Interactive Ranking of Skylines Using Machine Learning Techniques

Weiwei Cheng, Eyke Hüllermeier, Bernhard Seeger, Ilya Vladimirskiy
Fachbereich Mathematik und Informatik
Philipps-Universität Marburg
{cheng,eyke,seeger,ilya}@mathematik.uni-marburg.de

Abstract

So-called skyline queries have received considerable attention in the field of databases in recent years. Roughly speaking, the skyline of a given set of objects, represented in terms of attributes with preferentially ordered domains, is given by the Pareto-optimal elements of this set. An important problem of Skyline queries is that answer sets can become extremely large. From an application point of view, a system response in terms of a *ranking* of elements, ordered according to the user's preferences, would hence be more desirable than an unordered set. In this paper, we propose a method for constructing such a ranking in an interactive way. The key idea of our approach is to ask for user feedback on intermediate results, and to use this feedback to improve, via the induction of a latent utility function, the current ranking so as to represent the user's preferences in a more faithful way.

1 Introduction

The *skyline* operator and corresponding skyline queries were first introduced by Baorzsaonyi et al. in 2001 [Borzsonyi *et al.*, 2001] and, since then, have attracted considerable attention in the field of databases. A skyline query is a special type of *preference query*: The skyline of a d-dimensional dataset consists of the data objects that are non-dominated in a Pareto sense and, therefore, potentially optimal for a user. Stated differently, each object that is not on the skyline is of no interest, as it is definitely worse than at least one other object in the dataset; more details about skylines will follow in Section 2.1.

Pareto-dominance is an extreme conception of dominance, as it is very demanding and, therefore, does not discriminate well between alternatives. Consequently, a skyline query may produce huge answer sets, a problem that aggravates with the dimensionality of the dataset and impairs the usefulness of skyline queries. One idea to avoid this problem is to exploit the preferences of a particular user. In fact, a specific user will usually not be indifferent between all skyline objects. Instead, the preference relation of this user will be a *refinement* of the Pareto-dominance relation, which is the weakest assumption on preferences one can make and is valid for *every* user. If the user's refined preference relation was known, it could be used to further reduce the set of candidate objects returned by the system or, even better, to order these objects according to their degree of preference.

In this paper, we present a special approach to information retrieval (IR) that combines skyline computation and ranking. The idea is to apply machine learning techniques in order to elicit a specific user's preferences, and to use this knowledge to compute a *ranking* of the skyline objects. This idea is completely in line with research trends in the IR field that are focused on user modeling [Shen *et al.*, 2005] and seek to exploit user feedback [Rochio, 1971; Robertson and Jones, 1976; Salton and Buckley, 1990; Shen and Zhai, 2005] for making retrieval systems interactive, context-aware, and adaptive [Detyniecki *et al.*, 2006].

To realize this idea, some kind of "training data" in the form of preference information is of course needed. In order to minimize the user's effort, we integrate corresponding preference questions into the learning process in a dynamic way. Roughly speaking, instead of separating the training phase from the application, the idea is to immediately exploit every piece of information: Each preference information is used to improve the current ranking of the skyline; in case the user is still not satisfied with the result, new information is requested, and this process is repeated until the user has eventually found what he was searching for. This procedure essentially corresponds to Rochio's well-known relevance feedback loop [Rochio, 1971].

The paper is organized as follows: The next section gives some background information on the skyline operator and research on ranking in the field of machine learning. Our approach to ranking of skylines is then introduced in Section 3 and empirically evaluated in Section 4. Section 5 outlines some reasonable extensions to be addressed in future work, and Section 6 concludes the paper.

2 Background

2.1 The Skyline Operator

Consider a set of objects $\mathbb{O}$ represented in terms of a fixed number d of attributes with preferentially ordered domains, that is, "the less the better" or "the more the better" attributes. In multi-criteria decision making, such attributes are also called *criteria*; typical examples are the price of a hotel and its distance from the beach. We shall denote the i^{th} attribute by A_i and its domain by $\mathcal{D}_i$. Thus, an element of $\mathbb{O}$ is a vector

$$\mathbf{a} = (a_1 \dots a_d) \in \mathcal{D}_1 \times \dots \times \mathcal{D}_d.$$

Without loss of generality, we restrict ourselves to "the more the better" attributes.

Given to objects $\mathbf{a}, \mathbf{b} \in \mathbb{O}$, the former is said to (Pareto) dominate the latter, $\mathbf{a} \succ \mathbf{b}$, if $a_i \geq b_i$ for all and $a_i > b_i$ for at least on $i \in \{1 \dots d\}$. An object $\mathbf{b}$ is non-dominated

if $\mathbf{a} \not\succ \mathbf{b}$ for all $\mathbf{a} \in \mathbb{O}$. The skyline of $\mathbb{O}$ is then given by

$$\mathbb{S} \stackrel{\text{df}}{=} \{\, \mathbf{a} \in \mathbb{O} \mid \mathbf{a} \text{ is non-dominated} \,\}.$$

As mentioned previously, the skyline operator has recently attracted a lot of attention in the field of databases. A typical skyline query in SQL syntax may look as follows [Borzsonyi *et al.*, 2001]:

```
SELECT *
FROM Hotels
WHERE city = 'New Port'
SKYLINE OF price MIN, distance
   MIN
```

This query returns the set of hotels that are Pareto-optimal with respect to the dimensions price and distance (from the beach), which both ought to be minimized. Fig. 1 gives a graphical illustration of this example.

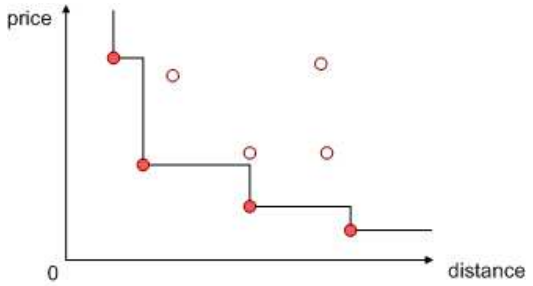

Figure 1: Skyline (filled points) for the hotel example.

The problem of computing a skyline in an efficient way has received special attention, and a large number of methods has already been devised (e.g., [Papadias *et al.*, 2005; Chomicki *et al.*, 2002; Kossmann *et al.*, 2002]). In this regard, index-based methods [Tan *et al.*, 2001] perform especially well, in particular when the size of the dataset is large. As potential disadvantages, one may note that such methods are restricted to numerical attributes for which an index can be created, and that index-based methods become problematic for high dimensions ("*curse of dimensionality*"). In our current implementation, we resort to the simple *block nested loop* approach, for which no kind of preprocessing is required [Borzsonyi *et al.*, 2001]. It maintains a set S of objects in main memory, which is initially empty, and scans the dataset. For each element $\mathbf{a} \in \mathbb{O}$, there are three possibilities:

(a) $\mathbf{a}$ is dominated by an object in S and, hence, can be discarded;

(b) one or more objects in S are dominated by $\mathbf{a}$ and, hence, can be replaced by $\mathbf{a}$;

(c) neither (a) nor (b) applies, so $\mathbf{a}$ is added to S without removing other elements.

One easily verifies that S finally corresponds to the skyline of $\mathbb{O}$, that is, $S = \mathbb{S}$.

Apart from algorithms for skyline computation, a number of conceptual modifications of skylines has been proposed in the literature, including, e.g., dynamic skylines [Papadias *et al.*, 2005], subspace skylines [Pei *et al.*, 2006], and skybands [Papadias *et al.*, 2005].

2.2 Ranking in Machine Learning

Ranking problems have received a great deal of attention in the field of machine learning in recent years. Here, the term "ranking" is used in different ways. In particular, a basic distinction between so-called *label ranking* and *object ranking* can be made [Fürnkranz and Hüllermeier, 2005]. The problem of label ranking can be seen as an extension of the basic setting of classification learning. Instead of learning a model that predicts, for each query instance, one among a finite set of class labels, the problem is to learn a model that predicts a complete ranking of all labels [Fürnkranz and Hüllermeier, 2003; Har-Peled *et al.*, 2002].

The problem considered in this paper is more related to object ranking or "learning to order things" (see, e.g., [Cohen *et al.*, 1999; Domshlak and Joachims, 2005]). Here, the task is to learn a function that, given a subset of objects $\mathbb{O}$ from an underlying reference set $\mathcal{X}$ as input, outputs a ranking of these objects. Training data typically consists of exemplary pairwise preferences of the form $\mathbf{x} \succ \mathbf{x}'$, with $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. As a key difference between object and label ranking, note that the latter associates a ranking of a fixed number of labels with every instance $\mathbf{x}$, whereas the former is interested in ranking the instances themselves.

From a learning point of view, an important issue concerns the evaluation of a predicted ranking. To this end, a loss function is needed that, given a true ranking τ and a prediction $\widehat{\tau}$ thereof, measures the "distance" between the former and the latter. The loss function is important, as it reflects the purpose that a ranking is used for. For example, in order to evaluate a ranking as a whole, one can resort to well-known rank correlation measures such as Kendall's tau [Kendall, 1955]. In many applications, however, the ranking itself is not of primary concern. Instead, a ranking is only used as a means to find certain objects, hence, only the positions of these objects are important. In this case, measures such as *precision* and *recall*, commonly used in information retrieval, are more suitable. In our experimental study, we shall use both types of measures (see Section 4.2).

3 Ranking on Skylines

As mentioned earlier, our goal is to give a *user-specific* answer to a skyline query by refining the general Pareto-dominance into a more specific preference relation valid for the user. In this regard, an important question is how to represent the user's preference. A common approach is to use a *utility function* for this purpose, that is, a mapping $U : \mathbb{O} \to \mathbb{R}$ that assigns a real utility degree to each object $\mathbf{a} \in \mathbb{O}$. Obviously, given a utility function, ranking becomes quite easy: $\mathbf{a}$ will precede $\mathbf{b}$ in the user-specific ranking if $U(\mathbf{a}) \geq U(\mathbf{b})$.

The assumption of a (latent) utility function may appear quite restrictive. However, apart from the fact that this assumption is commonly made also in many other fields, ranging from economic utility theory to information retrieval, it should be mentioned that the user is not supposed to be aware of this function, let alone to reveal it in an explicit form. In fact, we shall not require user feedback in the form of utility degrees for particular objects, i.e., we shall not directly ask for utility degrees $U(\mathbf{a})$. Instead, we shall only ask for the expression of *comparative preferences* of the form "I like object $\mathbf{a}$ more than object $\mathbf{b}$", which is much weaker and arguably more easy to elicit. Information of that kind imposes a constraint on the utility function, namely $U(\mathbf{a}) > U(\mathbf{b})$. From a learning point of view, the basic problem is hence to find a utility function $U(\cdot)$ that is compatible with a set of constraints of this type. As

a potential advantage of learning a latent utility function let us also mention that, provided that reasonable assumptions about this function can be made, this is a form of background knowledge that can highly increase the efficacy of the learning process.

The *monotonicity* required for a utility function in our context constitutes an interesting challenge from a machine learning point of view. In fact, recall that all attributes are assumed to be "the more the better" criteria. Therefore,

$$(\mathbf{a} \geq \mathbf{b}) \;\Rightarrow\; (U(\mathbf{a}) \geq U(\mathbf{b})) \tag{1}$$

should hold for all $\mathbf{a}, \mathbf{b} \in \mathbb{O}$, where $\mathbf{a} \geq \mathbf{b}$ means $a_i \geq b_i$ for all $i = 1 \ldots d$. Interestingly, this relatively simple property is not guaranteed by many standard machine learning algorithms. That is, a model that implements a utility function $U(\cdot)$, such as a decision tree, may easily violate the monotonicity property, even if this condition is satisfied by all examples used as training data.

In this paper, we shall proceed from a very simple model, namely a linear utility function

$$U(\mathbf{a}) \;=\; \langle \mathbf{w}, \mathbf{a} \rangle \;=\; w_1 a_1 + \ldots + w_d a_d, \tag{2}$$

for which monotonicity can easily be guaranteed. In fact, for the model (2), the monotonicity property is equivalent to the non-negativity of the weight vector $\mathbf{w}$, i.e., $w_i \geq 0$ for $i = 1 \ldots d$.

Despite its simplicity, the linear model (2) has a number of merits. For example, it is easily interpretable, as a weight w_i is in direct correspondence with the *importance* of an attribute. Thus, it also allows one to incorporate additional background knowledge, e.g., that attribute A_i is at least twice as important as Attribute A_j, in a convenient way $(w_i > 2w_j)$. Finally, the linear model is attractive from a machine learning point of view, as it is amenable to efficient learning algorithms and, moreover, to non-linear extensions via "kernalization" [Schölkopf and Smola, 2001].

Before going into more technical detail, we give a rough outline of our approach as a whole. As a point of departure, we consider a user who is searching a database for an object that satisfies his needs. Roughly speaking, we assume that the user is searching an object with high enough utility (a "top-K" object) but does not necessarily insist on finding the optimal one.

1. The first step consists of computing the skyline $\mathbb{S}$ of a relation constructed by the user (e.g., by an SQL query); in particular, this involves a projection to a subset $A_1 \ldots A_d$ of attributes the user considers relevant. The objects in $\mathbb{S}$ are the potential candidates regarding the user's final choice.
2. Starting with a utility function trained on a small initial training set,[1] the objects $\mathbb{S}$ are sorted according to their degree of utility, and the ranking is presented to the user.
3. The user is asked to inspect the ranking, typically by looking at the top elements. If a suitable object is found, the process terminates.
4. In case the user is not yet satisfied, he will be asked for additional feedback, which in turn is used to expand the training data.
5. The preference model (utility function) is re-trained, an improved ranking is derived, and the process continues with 3.

[1] A minimal number of training examples is necessary to make the learning problem "well-posed". This number depends on the learning algorithm and the underlying model class. In our experiments, we always started with 5 exemplary pairwise preferences.

3.1 The Learning Algorithm

Suppose to be given a set of training data $\mathbb{T}$, which consists of pairwise preferences of the form $\mathbf{a} \succ \mathbf{b}$, where $\mathbf{a}, \mathbf{b} \in \mathbb{S}$. As mentioned previously, the basic learning problem is to find a utility function which is as much as possible in agreement with these preferences and, moreover, satisfies the monotonicity constraint (1). Besides, this function should of course generalize as well as possible beyond these preferences.

Due to the assumption of a linear utility model, our learning task essentially reduces to a binary classification problem: The constraint $U(\mathbf{a}) > U(\mathbf{b})$ induced by a preference $\mathbf{a} \succ \mathbf{b}$ is equivalent to $\langle \mathbf{w}, \mathbf{a} - \mathbf{b} \rangle > 0$ and $\langle \mathbf{w}, \mathbf{b} - \mathbf{a} \rangle < 0$. From a binary classification point of view, $\mathbf{a} - \mathbf{b}$ is hence a positive example and $\mathbf{b} - \mathbf{a}$ is a negative one.

Binary classification is a well-studied problem in machine learning, and a large repertoire of corresponding learning algorithms is available. In our approach, we use a Bayes point machine, which seeks to find the midpoint of the region of intersection of all hyperplanes bisecting the version space into two halves of equal volume. This midpoint, the Bayes point, is known to be approximated by the center of mass of the version space [Herbrich *et al.*, 2001]. More specifically, we use an approximate method proposed in [Herbrich *et al.*, 2001] that makes use of an ensemble of perceptrons trained on permutations of the original training data. This approach has several advantages, notably the following: Firstly, it allows us to incorporate the monotonicity constraint in a relatively simple way. Secondly, as will be detailed in Section 3.2, the ensemble of perceptrons is also useful in connection with the selection of informative queries given to the user.

The simple perceptron algorithm is an error-driven online algorithm that adapts the weight vector $\mathbf{w}$ in an incremental way. To guarantee monotonicity, we simply modify this algorithm as follows: Each time an adaptation of $\mathbf{w}$ produces a negative component $w_i < 0$, this component is simply set to 0. Roughly speaking, the original adaptation is replaced by a "thresholded" adaptation. In its basic form, the perceptron algorithm provably converges after a finite number of iterations, provided the data is linearly separable. Even though we shall not go into further detail here, we note that this property is provably preserved by our modification.

The center of mass of the version space (and hence the Bayes point) is approximated in terms of the average of the perceptrons' weight vectors. Obviously, monotonicity of the single perceptrons implies monotonicity of this approximation.

3.2 Generating Queries

In case the user is not satisfied with the current ranking, our approach envisions a training step in which the model is updated on the basis of additional user feedback. This feedback is derived from a query, in which the user is asked for his preference regarding two objects $\mathbf{a}$ and $\mathbf{b}$ and, correspondingly, consists of a pairwise preference $\mathbf{a} \succ \mathbf{b}$ or $\mathbf{a} \prec \mathbf{b}$ that complements the training set $\mathbb{T}$. The simplest way to generate a query pair $(\mathbf{a}, \mathbf{b})$ is to choose it at random from $\mathbb{S} \times \mathbb{S}$. However, realizing that the information content

of different query pairs can be quite different, the goal of this step should of course be the selection of a maximally informative query, i.e., an example that helps to improve the current model as much as possible. This idea of generating maximally useful examples in a targeted way is the core of *active learning* strategies.[2]

In the literature, various strategies for active learning have been proposed, most of them being heuristic approximations to theoretically justified (though computationally or practically infeasible) methods. Here, we resort to the *Query By Committee* approach [Seung *et al.*, 1992]. Given an ensemble (committee) of models, the idea is to find a query for which the disagreement between the predictions of these models is maximal. Intuitively, a query of that kind corresponds to a "critical" and, therefore, potentially informative example.

In our case, the models are given by the perceptrons involved in the Bayes point approximation. Moreover, two models disagree on a pair $(\mathbf{a}, \mathbf{b}) \in \mathbb{S} \times \mathbb{S}$ if one of them ranks $\mathbf{a}$ ahead of $\mathbf{b}$ and the other one $\mathbf{b}$ ahead of $\mathbf{a}$.

Needless to say, various strategies to find a maximally critical query, i.e., a query for which there is a high disagreement between the committee members, are conceivable. Our current implementation uses the following, relatively simple approach: Let $\mathcal{W} = \{\mathbf{w}_1 \dots \mathbf{w}_m\}$ be the set of weight vectors of the perceptrons that constitute the committee, respectively. In a first step, the two maximally conflicting models are identified, that is, two weight vectors $\{\mathbf{w}_i, \mathbf{w}_j\} \subset \mathcal{W}$ such that $\|\mathbf{w}_i - \mathbf{w}_j\|$ becomes maximal. Then, the two rankings τ_i and τ_j associated, respectively, with these models are considered. Starting at the top of these ranking, the first conflict pair $(\mathbf{a}, \mathbf{b})$ is found and selected as a query; obviously, this pair is identified by the first position p such that τ_i and τ_j have different objects (namely $\mathbf{a}$ and $\mathbf{b}$, respectively) on this position.[3]

4 Experimental Results

This section presents the results of some experimental studies that we conducted in order to get a first idea of the efficacy of our approach. In this regard, an important question was whether our idea of using machine learning techniques is effective in the sense that it helps to improve the ranking quality relatively quickly, that is, with an acceptable amount of user feedback. Besides, we investigated more specific questions, such as the increase in performance due to the use of a monotone learner and an active learning strategy, and the dependence of the ranking quality (training effort) on the dimensionality of the data.

4.1 Data

Our experiments are based on both artificial and real-world data. The artificial data is repeatedly extracted from a set of 50,000 points, generated at random according to a uniform distribution in the 9-dimensional unit hypercube. First, the skyline of these points is computed. Then, for each experiment, a random weight vector $\mathbf{w}$ is generated (whose entries w_i are independent and uniformly distributed in $[0, 1]$) and the skyline is sorted according the utility degrees defined by this vector.

As real-world data, we used the ranking of the top-200 universities world-wide provided by [O'Leary, 2006]. This data set is particularly suitable due to the following reasons: Firstly, it includes information about the ground-truth, namely the correct ranking. Secondly, the data fits the setting of Skyline computation, as the universities are evaluated in terms of six (normalized) numerical attributes (peer review score, recruiter review score, international faculty score, international students score, staff-to-student ratio, citation-to-staff ratio). Thirdly, the data even meets the assumptions of our linear utility model, as the universities are ranked according to a total score which is a weighted linear combination of the individual scores.

4.2 Quality Measures

To measure the quality of a prediction, a kind of distance function is needed that compares a predicted ranking $\widehat{\tau}$ with the true target ranking τ. As mentioned before, different types of measures can be used for this purpose. To measure the quality of the predicted ranking as a whole, we use the well-known Kendall tau coefficient that essentially calculates the number of pairwise rank inversions, i.e., the number of *discordant* pairs $(\mathbf{a}, \mathbf{b})$:

$$\#\left\{(\mathbf{a}, \mathbf{b}) \mid \tau(\mathbf{a}) < \tau(\mathbf{b}),\, \hat{\tau}(\mathbf{a}) > \hat{\tau}(\mathbf{b})\right\},$$

where $\tau(\mathbf{a})$ is the position of object $\mathbf{a}$ in the ranking τ. More specifically, the Kendall tau coefficient normalizes this number to the interval $[-1, +1]$ such that $+1$ is obtained for identical rankings and -1 in the case of reversed rankings.

To complement the rank correlation, we employed a second measure that is closely related to the recall measure commonly used in information retrieval. Let $\mathcal{K}$ be the set of top-k elements of the ranking τ, that is, $\mathcal{K} = \{\mathbf{a} \in \mathcal{S} \mid \tau(\mathbf{a}) \leq k\}$, where k is an integer that is usually small in comparison with the size of the skyline (as a default value, we use $k = 10$); likewise, let $\widehat{\mathcal{K}}$ denote the top-K elements of $\widehat{\tau}$. We then define

$$\text{recall}(\tau, \widehat{\tau}) = \frac{\#(\mathcal{K} \cap \widehat{\mathcal{K}})}{k}. \tag{3}$$

This measure corresponds to the percentage of true among the predicted top-k elements. It is motivated by the assumption that, typically, a user will only check the top-k elements of a ranking. Thus, the more $\mathcal{K}$ and $\widehat{\mathcal{K}}$ are in agreement, the higher the chance that the user finds a satisfying object.

4.3 Experiment 1

In a first experiment, we applied our approach to the data sets described in Section 4.1. To investigated the effect of ensuring monotonicity of the learner, we used two different versions:

- Monotone: Our method that ensures monotonicity (and uses active learning to generate queries).
- Non-monotone: The non-monotone version of our learning algorithm, that is, a Bayes point machine using standard perceptrons as base learners.

The results are shown in Fig. 2–5. As can be seen, incorporating monotonicity seems to have an important effect on the predictive accuracy of the learner.

[2] The idea of active learning has already been applied in other fields of information retrieval, for example in image retrieval [Tong and Chang, 2001].

[3] In principle, an additional strategy is needed for the case where $\tau_i = \tau_j$. However, even though this problem is theoretically possible, it never occurred in our experiments. Therefore, we omit further details here.

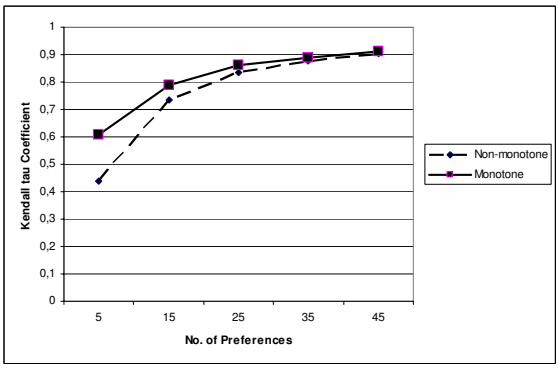

Figure 2: Rank correlation for synthetic data: Monotone vs. non-monotone learning.

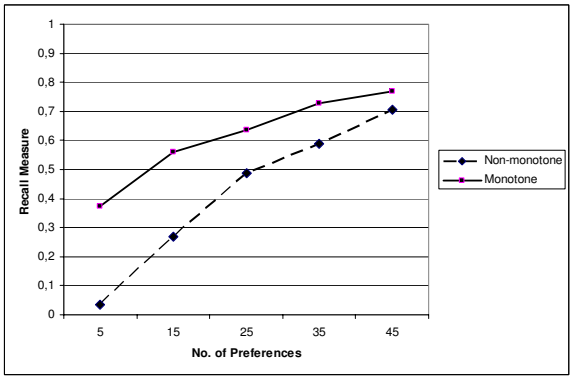

Figure 3: Recall for synthetic data: Monotone vs. non-monotone learning.

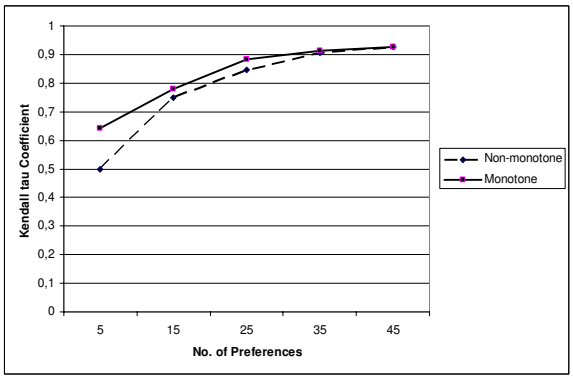

Figure 4: Rank correlation for real data: Monotone vs. non-monotone learning.

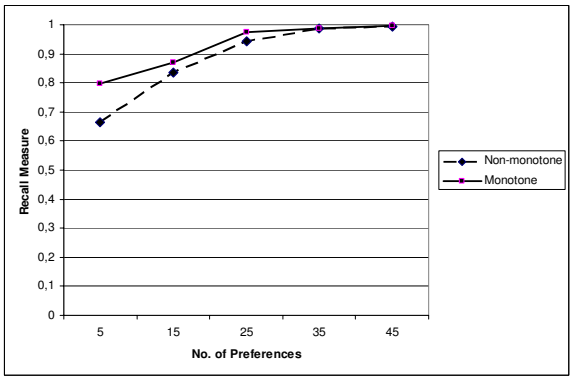

Figure 5: Recall for real data: Monotone vs. non-monotone learning.

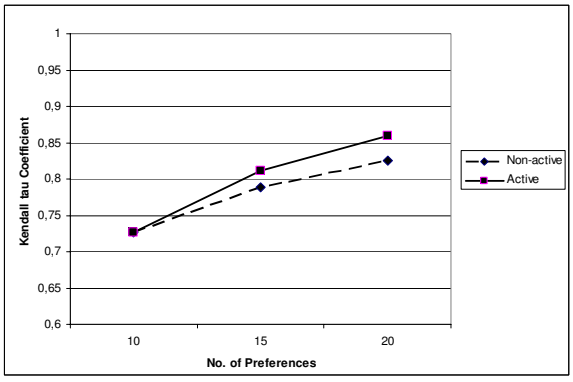

Figure 6: Rank correlation for synthetic data: Active vs. non-active learning.

4.4 Experiment 2

In a second experiment, we investigated the effect of our active learning strategy. To this end, we compared the results for two different approaches:

- Active: Our method that selects queries in an active way (and ensures monotonicity).
- Non-active: The method obtained by replacing our active learning component by a non-active strategy that selects a query at random, i.e., by randomly selecting two objects $\mathbf{a}, \mathbf{b} \in \mathbb{S}$.

The results are shown in Fig. 6–9. As can be seen, active learning indeed pays off and clearly outperforms the alternative strategy of selecting queries at random.

4.5 Experiment 3

In a third experiment, we investigated the influence of the dimensionality of the data. To this end, we used projections of the original synthetic data set to subspaces of different dimensions. The corresponding performance curves are shown in Fig. 10. Since the dimensionality of the data does have an influence on the size of the skyline and, therefore, on the length of a ranking, the recall measure (3) does not guarantee a fair comparison. Therefore, the performance is only compared in terms of rank correlation.

As expected, the results indicate that the difficulty of the problem increases with the dimensionality of the data. Fortunately, however, the dependence between dimensionality and training effort seems to be "only" linear. This is suggested by the results shown in Fig. 11, where the number of queries needed to reach a certain quality level is plotted against the dimensionality of the data.

5 Extensions

Despite the promising results reported in the previous section, our approach calls for several extensions. In the following, we shall outline some concrete points that we plan to address in future work.

- *More general utility models:* Due to the fact that a ranking is to some extent insensitive toward modifications of the utility model (different models may induce the same or at least similar rankings), sufficiently good rankings may already be obtained with utility models that are only approximately correct. Still, the assumption of a linear utility function is of course quite restrictive, and such models will probably not be flexible enough in practical applications. Therefore, we already conducted first experiments with more general,

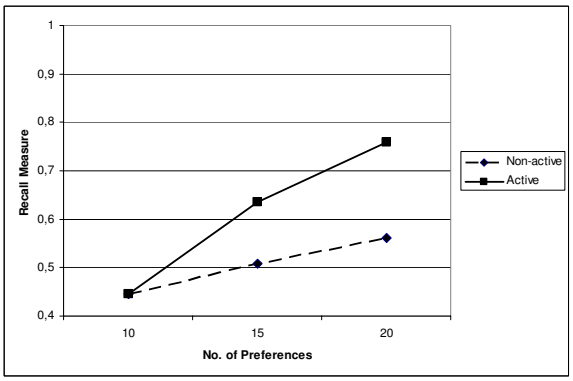

Figure 7: Recall for synthetic data: Active vs. non-active learning.

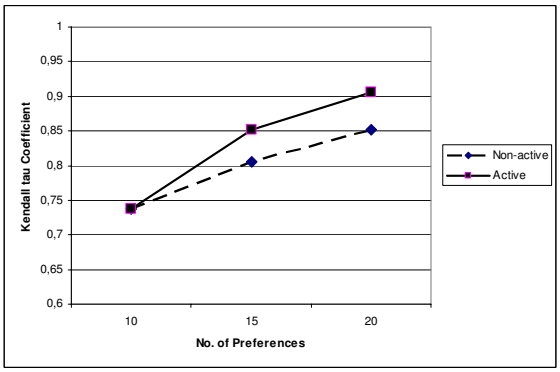

Figure 8: Rank correlation for real data: Active vs. non-active learning.

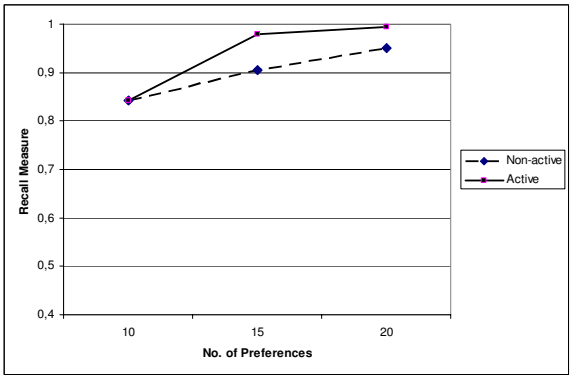

Figure 9: Recall for real data: Active vs. non-active learning.

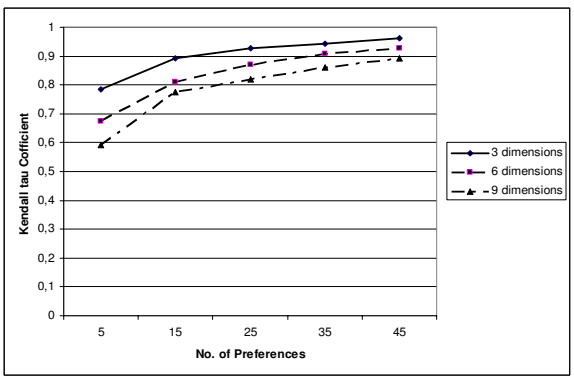

Figure 10: Rank correlation for synthetic data depending on the dimension of the data.

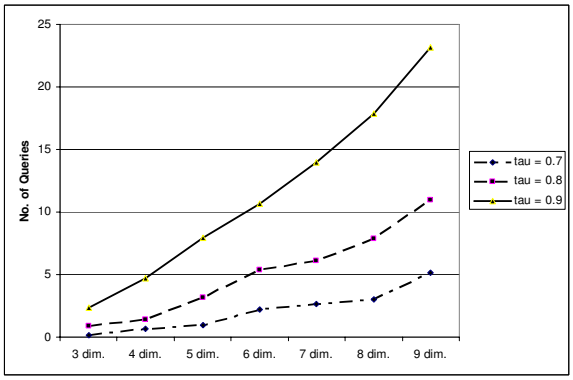

Figure 11: Number of queries needed to reach a certain quality level.

non-linear utility models and obtained results quite comparable to those presented in this paper. The basic idea is to "kernalize" the linear utility function, which is a standard approach in the field of kernel-based learning.

- *Robust learning algorithms:* In practice, user feedback will not always be correct. Instead, the preferences stated by a user may contain errors. From a machine learning point of view, it is therefore important to make learning algorithms tolerant toward "noisy" training data.
- *Learning order relations on attribute domains:* The standard setting of skyline computation assumes all attributes to be criteria, i.e., to have totally ordered domains. In practice, this assumption is quite restrictive and does obviously not hold for attributes such as, say, `color` [Balke and Güntzer, 2005]. From a learning point of view, an obvious idea is to start without prior assumptions, and instead to learn the corresponding order relation from the user's revealed preferences, e.g., to learn that a user prefers `green` to `red`. In this regard, one may also give up the assumption of a total order and allow for partial orders. Moreover, since preferences for attribute values are clearly not independent, it may become necessary to learn order relations not only in one-dimensional but perhaps also in higher-dimensional subspaces.
- *Integration of skyline computation and ranking*: Until now, skyline computation and ranking are simply carried out in succession, so the integration between them is not very tight. This will change, however, in connection with the aforementioned learning of order relations on attribute domains, since a change of an order relation will also change the dominance relation between objects and, therefore, the skyline. Consequently, skyline computation and ranking will have to be carried out in an alternate rather than a consecutive manner.
- *Deviating from the skyline*: A monotone utility model is in agreement with Pareto-dominance in the sense that the object $\mathbf{a}^* \in \mathbb{O}$ with highest utility is non-dominated and, hence, an element of the skyline. The other way round, however, it is well possible that, according to a given utility model, a user prefers an object $\mathbf{a}$ which is not on the skyline to an object $\mathbf{b}$ which is on the skyline. In principle, this is unimportant as long as we assume that a user is only searching for a

single object. However, if the user is looking for more than one object, it will be useful to include dominated alternatives in the ranking, namely those objects with high utility. And even if the user is searching only a single object, including such objects may speed up the search process, as it increases the likelihood of finding a satisfying alternative. On the other hand, one may think of disregarding those points of the skyline which have a rather low utility, as it is unlikely that such elements will ever move to the top of the ranking. A corresponding pruning strategy would offer another means to increase efficiency. Of course, both measures, the adding of dominated alternatives as well as the pruning of non-dominated ones, presuppose a certain degree of reliability of the current utility model.

- *Valued preferences:* Instead of only asking a user whether he prefers object $\mathbf{a}$ to $\mathbf{b}$ or $\mathbf{b}$ to $\mathbf{a}$, one may allow him to express a *degree of preference*; as a special case, this includes the expression of indifference. An interesting question is whether additional information of that kind can be helpful for producing good rankings.
- *From single users to user groups:* In this paper, we focused on learning the preferences of a single user. In practice, a system will of course be used by more than one person. In this regard, an obvious idea is to exploit the preferences expressed by one user to support another one. In fact, this strategy can help to reduce the feedback requested from the latter user, provided that the preferences are sufficiently similar. A hypothetical transfer of preferences could be done, for example, on the basis of the active user's preferences expressed so far, an idea which is also on the basis of collaborative filtering techniques [Goldberg *et al.*, 1992; Breese *et al.*, 1998]. Another idea is to use clustering techniques in order to find homogeneous groups of users, and to learn utility models that are representative of these groups [Chajewska *et al.*, 2000; 2001].

6 Conclusion

This paper has presented a first study on the combination of skyline computation and machine learning techniques for ranking. The basic motivation is to make skyline queries more user-friendly by providing answers in the form of a ranking of objects, sorted in terms of the user's preferences, instead of returning an unordered and potentially huge answer set.

Our first results are promising in the sense that, apparently, relatively accurate rankings can be produced with an acceptable effort in terms of user feedback. The empirical results are interesting also at a more technical level, since they show that enforcing monotonicity of the learned utility model does indeed improve performance, just like using an active learning strategy to select informative queries.

As outlined in the previous section, we plan to extend our approach in various directions, some of which are currently investigated as part of ongoing work.

References

[Balke and Güntzer, 2005] W. Balke and U. Güntzer. Efficient skyline queries under weak Pareto dominance. In R. Brafman and U. Junker, editors, *Proc. Multidisciplinary IJCAI-05 Workshop on Advances in Preference Handling*, pages 1–6, Edinburgh, Scotland, 2005.

[Borzsonyi *et al.*, 2001] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *IEEE Conf. on Data Engineering*, pages 421–430, Heidelberg, Germany, 2001.

[Breese *et al.*, 1998] J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collarborative filtering. In *Proceedings UAI–98*, Madison, WI, 1998.

[Chajewska *et al.*, 2000] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings AAAI–2000*, pages 363–369, 2000.

[Chajewska *et al.*, 2001] U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *18th International Conference on Machine Learning, ICML–01*, pages 35–42, 2001.

[Chomicki *et al.*, 2002] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting, 2002.

[Cohen *et al.*, 1999] W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10, 1999.

[Detyniecki *et al.*, 2006] M. Detyniecki, JM. Jose, A. Nürnberger, and CJ. van Rijsbergen, editors. *Adaptive Multimedia Retrieval: User, Context, and Feedback*. Number 3877 in LNCS. Springer-Verlag, Heidelberg, 2006.

[Domshlak and Joachims, 2005] Carmel Domshlak and Thorsten Joachims. Unstructuring user preferences: Efficient non-parametric utility revelation. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, page 169, Arlington, Virginia, 2005. AUAI Press.

[Fürnkranz and Hüllermeier, 2003] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Proc.* ECML–2003*, 13th European Conference on Machine Learning*, Cavtat-Dubrovnik, Croatia, September 2003.

[Fürnkranz and Hüllermeier, 2005] J. Fürnkranz and E. Hüllermeier. Preference learning. *Künstliche Intelligenz*, 1/05:60–61, 2005.

[Goldberg *et al.*, 1992] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[Har-Peled *et al.*, 2002] Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification and ranking. Technical report, Champaign, IL, USA, 2002.

[Herbrich *et al.*, 2001] Ralf Herbrich, Thore Graepel, and Colin Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.

[Kendall, 1955] M.G. Kendall. *Rank correlation methods*. Charles Griffin, London, 1955.

[Kossmann *et al.*, 2002] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings VLDB-2002*, Hong Kong, China, 2002.

[O'Leary, 2006] John O'Leary. World university rankings editoral - global vision ensures healthy competition. *The Times Higher Education Supplement*, 2006.

[Papadias *et al.*, 2005] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.

[Pei *et al.*, 2006] Jian Pei, Yidong Yuan, Xuemin Lin, Wen Jin, Martin Ester, Qing Liu, Wei Wang, Yufei Tao, Jeffrey Xu Yu, and Qing Zhang. Towards multidimensional subspace skyline analysis. *ACM Trans. Database Systems*, 31(4):1335–1381, 2006.

[Robertson and Jones, 1976] SE. Robertson and KS. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[Rochio, 1971] JJ. Rochio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System*, pages 313–323. Prentice Hall, Englewood Cliffs, NJ, 1971.

[Salton and Buckley, 1990] G. Salton and C. Buckley. Improving retrieval performance by retrieval feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[Schölkopf and Smola, 2001] B. Schölkopf and AJ. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[Seung *et al.*, 1992] H. S. Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992.

[Shen and Zhai, 2005] Xuehua Shen and Chengxiang Zhai. Active feedback in ad-hoc information retrieval. In *Proc. CIKM'2005, ACM Conference on Information and Knowledge Management*, pages 59–66, 2005.

[Shen *et al.*, 2005] Xuehua Shen, Bin Tan, and Chengxiang Zhai. Implicit user modeling for personalized search. In *Proc. CIKM'2005, ACM Conference on Information and Knowledge Management*, pages 824–831, 2005.

[Tan *et al.*, 2001] Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. Efficient progressive skyline computation. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 301–310, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Tong and Chang, 2001] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international Conference on Multimedia*, pages 107–118, Ottawa, Canada, 2001.