.39	0.92	0.91
	R@1%	0.57	0	0.36	0.45	0.67
	Time	**0**	**0**	**0**	2	3
(2)	ρ	**0.95**	0.10	**0.93**	0.99	**0.99**
	R@10%	**0.83**	**0.40**	**0.72**	**0.94**	0.91
	R@1%	**0.66**	**0.45**	0.51	**0.64**	**0.93**
100 Queries						
(1)	ρ	0.94	**0.29**	0.72	0.99	0.98
	R@10%	0.75	0.20	0.41	0.92	0.91
	R@1%	0.53	0	0.36	0.45	0.67
	Time	0	2	3	3	10
(2)	ρ	**0.98**	0.10	**0.81**	0.99	**0.99**
	R@10%	**0.89**	**0.35**	**0.49**	**0.95**	0.91
	R@1%	**0.78**	**0.42**	**0.37**	**0.74**	**0.93**

(1): RankingSVM.1, (2): AHPRank.1, (a): Hepatitis, (b): Connect, (c): Mushroom, (d): T40, (e): Retail.

The main observation from Table 7 is that AHPRank.1 outperforms RankingSVM.1. Looking at the Hepatitis dataset, for example, AHPRank.1 achieves an accuracy of 78% with only 10 queries, while RankingSVM.1 achieves 73%. At 50 queries, AHPRank.1 achieves an accuracy of 95%, compared to 94% for RankingSVM.1. At 100 queries, AHPRank.1 reaches 98% accuracy, while RankingSVM.1 remains stable at 94%. In terms of recall, AHPRank.1 discovers the most relevant patterns in the first 50K and 5K patterns (out of 500K) with accuracies of 78% and 66% over 10 queries, respectively, compared to 61% and 44% for RankingSVM.1.

For the `Retail` dataset, `AHPRank.1` achieves high accuracy over 10 queries, which remains stable over the following 90 queries. `RankingSVM.1` results are less impressive on `Retail`, particularly on the recall metric. However, the main difference is in the waiting time `Time`, where `RankingSVM.1` can keep the user waiting for more than 10 s between two queries. The same observation can be made on the other datasets, where `RankingSVM.1` can be hampered by overall waiting time, even with queries of size 2. In contrast, `AHPRank.1` shows an instantaneous behavior, taking less than 0.1 s between two queries. This represents a limitation in the use of `RankingSVM.1`, especially when the learning is integrated into an interactive data mining process, where a reasonable latency time for a human user is around a few seconds [40].

6.8. Interactive Learning Results

In this section, we address the last research question **RQ4**, and conduct two experiments to evaluate the robustness and performance of our approach in the presence of human mistakes.

Our first experiment aims to evaluate the robustness of our approach in the face of human mistakes. To simulate situations where the user feedback may be incorrect, we randomly select a set of queries and swap the user preference with a probability `Err`. Specifically, a user who prefers pattern P_i over P_j will mistakenly prefer P_j over P_i with a probability `Err`.

Figure 5 compares the performance of `RankingSVM.1` and `AHPRank.1` under different levels of human error, ranging from 0% to 40%, and reports results averaged over 10 runs on the whole set of datasets using CHI-EMU and submitting 20 queries to the user. Our results show that `AHPRank.1` is quite stable and robust even with high levels of human error, up to 40% (8 mistakes out of 20). The overall correlation between the learned function and the user (ρ) remains stable at 77% without mistakes and drops only to 70% under `Err` $=$ 40%. However, `RankingSVM.1` is stable only up to `Err` $=$ 20%, and then its accuracy drops significantly, with ρ decreasing from 71% to 54%. In terms of recall, the decline is less than 2% at `R@10%` and 8% at `R@1%` under `AHPRank.1`, while `RankingSVM.1` experiences a decline exceeding 10% at both `R@10%` and `R@1%`. In terms of CPU time, we observe waiting times ranging from 6 to 12 s under `RankingSVM.1`, whereas our approach never exceeds 0.02 s between two queries.

These results suggest that our approach is robust and can handle user mistakes, while `RankingSVM.1` is less robust and can suffer from significant drops in accuracy when faced with human mistakes. Additionally, our approach provides faster query processing times, which is critical in interactive data mining scenarios where human users expect near-instantaneous feedback.

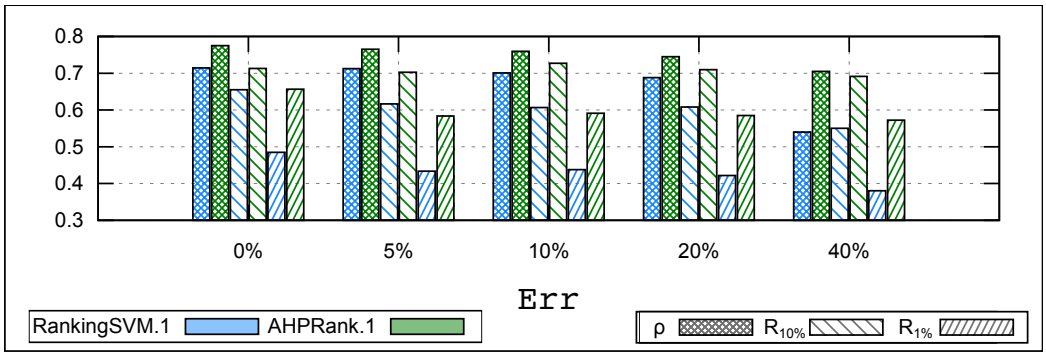

Figure 5. Learning under mistakes (20 queries).

Our second experiment aims to evaluate the robustness of our approach when faced with an undecided user who begins with a set of preferences A but ends up with preferences B. Here, the initial user preferences A are biased towards the presented patterns until the user is more comfortable with preferences B.

To simulate such situations where the user's target function changes during the learning process, we conduct an experiment where we start with LEX-EMU as the target function to learn, and after x queries, we switch to RAND-EMU. This learning process takes $(20 - x)$ more queries, and we refer to it as the LEX2RAND target. We choose to switch from LEX-EMU to RAND-EMU because the two functions are linear. We compare `RankingSVM.1` and `AHPRank.1` on the LEX2RAND(x) target for a total of 20 queries, where $x \in 0, 5, 10$. Note that with $x = 0$, LEX2RAND(0) is equivalent to RAND-EMU.

Figure 6 shows the averaged results of 10 runs on all datasets over 20 queries. The main observation that we can draw is that `AHPRank.1` is stable even when faced with a changing linear function to learn, even if the user's preferences change halfway through the learning process (i.e., after 10 queries). However, changing the target function during the learning process can significantly impact the accuracy of `RankingSVM.1` (a decline of 44% in terms of ρ). Regarding CPU time, our approach is 50 times faster than `RankingSVM.1`.

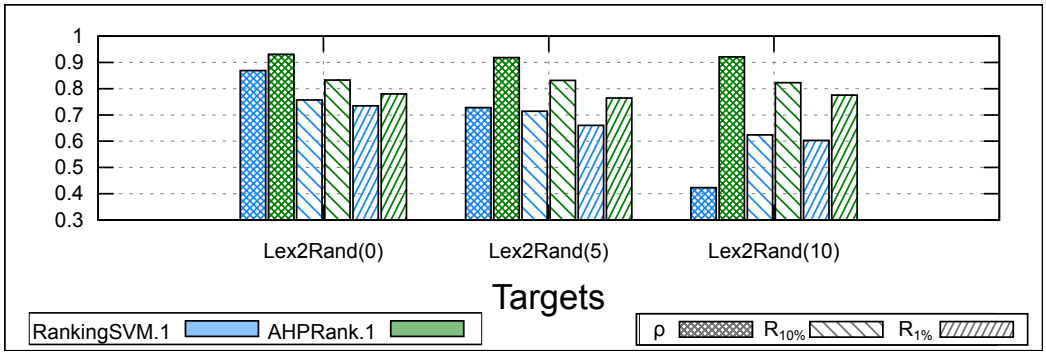

Figure 6. Learning under swaps (20 queries).

7. Discussion

In this section, we review the key experimental findings from the proposed `AHPRank` approach, as well as its limitations and potential for tackling real-world applications. Additionally, we suggest directions for future research.

7.1. Results Analysis: Pros and Cons of the Proposed Approach

According to our experimental study given in Section 6, the results highlight that `AHPRank` has superior scalability compared to the state-of-the-art `RankingSVM` in passive learning mode, particularly when the number of data points is high. Although `RankingSVM` has a high level of accuracy, it requires more training time, which makes it impractical for an interactive learning context. In active mode, our sensitivity-based heuristic was shown to be effective in selecting informative data points, and `AHPRank` was shown to be more robust than `RankingSVM` with respect to incorrect rankings.

In summary, our research findings indicate that `AHPRank` offers an interesting trade-off between ranking accuracy and running time, making it a practical and efficient solution for various real-world applications. Its time complexity scales linearly with the training data size, and it can achieve high ranking accuracy in just a few seconds. Overall, our results suggest that `AHPRank` is a promising approach for ranking tasks.

As with any ranking methodology, `AHPRank` has certain limitations that should be taken into consideration for further improvements:

1. **Expert judgment bias:** `AHPRank` relies heavily on subjective expert judgment, which can introduce bias into the ranking process.
2. **Limited applicability:** `AHPRank` may not be applicable in all scenarios, such as cases where there are no clear criteria or when the criteria are too subjective.
3. **Data availability:** The accuracy of `AHPRank` depends on the availability and quality of data. If the data are incomplete or inaccurate, they may affect the ranking results.
4. **Complexity:** The AHP methodology can be complex to implement and interpret, which may make it difficult to use by non-experts.

5. **Sensitivity to input parameters:** `AHPRank` requires input parameters, such as criteria, which can be sensitive to changes and may require careful tuning for optimal performance.

7.2. Real-World Applications

The use of `AHPRank` algorithms can provide significant benefits for companies, improving the efficiency of decision-making and recommendation processes. Here are some potential industrial applications of `AHPRank`:

- **Cooperative, Connected and Automated Mobility:** One potential application of `AHPRank` in the context of autonomous vehicles is to rank the importance of different sensor inputs for decision-making. For example, cameras, lidars, and radars are commonly used in autonomous vehicles to perceive the surrounding environment. However, some sensors may be more reliable or informative than others in certain scenarios. By using `AHPRank`, the importance of each sensor input can be determined based on the preferences of the vehicle user or the specific driving scenario. Another potential use of `AHPRank` in the context of autonomous vehicles is to rank different driving strategies or maneuvers based on safety and efficiency. For instance, `AHPRank` can be used to determine the optimal speed and following distance when driving in heavy traffic, or to prioritize which safety features to activate in emergency situations. Furthermore, `AHPRank` can also be used to prioritize maintenance and repair tasks for autonomous vehicles. By considering factors such as cost, safety, and reliability, `AHPRank` can help identify the most critical components that require immediate attention.
- **ChatGPT:** The integration of `AHPRank` with ChatGPT can have several benefits. Firstly, it can improve the understanding of user intent by ranking the relevance of different topics and keywords in their queries. This, in turn, can help ChatGPT generate more accurate and relevant responses. Additionally, `AHPRank` can be used to rank the generated responses based on the user's preferences, leading to a more personalized and satisfying experience for the user. Furthermore, the use of `AHPRank` in ChatGPT can also assist in selecting the most appropriate response from a set of possible responses. This can be performed by ranking the responses based on various factors such as clarity, accuracy, and relevance to the user's query. By using `AHPRank` to rank these responses, ChatGPT can provide the most suitable response to the user. Overall, the combination of ChatGPT and `AHPRank` can lead to a more efficient and effective conversational AI system that better understands the user's intent and provides personalized and relevant responses. This can ultimately enhance the user's experience and satisfaction with the AI system.
- **Search engines:** `AHPRank` can also be used to improve the efficiency and accuracy of search engines in real-time. By incorporating a more comprehensive and dynamic approach to measuring the relevance and importance of web pages, `AHPRank` has the potential to enhance search engine ranking algorithms. This can lead to a better overall search experience for users, both internally and externally, by providing more relevant and informative results.
- **E-commerce:** Online retailers can benefit from using `AHPRank` to improve their product recommendation systems and search results. By incorporating a more comprehensive and dynamic approach to measuring the relevance and importance of products, `AHPRank` can help increase the accuracy and effectiveness of these systems. This, in turn, can lead to higher conversion rates, increased sales, and enhanced customer satisfaction. Moreover, recent studies in customer relationship management [41] have highlighted the importance of incorporating customer information to enable more effective interactions with clients. `AHPRank` can be used in combination with customer data to support decision-making and planning of coordinated entrepreneurial marketing strategies aimed at attracting and retaining profitable customers. For instance, by using `AHPRank` to rank products based on customer preferences and feedback, re-

tailers can tailor their offerings to better meet the needs and interests of their target customers.

- **Content recommendation:** Content recommendation is a crucial aspect of media companies and content providers as it has a direct impact on user engagement and satisfaction. By using `AHPRank`, these companies can provide personalized content recommendations to their users based on their interests, preferences, and behaviors. This approach will not only enhance the user experience but also increase the likelihood of retaining users and increasing engagement. One of the key advantages of using `AHPRank` for content recommendation is its ability to incorporate multiple criteria in the decision-making process, such as content type, user preferences, and viewing history. This comprehensive approach allows for more accurate and relevant recommendations, ultimately leading to improved user satisfaction. Moreover, content recommendation systems can also be enhanced by incorporating real-time feedback from users, such as user ratings and reviews, to continually improve the recommendation process. This can be achieved by incorporating the feedback into the decision-making process of the `AHPRank` algorithm, allowing for more personalized and accurate recommendations.
- **Supply chain optimization:** A crucial aspect for companies to reduce costs and increase efficiency. By using `AHPRank`, companies can rank suppliers, products, or logistical options based on multiple criteria, such as cost, delivery time, reliability, and quality. This approach enables companies to make informed decisions about suppliers and products, thereby reducing risks associated with supply chain management. In addition, `AHPRank` can also be used to optimize supply chain logistics by evaluating various transportation options, such as different routes or modes of transportation, and selecting the most efficient and cost-effective option. This can ultimately result in faster delivery times, reduced transportation costs, and improved customer satisfaction.

In a more concrete way, here are five real-life tasks where `AHPRank` can prove to be interesting to use:

1. **Supplier evaluation and selection:** `AHPRank` can be used to rank and select suppliers based on multiple criteria, such as quality, cost, and delivery time.
2. **Investment portfolio management:** `AHPRank` can help portfolio managers to rank and select assets based on various factors such as return, risk, and liquidity.
3. **Employee performance evaluation:** `AHPRank` can be used to evaluate employee performance based on multiple criteria such as productivity, teamwork, and innovation.
4. **Product design and development:** `AHPRank` can be used to rank different design options based on factors such as customer preferences, cost, and manufacturability.
5. **Marketing campaign optimization:** `AHPRank` can be used to rank different marketing strategies based on criteria such as target audience, reach, and cost-effectiveness.

Overall, the flexibility and scalability of the `AHPRank` approach make it a valuable tool for decision making in many industries.

7.3. Future Research Directions

There are several potential areas for future research on `AHPRank`. First, one could explore the use of more sophisticated machine learning techniques to further improve the ranking accuracy of `AHPRank`. For instance, deep learning models could be trained to automatically classify the `AHPRank` queries. This combination enables the replacement of the end-user with a module that can learn preferences from user data and respond to queries on their behalf. This can lead to more efficient and personalized user experiences. Second, there is a need to investigate the generalizability of `AHPRank` to different types of data and ranking tasks. Future work could focus on applying `AHPRank` to diverse domains such as healthcare, finance, and social media, and evaluating its effectiveness in each of these contexts. Third, there is an opportunity to extend `AHPRank` to handle dynamic data, where the underlying ranking criteria may change over time. This would require developing

new algorithms and techniques that can adapt to changes in the data and update the rankings in real-time. Finally, another potential area for future research is to explore the use of `AHPRank` in conjunction with other decision-making tools and techniques, such as optimization algorithms and simulation models. This could lead to the development of more comprehensive and integrated decision-making frameworks that can support complex business processes and strategic planning.

8. Conclusions

In this paper, we proposed a novel framework called `AHPRank` for learning pattern ranking functions using the analytic hierarchy process (AHP) multi-criteria decision-making method. Our algorithm can operate in both passive and active learning modes, allowing users to rank subsets of data points according to their preferences. We showed that the learned weights can be used to aggregate all measures into a single ranking function. The latter was demonstrated to closely match the user's ranking preferences through experiments and statistical analysis.

We applied our framework to the association rules mining case study and compared it with state-of-the-art learning methods. Our experimental results showed that `AHPRank` can efficiently learn the ranking function and outperform existing approaches in terms of ranking accuracy and running time. Furthermore, we showed that `AHPRank` can help users effectively prioritize and analyze patterns, leading to better decision making.

Our proposed framework has a wide range of potential applications across various domains, including marketing in e-commerce, finance, healthcare, and social network analysis. In e-commerce, for example, `AHPRank` can be used to recommend products to customers based on their interests, leading to more personalized and effective marketing strategies. In finance, `AHPRank` can assist investors in making informed decisions about where to allocate their funds by ranking investments based on their risk and return profiles. In healthcare, `AHPRank` can predict the most effective treatment for a given patient based on their individual characteristics, improving patient outcomes and reducing healthcare costs. In social network analysis, `AHPRank` can be used to rank posts based on their relevance or importance, helping users to more easily navigate and engage with social media platforms. By leveraging the `AHPRank` algorithm, users can benefit from more accurate and timely ranking patterns, ultimately enhancing their ability to make informed decisions based on the available data.

Future work could include exploring the use of `AHPRank` in combination with other machine learning techniques or developing an interactive system that can learn and update user preferences over time. In summary, our proposed framework offers a practical and efficient solution for learning pattern ranking functions, which can lead to better decision making in various applications.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, data curation, writing and editin: All authors. All authors have read and agreed to the published version of the manuscript.

Funding: This work is carried out as part of the AI4CCAM Horizon Europe project, funded by the European Union under Grant Agreement 101076911.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Silberschatz, A.; Tuzhilin, A. On Subjective Measures of Interestingness in Knowledge Discovery. In Proceedings of the KDD, Montreal, QC, Canada, 20–21 August 1995; pp. 275–281.
2. Bie, T.D. An information theoretic framework for data mining. In Proceedings of the KDD, San Diego, CA, USA, 21–24 August 2011; ACM: New York, NY, USA, 2011; pp. 564–572.
3. Dzyuba, V.; van Leeuwen, M. Interactive discovery of interesting subgroup sets. In Proceedings of the International Symposium on Intelligent Data Analysis, London, UK, 17–19 October 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 150–161.

4. Boley, M.; Mampaey, M.; Kang, B.; Tokmakov, P.; Wrobel, S. One click mining: Interactive local pattern discovery through implicit preference and performance learning. In Proceedings of the IDEA@KDD, Chicago, IL, USA, 11 August 2013; ACM: New York, NY, USA, 2013; pp. 27–35.
5. Dzyuba, V.; van Leeuwen, M. Learning What Matters-Sampling Interesting Patterns. In Proceedings of the PAKDD, Jeju, Republic of Korea, 23–26 May 2017; Volume 10234, pp. 534–546.
6. Xin, D.; Shen, X.; Mei, Q.; Han, J. Discovering interesting patterns through user's interactive feedback. In Proceedings of the KDD, Philadelphia, PA, USA, 20–23 August 2006; ACM: New York, NY, USA, 2006; pp. 773–778.
7. Bao, X.; Gu, T.; Chang, L.; Xu, Z.; Li, L. Knowledge-based interactive postmining of user-preferred co-location patterns using ontologies. *IEEE Trans. Cybern.* **2021**, *52*, 9467–9480. [CrossRef] [PubMed]
8. Chang, L.; Zhang, Y.; Bao, X.; Gu, T. IDMBS: An Interactive System to Find Interesting Co-location Patterns Using SVM. In Proceedings of the Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, 11–14 April 2022; Proceedings, Part III; Springer: Berlin/Heidelberg, Germany, 2022; pp. 518–521.
9. Shalev-Shwartz, S.; Tewari, A. Stochastic Methods for l_1-regularized Loss Minimization. *J. Mach. Learn. Res.* **2011**, *12*, 1865–1892.
10. Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; Hullender, G. Learning to rank using gradient descent. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 89–96.
11. Qin, T.; Liu, T.Y.; Li, H. A general approximation framework for direct optimization of information retrieval measures. *Inf. Retr.* **2010**, *13*, 375–397. [CrossRef]
12. Saaty, T.L. What is the analytic hierarchy process? In *Mathematical Models for Decision Support*; Springer: Berlin/Heidelberg, Germany, 1988; pp. 109–121.
13. Belmecheri, N.; Aribi, N.; Lazaar, N.; Lebbah, Y.; Loudni, S. Une méthode d'apprentissage par optimisation multicritère pour le rangement de motifs en fouille de données. In Proceedings of the Extraction et Gestion des Connaissances, RNTI-E-38, Blois, France, 24–28 January 2022; pp. 289–296.
14. Bhuiyan, M.A.; Al Hasan, M. PRIIME: A generic framework for interactive personalized interesting pattern discovery. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 606–615.
15. Lee, S.J.; Schneijderberg, C.; Kim, Y.; Steinhardt, I. Have academics' citation patterns changed in response to the rise of world university rankings? a test using first-citation speeds. *Sustainability* **2021**, *13*, 9515. [CrossRef]
16. Fleming, T.R.; Harrington, D.P. Nonparametric estimation of the survival distribution in censored data. *Commun. Stat.-Theory Methods* **1984**, *13*, 2469–2486. [CrossRef]
17. Ziakis, C.; Vlachopoulou, M.; Kyrkoudis, T.; Karagkiozidou, M. Important factors for improving Google search rank. *Future Internet* **2019**, *11*, 32. [CrossRef]
18. Zimmer, F.; Scheibe, K.; Stock, M.; Stock, W.G. Fake news in social media: Bad algorithms or biased users? *J. Inf. Sci. Theory Pract.* **2019**, *7*, 40–53.
19. Bruns, A. *Are Filter Bubbles Real?* John Wiley & Sons: Hoboken, NJ, USA, 2019.
20. Buraga, S.C.; Amariei, D.; Dospinescu, O. An owl-based specification of database management systems. *Comput. Mater. Contin* **2022**, *70*, 5537–5550. [CrossRef]
21. Bottou, L.; Lin, C.J. Support vector machine solvers. *Large Scale Kernel Mach.* **2007**, *3*, 301–320.
22. Raedt, L.D. A Perspective on Inductive Databases. *SIGKDD Explor.* **2002**, *4*, 69–77. [CrossRef]
23. Imielinski, T.; Mannila, H. A Database Perspective on Knowledge Discovery. *Commun. ACM* **1996**, *39*, 58–64. [CrossRef]
24. Geng, L.; Hamilton, H.J. Interestingness measures for data mining: A survey. *ACM Comput. Surv.* **2006**, *38*, 9. [CrossRef]
25. Kuznetsov, S.O.; Makhalova, T. On interestingness measures of formal concepts. *Inf. Sci.* **2018**, *442*, 202–219. [CrossRef]
26. Saaty, T.L.; Vargas, L.G. Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios. *Math. Model.* **1984**, *5*, 309–324. [CrossRef]
27. Takeda, E.; Cogger, K.; Yu, P. Estimating criterion weights using eigenvectors: A comparative study. *Eur. J. Oper. Res.* **1987**, *29*, 360–369. [CrossRef]
28. Gass, S.; Rapcsák, T. Singular value decomposition in AHP. *Eur. J. Oper. Res.* **2004**, *154*, 573–584. . [CrossRef]
29. Blankmeyer, E. Approaches to consistency adjustment. *J. Optim. Theory Appl.* **1987**, *54*, 479–488. [CrossRef]
30. Brunelli, M. *Introduction to the Analytic Hierarchy Process*; Springer: Berlin/Heidelberg, Germany, 2014.
31. Saaty, T.L. A scaling method for priorities in hierarchical structures. *J. Math. Psychol.* **1977**, *15*, 234–281. [CrossRef]
32. Dzyuba, V.; van Leeuwen, M.; Nijssen, S.; Raedt, L.D. Interactive Learning of Pattern Rankings. *Int. J. Artif. Intell. Tools* **2014**, *23*, 1460026. [CrossRef]
33. Kendall, M.G.; Smith, B.B. The Problem of *m* Rankings. *Ann. Math. Stat.* **1939**, *10*, 275–287. [CrossRef]
34. Ailon, N. An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity. *J. Mach. Learn. Res.* **2012**, *13*, 137–164.
35. Erkut, E.; Tarimcilar, M. On Sensitivity Analysis in the Analytic Hierarchy Process. *IMA J. Manag. Math.* **1991**, *3*, 61–83. [CrossRef]
36. Pan, V.Y.; Chen, Z.Q. The Complexity of the Matrix Eigenproblem. In Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, GA, USA, 1–4 May 1999; Vitter, J.S., Larmore, L.L., Leighton, F.T., Eds.; ACM: New York, NY, USA, 1999; pp. 507–516. [CrossRef]
37. Saaty, T.L.; Ozdemir, M.S. Why the magic number seven plus or minus two. *Math. Comput. Model.* **2003**, *38*, 233–244. [CrossRef]

38. Tan, P.N.; Kumar, V.; Srivastava, J. Selecting the right objective measure for association analysis. *Inf. Syst.* **2004**, *29*, 293–313. [CrossRef]
39. Ringuest, J.L. A chi-square statistic for validating simulation-generated responses. *Comput. Oper. Res.* **1986**, *13*, 379–385. [CrossRef]
40. Lallemand, C.; Gronier, G. Enhancing User eXperience During Waiting Time in HCI: Contributions of Cognitive Psychology. In Proceedings of the Designing Interactive Systems Conference DIS '12, Newcastle Upon Tyne, UK, 11–15 June 2012; ACM: New York, NY, USA, 2012; pp. 751–760. [CrossRef]
41. Guerola-Navarro, V.; Gil-Gomez, H.; Oltra-Badenes, R.; Soto-Acosta, P. Customer relationship management and its impact on entrepreneurial marketing: A literature review. *Int. Entrep. Manag. J.* **2022**, 1–41. [CrossRef]