What do you prefer? Using Preferences to Enhance Learning Technology

Philipp Kärger, Daniel Olmedilla, Fabian Abel, Eelco Herder, and Wolf Siberski

***Abstract*— While the growing number of learning resources increases the choice for learners on how, what and when to learn, it also makes it more and more difficult to find the learning resources that best match the learners' preferences and needs. The same applies to learning systems that aim to adapt or recommend suitable courses and learning resources according to a learner's wishes and requirements. Improved representations for a learner's preferences as well as improved search capabilities that take these preferences into account leverage these issues. In this paper, we propose an approach for selecting optimal learning resources based on preference-enabled queries. A preference-enabled query does not only allow for hard constraints (like 'return lectures about Mathematics') but also for soft constraints (such as 'I prefer a course on Monday, but Tuesday is also fine') and therefore allows for a more fine-grained representation of a learner's requirements, interests, and wishes. We show how to exploit the representation of a learner's wishes and interests with preferences and how to use preferences in order to find optimal learning resources. We present the Personal Preference Search Service (PPSS), which offers significantly enhanced search capabilities for learning resources by taking the learner's detailed preferences into account.**

***Index Terms*— preference, learning resource, metadata, personalization**

I. Introduction

Personalization of online learning has been a hot issue in the elearning community since the 1990s. By making use of the learners' individual goals, preferences, interests, and knowledge, the interaction with the students can be adapted to their individual needs. Personalization can serve several goals, including tailoring of information presentation and helping learners to find information by support for browsing and search [1]. In particular the latter goal has become very relevant, now that learning has become a lifelong activity—in our knowledge-based society workers need to continuously increase their knowledge and competences to keep up-to-date [2].

The community of lifelong learners is varied, as is the variety of learning activities that are offered—ranging from targeted group-based courses to self-organized home-based learning making use of on-line material. These courses might target students with specific cognitive abilities and learning styles [3]. Moreover, due to other obligations—professional and personal—, learners may have specific constraints with respect to study load, location and the time of course delivery. It is a known fact that learners may have a lot of different preferences of all these kinds [4] and it is a challenge to consider all these particularities in the personalization process.

For this reason, we present a technique based on *Preference Handling* for representing a learner's preferences and exploit this representation for selecting learning resources that best match these preferences. This technique is not only relevant during the process of searching for learning resources, but also for the adaptation of course delivery to the goals, preferences, interests, and knowledge that are stated in the form of preferences and stored in the learner model.

Search capabilities in educational repositories and networks have been improved in recent years by the introduction of personalization and semantic-based queries. These techniques are typically realized by extending the query with hard constraints, which are either stored in the learner model or manually specified by the learner during the search process. These hard constraints represent the user's wishes, that is, conditions that should be fulfilled. But a closer look reveals that in most cases a learner's constraints are not hard constraints. Typically, a user may want to express that she wants "courses that are in English, but otherwise German would also suffice, and that courses preferably should be given on Monday rather than Tuesday or Friday". The words "preferably", "otherwise" and "rather than" indicate soft constraints using which a user specifies *what she prefers*, in other words her preferences. These preferences represent a list of alternatives that can be used to filter out suboptimal, non-relevant results, and let the learner concentrate on the most suitable alternatives. For example, if two courses are found, both of them on Monday, and one of the courses is in English and the other is in German, intuitively the latter can be discarded, since given the same (or less preferred) conditions, the user prefers English over German. This way, only the most optimal results according to the user's preferences are returned. This improves the satisfaction of the users and reduces the time they must spend in order to scan large query result sets that also contain non-preferred (suboptimal) options.

It is important to note that the term *user preferences* has been extensively used in the field of user modeling [5] and adaptive hypermedia [3], [6]. Typically, these user preferences are a set of properties that specify the learners' goals, interests, and needs (and which are added in the queries as hard constraints or used to compute rankings). By contrast, our method is more expressive: it allows these learner properties to be modeled as soft constraints and allows users to indicate which properties they prefer to another by allowing for a *preference order*.

This paper describes how preferences can be used to create a fine-grained model of a learner's needs, which represents more accurately their wishes and interests. It shows how preference-based queries can be used in order to improve the selection of learning resources. Further, we show how to efficiently retrieve learning resources that best match the user's preferences, while discarding suboptimal solutions. In [7], we provided a first idea on how to use preferences for searching learning resources. In this paper, we base our elaborations on [7] and further refine it by

Authors are with the L3S Research Center, Hannover, Germany
E-mail: {kaerger,olmedilla,abel,herder,siberski}@L3S.de

allowing for more expressive preference compositions and a more flexible and powerful approach for expressing partial orders. In particular this paper adds the following contributions to [7]:

- support for preference compositions in order to be able to combine two types of multidimenional compositions (lexicographic and Pareto).
- support for a more flexible and powerful specification of partial orders in the queries, by including a representation of placeholders in its syntax and evaluation semantics.
- development and integration of the aforementioned extensions in our implementation.
- provision of more (and more detailed) definitions, descriptions, examples, etc.

The paper is organized as follows. In Section II we motivate our approach by a running scenario. Details about preferences, theoretical background and the use of preferences in query processing are presented in Section III. Section IV shows how preferences support the selection of learning resources by applying the preference theory to our running scenario. In Section V we describe our prototype implementation and a first experimental evaluation. Finally, Section VI compares our approach with existing initiatives and Section VII concludes the paper.

II. Motivation Scenario

In the following, we picture a scenario to motivate our approach, which will be used later in the paper to demonstrate how preference-based search works and how preferences support learners in finding suitable courses. We will use this example throughout the paper for illustrating our approach.

Bob has just bought his first digital camera and now he is looking for a course about photography. He is not sure what different kinds of courses are available, but he has certain ideas of his likes and dislikes. Figure 1 summarizes the preferences mentioned in the scenario.

A. For instance, Bob prefers a class-room course, in which he can learn with and get inspired by fellow learners, above a rather solitary distance learning course.

B. Bob is neither a professional in photography nor does he plan to become one, so he does not insist on gaining a certificate. But should there be a course that *does provide* a certificate at the same or better conditions (price, etc.), he would prefer to take the one with the certificate.

C. However, he definitely would not like to have to pass an exam in order to get the certificate.

D. Bob believes that he will enjoy doing image processing with his computer. Hence, he also wants a course that comprises some kind of homework.

E. Bob wants a course offered in the evening. He would prefer a course on one of the working days, except on Monday when he has a weekly appointment with a friend for jogging. If needed, he could reschedule this appointment, though. He also likes to keep the Friday evening free for meeting with his chess club. If there are no courses available during the week, he might consider a course on Saturday or Sunday.

F. Bob would like the course to take place once a week. A course with two meetings per week, or one meeting every two weeks, would be fine as well. But he absolutely dislikes weekend block courses, as he is not willing to stay away from home for a longer time over the weekend.

G. However, since he just got his new camera he wants the course to start as soon as possible as not to lose any time.

H. As Bob is an avid cyclist, he does not mind riding up to 10 km to the course, provided that he can follow a scenic track with cycle lanes. If the course takes place in the south of the city center he can take the way through the park, otherwise he has to struggle through busy traffic.

I. Concerning financial issues, Bob also has some constraints: he is not willing to pay more than 100 euros for the course.

This combination of different interests and likes and dislikes is typical for complex domains such as courses: more often than not people are not able to exactly specify their wishes in terms of hard constraints. The expressions used in our scenario show that Bob has merely an idea of what he is looking for. These kind of soft constraints are useful for users to define their expectations, in particular when they do not yet know what exactly they are looking for. Although this set of hard and soft constraints is a common way of how a user describes her wishes, it is so far not possible to specify such complex search requests with current search interfaces. Moreover, from a learner modeling point of view it would be much more accurate to allow for these more vague statements about a user's wishes.

A learning platform that provides extended search capabilities to take into account all the known or explicitly provided hard and soft constraints is desirable. With such a platform, Bob would be able to specify some of his ideas of the desired course: it should deal with digital photography, it does not need to provide a certificate, it should start immediately, etc. Additionally, the system exploits its knowledge about Bob, such as his age, which languages he prefers beyond his native language. It also uses Bob's preferences gained from his past interactions, such as his fondness of meeting people, the location where he lives, his regular meeting on Fridays and Mondays. By taking all these constraints into account, the system would be able to perform a query comprising most of the particularities in Bob's idea of a course. Probably there will be no course that matches all of the constraints, but the system will provide Bob with a small result set, containing the courses with—according to his preferences—the lowest deviation from the given preferences and therefore from his ideal course.

III. Preferences

Preferences are a way to model a user's needs, wishes, and expectations. Beyond the specification of a preferred, single desired value or behavior (such as "I like green.") the notion of preference that we suggest allows for alternatives (such as "If possible, I prefer green. Otherwise, yellow is fine as well."). Therefore, preferences provide means for expressing soft constraints instead of hard constraints. These preference orders, which are built up by assembling several preferred alternatives, comprise precise knowledge about what a user would like to get. But the term "if possible, ..." already suggests that precise and meaningful semantics of these preference orders has to be provided: the soft constraints have to be relaxed step by step until the most preferred object, or rather the most preferred combination of attributes, is found. In this section, we will first outline some properties and different kinds of preferences. Then we will introduce the theory of preferences and show how their semantics deliver the desired behavior of a constraint relaxation.

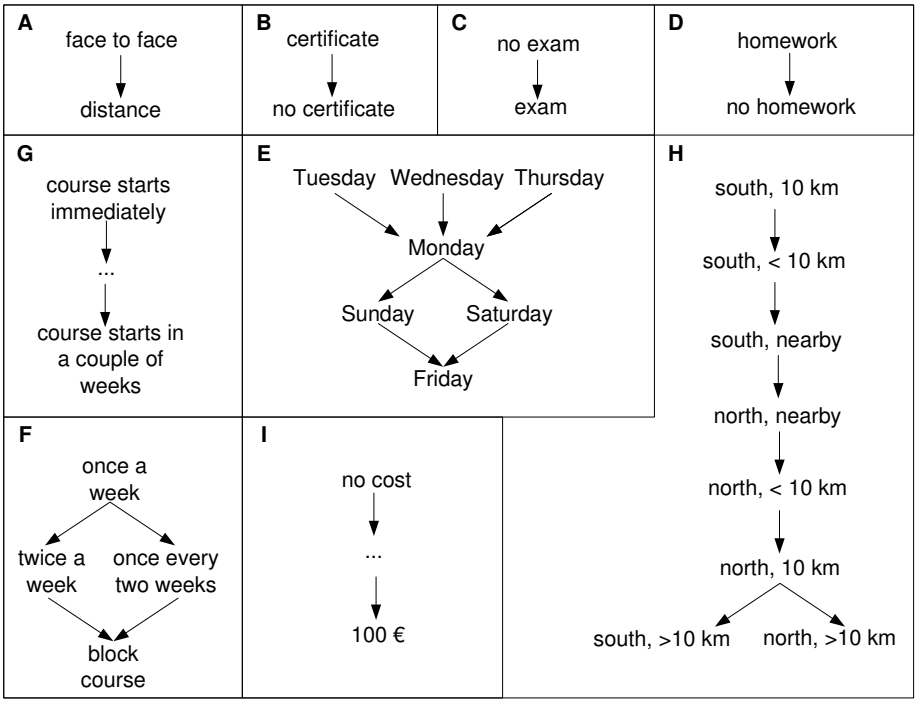

Fig. 1. Bob's preferences, represented as Hasse diagrams of the preference relations. Each preference is represented by a directed connected graph. The direction of an edge represents that the node at the beginning is preferred to the node the edge points to. Non-connected nodes represent indifferent alternative values.

A. Properties of Preferences

In this section we introduce some basic properties and requirements that a preference model may have. We will revisit these properties throughout the paper.

Total vs. partial orders. It may be difficult for Bob to define a total order preference for all the attributes of his desired course. It is a time consuming process and it may well be the case that he is indifferent for some attribute. Partial order preferences (such as Bob's preference concerning the day of the week) allow for indifferences and are more general than total order preferences (such as the one concerning a course's certificate). In Figure 1 we depicted the partial orders of Bob's preferences in the form of Hasse diagrams [8].

Conditional Preferences. A preference may vary depending on some attributes. For example, Bob preference concerning the location of the course does not only include the length of his bike ride; in addition, it includes preferences on the direction of the ride as well (in this case, whether it is north or south).

Quantitative vs. qualitative preferences. In a quantitative (weighted or numerical) model of a user's preferences, the value of to what extent a user prefers one alternative over another is explicitly given. In our scenario, none of the preferences that Bob provided were quantitative. From a usability point of view, it is not desirable that a user has to specify a numerical representation for each attribute and each attribute's value to indicate how much a certain alternative is preferred. Moreover, in most cases, the composition of several attributes (such as Bob's desired location of the course together with the desired price) is an arbitrary numerical function, in which the quantitative preference definitions of the involved attributes need to be normalized. Therefore, we argue that, in general, qualitative (i.e., non-weighted) preferences are more suitable for a user and in this work we will concentrate on qualitative preferences.

Prioritized Preferences. If a user's preference includes more than one attribute, the user may consider one attribute's preference more important than another. For example, Bob's preference concerning an exam is more important to him than the one about the certificate. Therefore, it is desirable to allow for prioritized preferences.

Default and learned preferences. It is not always needed that users define their preferences manually. Some preferences might be automatically learned from the user's behavior. As an example, if the learner's schedule is already tight on Friday, other days of the week may be more suitable. Or if a student's results in oral exams have not been as good as the results for written courses, courses with written exams may be automatically preferred by the system. Moreover, one could consider default preferences that apply to learners that fit into a certain group profile [9] or even general default preferences, such as preference for the cheapest price, courses with certification, the lowest distance, the highest reputation, etc.

Hard and soft constraints. As we already pointed out, preferences model constraints that a user may allow to be relaxed, if needed. Although we focus our work on these kinds of soft constraints, we want to point out that hard constraints may be still part of a user profile. For example, Bob's constraint that a course should not be more expensive than 100 euros is a hard constraint. Hard constraints are part of most existing user models; they are complementary to the theory we present and supported in our implementation.

B. Formal Model of Preferences

In order to model the kinds of hard and soft constraints that Bob is able to specify his preferences with, we will now introduce the notion of *Preferences* and *Preference-based Queries*. As we have seen in the scenario, we aim for advanced search for suitable courses. A query model in which users can only specify hard constraints on course characteristics does not support this kind of search. The notion of preference-based querying in the context of databases has been formalized independently by Kießling [10] and Chomicki [11]. To describe user preferences in a way that is exploitable for querying, we rely on the preference-based query formalization as proposed by Chomicki in [11]. In this extension to relational algebra, preferences are expressed as binary relations over a set of objects O.

Definition 1: [Object-Level Preference Relation] Let A be the set of available attributes of the elements in O, and U_i the respective set of possible values of an attribute a_i. Then any binary relation $\succ$ which is a subset of $(U_1 \times \ldots \times U_n) \times (U_1 \times \ldots \times U_n)$ is an object-level preference relation over the object set O.

An object level preference provides the means to compare two objects and to decide which one is preferred. But typically, a user's preferences are not directly defined on object level. Therefore, although an object-level preference allows for comparison of objects, it is not the right means to express a preference, as statements about which object is preferred to another are not explicitly given by the user. In our scenario, Bob does not provide an object-level preference: he does not state relationships between whole objects like "I prefer course A, which is held on Monday and costs 72 Euro and provides a certificate and ..., to a course B, which is held on Tuesday and costs 44 Euro and also provides a certificate and so on.". In contrast, Bob's preferences are defined on attribute level: on the attribute values of each object attribute and not on the objects themselves. For example, for the attribute "day of the week" Bob states that he prefers the value Tuesday to the value Monday. Therefore, preferences are rather stated with respect to the attribute values of each single attribute. Consequently, certain values are preferred over others, thus forming a partial order of attribute values:

Definition 2: [Attribute-Level Preference Relation] Let A be the set of available attributes of the elements in O and $S = \{a_1, \ldots, a_n\} \subset A$ a non-empty set of attributes with $V = \bigcup U_i$ the union of the attributes domains of possible values. The attribute level relation $\succ_S$, which is a subset of $V \times V$, is an attribute-level preference relation over the value set of the attributes in S.
If $|S| = 1$ we call $\succ_S$ a single-attribute preference relation. Otherwise, it is called a multi-attribute preference relation.

Example 1: Bob's preference concerning the desired location of the course is a multi-attribute preference: it spans over the attributes *direction* and *distance*. Multi-attribute preferences provide means to model conditional preferences. Bob's preference for a longer bike ride depends on the direction of the ride. This conditional preference can be modelled by specifying a relation spanning over these two attributes.

For a more convenient notation, we introduce an additional relation to denote indifference (or *incomparability*) between two objects.

Definition 3: [Indifference Relation] Given an attribute level preference relation $\succ_S$ and two objects x and y, the indifference relation $\sim_S$ is defined as $x \sim_S y \equiv x \not\succ_S y \wedge y \not\succ_S x$.
If x either dominates y (i.e., $x \succ_S y$ holds) or both are indifferent (i.e., $x \sim_S y$ holds), we write $x \succeq_S y \equiv x \succ_S y \vee x \sim_S y$.

Based on these attribute-level preference relations that are provided by the user, one is not yet able to compare objects: what if an object is better in terms of one attribute, but worse in terms of another attribute? If Bob states that Tuesday is preferred to Monday and that cheaper courses are preferred, how can one compare a more expensive course on Tuesday with a cheaper course on Monday? In other words, we need a way to combine all the attribute-level preferences in order to build up an object-level preference relation that allows us to compare objects.

This combination is provided by a so-called *composition* of attribute-level preferences:

Unidimensional Composition applies if preference relations over the same attribute have to be combined. This does not apply to our scenario and we refer the reader to [11] for more details.

Multidimensional Composition applies if attribute preference relations have to be composed that are stated over different attributes or attribute sets.

Multidimensional composition is needed when the relations are imposed over different sets of attributes, in order to expose a new preference relation over the Cartesian product of the sets of attributes. For a composed preference, the combined preference relations are called *dimensions of the composed preference relation*. According to [11], two multidimensional compositions are common:

- *lexicographic composition* combines two dimensions by considering one as more important than the other.
- *Pareto composition* allows for combining two preference relations without imposing a hierarchy on the dimensions—all dimensions are considered to be equal.

Lexicographic Composition. A lexicographic composition of two preference relations is based on the assumption that one relation can be considered more important than the other: there is a total ordering between the two attributes. Thus, objects are generally ordered according to the more important attribute and only in case of ties the less important attribute is considered to decide the order.

Definition 4: [Lexicographic Composition] The lexicographic composition $\succ_L$ of the preference relations $\succ_1, \ldots, \succ_n$ is defined as:

$$x \succ_L y \Leftrightarrow \exists k : (y \succ_k x \wedge \forall i < k : x \sim_i y).$$

The comparison of objects according to a lexicographic composition is similar to the definition of the word order in a lexicon: words starting with letters ranked higher in the alphabet are put before ones starting with lower ranked letters. Only in case both first letters are equal, the second letter is considered for deciding which word is listed first, and so on. Applied to our scenario, we may order Bob's preference concerning the exam and his preference concerning the certificate in a lexicographic manner: any course without an exam is preferred to a course requiring an exam independent of the certificate dimension. Only in case two courses show equal values in terms of the exam (either both require one or none), they are compared according to the certificate dimension.

In most of the cases, a user considers the dimensions of her preferences equally important. For example, in our scenario, most of Bob's preference dimensions play equally important roles for the comparison of courses. Therefore, the fair principle of Pareto domination has been introduced in order to combine several preference dimensions.

Pareto Composition. Pareto composition yields a new preference relation following the fair principle of Pareto domination. An object x is said to *Pareto-dominate* an object y iff x is better than y in terms of at least one of the preference relations and equal or better in terms of all other preference relations. Or, more formally:

Definition 5: [Pareto Composition] Given the preference relations $\succ_1, \ldots, \succ_n$ over the sets of attributes A_1,...,A_n, the Pareto composition $\succ_P$ of $\succ_1 \ldots \succ_n$ is defined as: $x \succ_P y \Leftrightarrow (\forall i : x \succeq_i y) \wedge \exists j : x \succ_j y$.
We say an object x Pareto dominates *another object y, iff $x \succ_p y$.*

This definition follows the principle of *weak* Pareto dominance (as used, for example, in [11] and [12]), which implies that incomparability in one preference dimension does not yield incomparability on the object level, because $\succeq_i$ contains all the incomparable attribute values for preference dimension i. In contrast to the principle of *weak* Pareto dominance, some approaches follow the *strong* Pareto dominance (such as, for example [10] and [13]), where an incomparability on the attribute level always implies incomparability on the object level: as soon as two objects are incomparable in terms of a single attribute, they are both incomparable. The choice between weak and strong Pareto dominance for the composition of preference relations makes a difference only for partial order attribute level preferences. For the composition of total order preference relations, there is no difference, as for a total order preference relation $\succ_i$ the corresponding indifference relation $\sim_i$ is empty.

Applied to our scenario, the principle of Pareto composition lets a low-cost course x dominate an expensive course y only iff in terms of all other preference relations (as imposed on the attributes *location*, *duration*, etc.) x is at least equally good as y. This principle has been exploited in the area of database systems for the so-called *skylining* [14], [15], [16]. In skyline queries, each single attribute is viewed as an *independent*, non-weighted query dimension. Best matches for skyline queries are determined according to the principle of Pareto optimality: each object that is not dominated by any other object is considered as optimal and as a best match. All these non-dominated objects are called the *skyline of the query*.

Now we have to apply the notion of preference, as developed so far, in order to select from the set of all available objects the ones that are optimal according to a given preference. This is catered by the so-called *winnow operator*, which selects all non-dominated (preferred) objects from a set of solutions given an object level preference relation.

Definition 6: [Winnow Operator ω] Given a set of objects O and a preference relation $\succ$, the winnow operator ω is defined as $\omega(O)$, in the following way:

$$\omega(O) = \{x \in O | \neg \exists x' \in O : \ x' \succ x\}.$$

C. Querying with Preferences

As pointed out earlier in this paper, *exact match* semantics of traditional databases do not sufficiently fit the queries as formulated by end users. On the one hand, this is because too specific query predicates often lead to empty result sets. For example, if Bob had specified only his *most* preferred attributes as hard constraints (i.e., connected by a logical *and*), the system would most likely come up with an empty result set, because no learning object fulfills all of Bob's most desired features. We refer to this query as a *conjunctive query*. On the other hand, too many unspecific hard constraints may yield huge numbers of results. In Bob's case that would mean that all the properties that he may think of are put into the query and connected with a logical *or*; in this case, many learning objects (including the most preferred and least preferred) will be provided to Bob. We will refer to this kind of queries as *disjunctive queries*. Both approaches lead to the fact that, in practice, users have to modify query constraints in a trial-and-error fashion until they get a result set of the right size.

In contrast to the exact match paradigm, the notion of *best match* semantics fits much better to typical user's search requests. In this case, the goal is to compute a set of results that fulfill the query's constraints as good as possible. These constraints are often referred to as soft constraints. Best match queries automatically adapt the specificity of a search to the available objects. Our proposed solution to achieve best matches is to exploit preference orders for querying. Preference-enabled queries are based on the observation that expressions of the form "I like A more than B" are easily stated by users when asked for their wishes. Therefore, it should be optimal if a query engine can derive best matches directly from such preference expressions instead of aiming only at exact matches. Besides preference-based queries, several other approaches to retrieve best matches have been proposed, such as query relaxation and top-k retrieval.

a) Query relaxation: This family of algorithms does not introduce new query language constructs (except sometimes an operator to distinguish between hard and soft constraints), but continues to loosen and/or remove query constraints until at least some results can be returned to the user [17], [18]. The advantage of this approach is that the queries themselves stay simple; only the evaluation algorithm is changed. Various relaxation algorithms have been proposed, including [19], [20], [21]. At the same time, the main advantage of query relaxation is also its main weakness: the user has no control over the relaxation process. In the worst case, the relaxation algorithm will remove the important constraints early, while keeping the less relevant. For example, Bob might end up with a certified course that starts 3 months later, because the algorithm relaxed on the starting date instead of on the less important certification constraint.

b) Top-k queries: Inspired by information retrieval algorithms for document search, query techniques that rank results by their relevance for the user have been introduced for databases as well. Instead of expressing constraints on the answers, the query includes a scoring function that assigns a relevance score to each potential answer [22]. The scoring function of a query is usually the weighted sum of individual scores for each relevant attribute. A formal extension of relational algebra by a specific top-k operator has been proposed in [23]. Algorithms and systems for efficient computation of top-k queries have been developed [24], [25], [26]. The main disadvantage of the top-k paradigm is

that in many cases it is very difficult for the user to specify the right scoring function, as this requires the user to put her wishes in a numeric relation to enable something that might be considered comparing apples with oranges. For example, it is virtually impossible for Bob to tell how many additional miles he would be willing to ride for a course starting one week earlier. But these are exactly the trade-offs that have to be specified as weights within the scoring function.

c) Preference-enabled queries: In contrast to these two approaches, preference-enabled queries do not require ranking. All objects contained in the result set are optimal according to the given preference orders. Assuming a ranking score function based on user preferences, any object that would be ranked lower (for example, according to some of the measurements in the approaches of top-k or query relaxation) would not be optimal anymore: it represents a worse alternative than the objects with a higher ranking. For this reason, any result set of a preference-enabled query is unordered: all results are equally relevant. To provide more effective search capabilities for preferences, query languages like SQL over relational databases [27] and SPARQL over RDF graphs [28] have been extended to facilitate preference-based retrieval algorithms.

In the next section, we show how preference expressions together with preference-enabled queries are applied to effectively search for learning resources.

IV. Preferences on Learning Resources

There are many ways in which on-line learning can be personalized to the learner. Preference-based reasoning provides adaptive educational engines with powerful means for selecting suitable learning resources, other learners (e.g., for collaboration) or even a suitable adaptive instructional strategy. In this section we will show how preference handling can be used to match the complexity and variety of a learner's needs and the plethora of possibilities—be it learning material or learning strategies—offered by current adaptive learning systems.

Traditional adaptation techniques include adaptive presentation and adaptive navigation. *Adaptive presentation techniques* include hiding, adding, annotating or highlighting text or multimedia material, based on their inferred relevance to the learner. *Adaptive navigation techniques* include the creation of guided tours, user-adaptive contextual menus (by means of reordering, hiding or adding items, annotation, highlighting), personalized search results, feedback during the planning process, and proactive recommendations [3]. Interface techniques to achieve these adaptations are manifold and can be used both in the process of searching for a learning resource and while interacting with the resource.

Adaptive learning is a prominent pedagogical approach where each individual learner is provided with a set of learning activities and resources that fits the individuals' learner preferences including portfolio, background knowledge, educational needs (such as learning styles) and situational circumstances of the learner. Level B of the IMS Learning Design Specification [29] provides means for creating alternative routes that address these individual differences. Similarly, popular adaptive hypermedia systems, such as AHA! and MOT provide rules that allow for addressing individual learning styles [30]. In contrast to MOT, in which authors explicitly select an adaptive instructional strategy, the AHA! system's adaptations are based on outcomes of the adaptive hypermedia's engine reasoning system.

Learning styles, or rather *cognitive styles* have been a driving factor in the design of adaptive educational hypermedia. In particular the distinction between *field dependent* and *field-independent* learners have received a great deal of attention. Whereas field-dependent learners take a more passive approach and approach the material more holistically, field-independent learners tend to be more serialistic in their approach to learning, reusing strategies and tools that they have used before. Concretely, this means that field-independent learners prefer free navigation tools—such as site search, rich interlinking and references to background articles and social filtering—, but that field-dependent learners need more structured material, as not to get 'lost' [31]. Other frequently addressed learning styles include example-oriented vs. activity-oriented learners and verbalizers vs. visualizers [30].

It may be clear that these various kinds of learning styles, as well as individual preferences including background knowledge and situational circumstances can be addressed by a variety of instructional strategies, which eventually result in one or more adaptations. However, there is currently only partial knowledge on which adaptations are useful for which learning style. One could, for instance, say that for field-dependent users a fixed path to follow is preferable to a highly interlinked learning resource. However, should the author of a learning resource have coupled the fixed-path condition with more examples and less hands-on material, this might not be the best choice for the field-dependent learner who is more activity-oriented.

Classical systems, such as AHA!, generate personalized paths that are based on the learner model as well as explicitly indicated preferences. However, if the system has to deal with conflicting preferences, treating these preferences as hard constraints might lead to situations in which no optimal path can be found. A preference-based model would allow for selecting the most appropriate candidate paths among the available ones, which caters as many of the preferences as can possibly be catered.

Preferences provide a framework to model all the specific properties, wishes, interests, and needs of a learner. They allow for a trade-off between several dimensions of a learning style's characteristics which would solve the problem of the fixed-path condition. However, the motivation of a learner highly depends on the fact that she is provided with material suiting her needs. As we have shown, representing these needs via preferences rather than via fixed single-value hard constraints, allows any educational system to provide material which may not be the overall best choice for all users but which is at least the best selection (among the available material) for the learner.

In the following section we will bring together the more informal needs and wishes of Bob presented in the scenario in Section II and the formal model of preferences provided in Section III.

A. Motivation Scenario revisited

In this section, we formally specify Bob's hard and soft constraints from our scenario in Section II. For each attribute that Bob provides a preference upon, an attribute level preference relation is imposed. Some preference relations can be expressed over a single attribute (such as Bob's preferences concerning the weekday of the course) and are therefore modeled as single-attribute preferences. Bob's preference relation about the venue

of the course depends on two attributes: the direction (north or south) and the distance from his home. Therefore, this conditional preference is modeled as a multi-attribute preference. Accordingly, we can formally define Bob's preferences. For example, the preference relation over the attribute *weekday* can be represented as:

$$\succ_{weekday} = \{(Tuesday, Monday), (Wednesday, Monday), (Thursday, Monday), \ldots\}$$

And his multi-attribute preference over the venue can be defined as follows:

$$\succ_{venue} = \{ (direction = south \wedge dist. = 10km, direction = south \wedge dist. < 10km), \ldots, (direction = north \wedge dist. = 10km, direction = north \wedge dist. < 10km)\}$$

In a similar way we can define the other attribute level preferences $\succ_{type\_of\_learning}$, $\succ_{homework}$, $\succ_{price}$, $\succ_{certificate}$, $\succ_{exam}$, $\succ_{duration}$, and $\succ_{start}$.

See Figure 1 in the beginning of the paper, in which the Hasse diagrams for the partial orders representing Bob's preference relations are shown. In order to select Bob's preferred courses out of all the available ones, we need an object level preference that is a composition of all these attribute level preferences.

From Section III we recall that there are two ways of combining attribute level preferences to a single object level preference: lexicographic composition and Pareto composition. The former applies to preference relations that are prioritized and the latter to preferences that are considered equally important.

Bob considers his preference for a course with no exam more important than the preference concerning a certificate. In any case, he prefers a course with no exam. Only if two courses bear the same attribute value in terms of exam, he would prefer the one with a certificate to a course without. Therefore, we first build up a lexicographic composition of $\succ_{exam}$ and $\succ_{certificate}$. We denote this composed preference relation as $\succ_{L_{ex,cert}}$.

As Bob considers his preferences on the remaining attributes as equally important, we build an object level preference by Pareto composing the already composed preference $\succ_{L_{ex,cert}}$ with $\succ_{type\_of\_learning}$, $\succ_{homework}$, $\succ_{cycle}$, $\succ_{price}$, $\succ_{duration}$, $\succ_{weekday}$, and $\succ_{start}$. These single preferences build a Pareto-composed preference relation $\succ_{Bob}$. Given two courses C_1 and C_2, $C_1 \succ_{Bob} C_2$ holds if all attributes of C_1 are equal or better than C_2 according to the attribute's preference relations and if at least one attribute C_1 is better than (and not equal to) C_2 (see definition 5 of the Pareto Composition).

Considering the relation $\succ_{Bob}$, the optimal course would be the one that fulfills all the top values of Bob's preferences, since all others would be dominated by this relation. And obviously, he would be really happy with a regular 3 month course happening once a week on Tuesday, Wednesday, or Thursday without an exam but with a certificate and all the other desired features. Unfortunately, in most of the cases, this course does not exist. However, the semantics of the two composition paradigms cater the desired soft constraint behavior: they provide the courses with an optimal trade-off between the desired and the actual features. This selection is provided by the winnow operator ω_{Bob} defined for $\succ_{Bob}$ and applied to the set O containing all courses that are possibly available to Bob.

Course	Weekday	Price	Distance	Direction
A	Tuesday	44 Euro	2 km	south
B	Monday	44 Euro	2 km	south
C	Wednesday	72 Euro	2 km	south
D	Wednesday	no cost	10 km	north
E	Wednesday	32 Euro	10 km	north

Fig. 2. Some available courses for Bob. B and E are suboptimal: B for weekday, and E for cost. A, C, and D are non-dominated.

We will now show on the basis of the dataset depicted in Figure 2 that the Pareto composition $\succ_{Bob}$ provides exactly the intended best match result: the courses in the skyline, or, more precisely, the courses that are not dominated by any other course. For the sake of simplicity, we omit some of the dimensions from the scenario. As delivered by the winnow operator ω_{Bob}, a course C is considered a best match according to Bob preferences, if there is no other course C' such that $C' \succ_{Bob} C$: there is no other course that dominates C. Given this, we can conclude that course B in Figure 2 is not preferred and therefore irrelevant, since it is dominated by A: A is equal to B according to the dimensions *price*, *distance*, and *direction*. However, A is better than B according to $\succ_{weekday}$ (Bob prefers a course on Tuesday to a course on Monday), which lets A dominate B. Therefore, Bob will not be interested in B, as A is a better alternative. Let us have a look at A and C: A is better than C concerning $\succ_{weekday}$, but in contrast C $\succ_{venue}$ A holds. Given the Pareto composition of $\succ_{weekday}$ and $\succ_{venue}$, A and C are not comparable, as none of them dominates one another. Hence, Bob is probably interested in receiving both as answers, since they are orthogonal alternatives. For attending course D, Bob has to ride to the north of the city, which he really dislikes. On the other hand, D is for free, so in exchange for accepting to ride to the north he will save money. $\succ_{Bob}$ ensures that this alternative will be included into the result set as well, since it is not dominated (even though it is Bob's last option in terms of $\succ_{venue}$).

From the courses depicted in Figure 2, the preference-based search with Bob's constraints, as described in the scenario, presents the courses A, C, and D. The search prunes the courses B and E. B is dominated by A, because on Monday Bob prefers not to reschedule his jogging appointment with his friend, and A is equally good as B in all other dimensions. E is dominated by D, because it is more expensive and not better in any other dimension.

In this section, we elaborated how preferences and their composition are applied to rule out sub-optimal, non-preferred objects. For a learner, this information filtering is of particular importance, as it guides a way through the growing space of possibilities of where and what to learn. In the next section we will show our implementation, in which we applied the preference principle to support students with the selection of courses at a university.

V. Implementation

To show that preference-based search is a promising approach for managing huge data sets of learning resources, we implemented a Web Service for preference-based queries over the whole database for lectures held at the University of Hannover, Germany. The data set comprises about 10,000 lectures each with about 10 attributes. This yields an RDF graph of over

100,000 triples. In order to realize the preference-enhanced search facilities, we implemented a Web Service called *Personal Preference Search Service (PPSS)*, which is integrated into the Personal Reader Framework [32]. We further provide a prototypical user interface[1] to this service that allows students to specify their preferences in order to find a manageable set of learning resources. In this section, we will first provide some insights in a preference-enabled query language for RDF data. Then we will give a short description of the Personal Reader framework and describe the architecture of our service and its integration into the framework. Finally, we provide an experiment that shows the percentage of objects ruled out for given queries.

A. A Preference-enabled Query Language

Querying with preferences in the context of the Semantic Web is a relatively new field. In [28], we made a first contribution by establishing an extension for the RDF query language SPARQL empowered with an implementation based on the ARQ SPARQL Processor [33], which is part of the Jena Semantic Web Framework [34].

To specify soft constraints in the form of preferences, the SPARQL language has been extended by the `PREFERRING`-construct. Figure 3 shows the query for our scenario (for the sake of readability, we left out some of the preference dimensions). Beyond the initial hard constraint construct `FILTER` (see line 11 in Figure 3) it is now possible to define soft constraints in a SPARQL query. The extension of SPARQL comprises two atomic preference expressions and two facilities for combining preference dimensions. For atomic preferences, the following expression types are offered:

- *Boolean preferences* (line 14, 22, 24 and 28 in Figure 3) are specified by a Boolean condition. Results that satisfy this condition are preferred over results that do not satisfy it.
- *Scoring preferences* are specified by the term `HIGHEST` (resp. `LOWEST`), followed by a numeric expression. Results for which this expression leads to a higher value are preferred over results with a lower value (and vice versa). Two types of scoring preferences are provided: either they are defined over a SPARQL domain with a total ordering (line 26 and 30) or they are defined over a partial order (given by the user in form of a set of pairs) (line 16 to 20). As the definition of a partial order over many values may be cumbersome, a placeholder, denoted by an asterisk, can be used to represent any other possible value. As an example, see line 16, in which instead of defining all the equally preferred optimal days of the week, an asterisk is introduced. This feature is particularly helpful if an attribute has a lot of possible values: omitting some values in the definition of a partial order yields an indifference for these omitted values with respect to any other value of that attribute.

These atomic preference expressions can be composed of two types of multidimensional composition (c.f. Section III):

- A *Pareto composed preference* consists of two preference expressions connected by an `AND`. Both expressions are evaluated independently. An object is preferred if it is better in one of both preferences, and at least equally good in the second one.

[1]available at http://semweb.kbs.uni-hannover.de:8081/PreferenceQueryGUI

```
 1 SELECT ?lecture
 2 WHERE {
 3     ?x j.0:name ?lecture.
 4     ?x j.0:learning_type ?type.
 5     ?x j.0:weekday ?weekday.
 6     ?x j.0:exam ?exam.
 7     ?x j.0:certificate ?certificate.
 8     ?x j.0:begin ?begin.
 9     ?x j.0:homework ?homework.
10     ?x j.0:price ?price.
11     FILTER (?price <= 100).
12 }
13 PREFERRING
14     ?type ='face_to_face'
15 AND
16    HIGHEST ?weekday ((*,'Monday'),
17                      ('Monday','Sunday'),
18                      ('Monday','Saturday'),
19                      ('Sunday','Friday'),
20                      ('Saturday','Friday'))
21 AND
22     ?exam = 'no'
23     CASCADE
24         ?certificate ='yes'
25 AND
26     LOWEST ?begin
27 AND
28     ?homework = 'yes'
29 AND
30     LOWEST ?price
```

Fig. 3. Preference-extended SPARQL Query for Bob's desired course.

- In a *cascading (lexicographic) preference*, two preference expressions are connected by a `CASCADE` (line 22 to 24): the first preference is evaluated first. Only for objects that are equally good with respect to the first preference, the second preference is considered.

B. The Personal Reader Framework

The Personal Preference Search Service has been realized as a service part of the *Personal Reader Framework* [32]. The Personal Reader Framework allows for the development of Web content readers that provide adaptation or personalization functionalities. The basic idea of the framework is that personalization functionality is encapsulated into Semantic Web Services, so-called *Personalization Services*, which deliver personalized content based on user profile information. Personalization Services usually focus on a certain domain and task. As an example, the *MyEar music recommender service* [35] delivers personalized podcasting feeds and *MyNews* deals with personalization of news feeds. User profile information, which is evaluated in order to adapt the content to the user, is shared between the different services via a centralized user modeling service (see Figure 4). In this way, users benefit from sharing their user profile among different personalized services, as the cold start problem is reduced and the need for defining their preference at different places is remedied: whenever users switch between different Personal Reader applications, the applications—or, more precisely, the Personalization Services—can utilize user profile information that is gained in other applications. For example, information about the musical taste of a user gathered by MyEar can be exploited by MyNews to filter news articles about music topics. A Personal Reader application is realized by syndicating content that is provided by (possibly multiple) Personalization

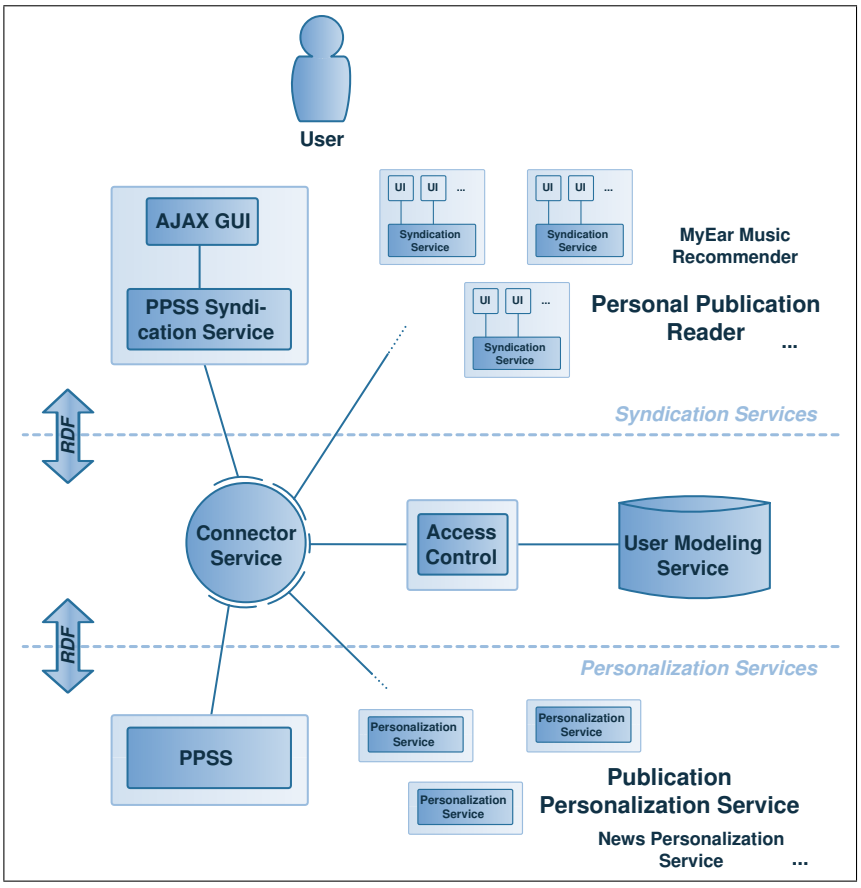

Fig. 4. The Architecture of the Personal Reader Framework comprising the Personal Preference Search Service (PPSS).

Services. The syndication of content and the implementation of further application logic is done in so-called *Syndication Services*, which may be equipped with different user interfaces, as depicted in Figure 4. Orchestration of different services is supported by a *Connector Service*, which allows for dynamic discovery and integration of adequate Personalization Services. Using RDF as data model and format for all communication between the different services, eases interaction between such services, which may not know each other in advance. For users, the service-oriented architecture of Personal Reader applications together with the shared user profile has the advantage that applications—although being possibly composed at runtime—still are able to provide adaptation and personalization functionality.

Given this setting, on the one hand the our Personal Preference Search Service (PPSS) is able to benefit from the shared user model, while on the other hand other services of the Personal Reader Framework will benefit from the functionality of the PPSS for their purposes. For example, the *Personalization Service for Curriculum Planning* [36] and the *MyEar Music Recommender* [35] can utilize the PPSS to offer an improved search for adequate courses and music files, respectively. The *Personal Publication Reader* [37], which allows users to browse publications within an embedded context, would be able to provide suggestions on publications that suit the user's preferences by integrating the PPSS.

C. The Personal Preference Search Service

As shown in Figure 4, the Personal Preference Search Service (PPSS) acts as a personalization service in the Personal Reader Architecture. It comprises an AJAX-based GUI (see Figure 5) and the PPSS Syndication Service. It also wraps the extended ARQ engine. If a query is submitted, the following steps are performed:

1) An RDF description of the preferences is created out of the user's input.
2) This RDF description is passed to the PPSS connected via the connector service. The PPSS creates a SPARQL query out of the RDF description.
3) The extended ARQ engine processes the query.
4) The PPSS generates an RDF description of the result set and passes it to the user interface.
5) The results are formatted and presented to the user.

These functionalities of the PPSS are separated. This fact, together with the architecture of the Personal Reader and the flexibility of the user interface, enables the system to query any RDF-based dataset of learning resources with arbitrary attributes of the objects in the dataset.

The prototypical user interface (shown in Figure 5) allows the user to specify her preferences for each attribute. By clicking on the *more*-button, additional alternative values can be provided. For each attribute, the asterisk placeholder can be introduced by selecting, for example, the entry "any other time". This user interface offers the definition of total-order, Pareto composed, single-attribute preferences (cf. Section III). Due to the com-

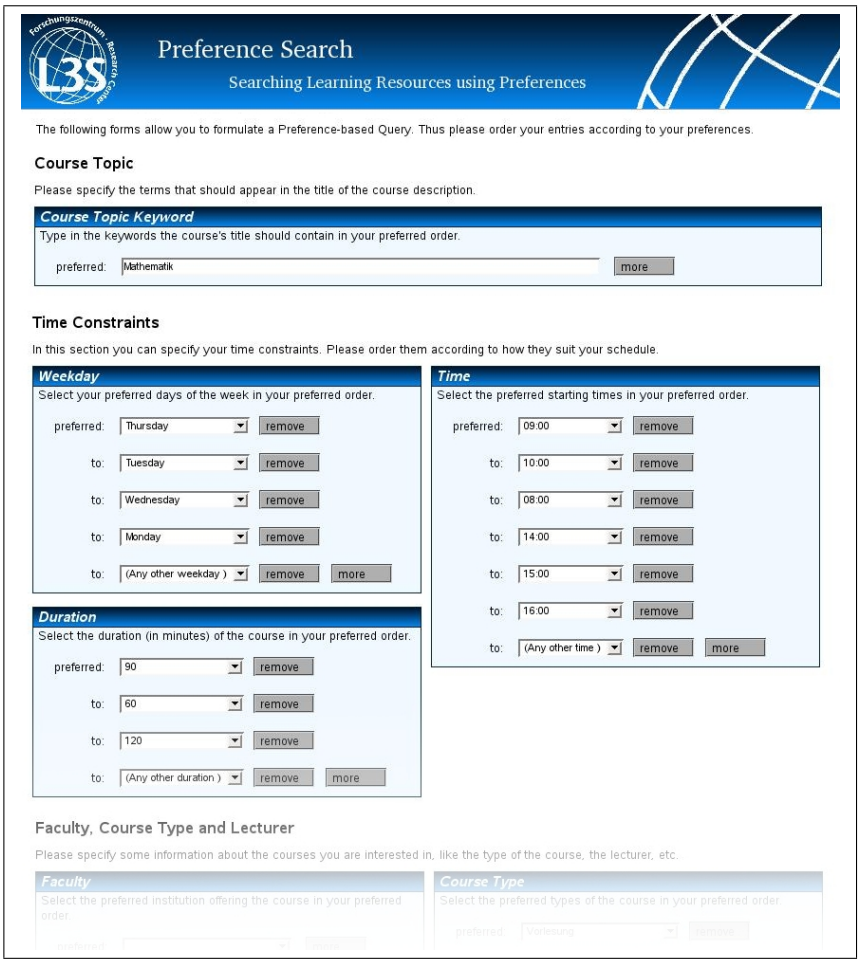

Fig. 5. The prototypical user interface of the Personal Preference Search Service allowing for the specification of preference orders.

plexity of an user interface that allows for definition of partial orders, prioritized and dependent dimensions, we currently do not allow for these kinds of preference structure, although our implementation is able to handle them.

A definition of a preference order with only one element (such as the *course topic keyword* in Figure 5) is considered a hard constraint and is added to the SPARQL query as part of the `FILTER`-construct. This allows the user not only to specify soft constraints in form of a preference order but also to specify hard constraints, for example that the course should have a topic containing “Mathematics”.

D. Experiment

We have performed a number of experiments with the lecture database of the learning management system of the University of Hannover. This system currently comprises 9829 lectures. In the following example, we provide a preference-enabled query and show how preference-enabled queries optimize the result set and provides the desired learning resources, without pruning relevant results or returning non-relevant objects:

> *Return courses about mathematics. I am interested in readings rather than in tutorials or seminars. If possible, I would like to attend a 90 minutes lecture. 60 minutes are also fine, but 120 minutes lectures are too long. I like to have the lecture in the morning rather than in the afternoon. Due to the lunch break, noon is not possible for me. I don’t want to have a lecture on Friday. Thursday would be my first choice, followed by Tuesday. Wednesday would also be acceptable, and would actually be better than Monday, as often I am still at my parents until Monday afternoon.*

Figure 5 shows how this query is specified in the user interface. The SPARQL query created to search for the desired course is shown in Figure 6. For each attribute, a preference relation is imposed (line 22, 26, 32, and 41) and all the relations are Pareto composed via the `AND`-keyword. Issuing this query to the database of the University of Hannover yields the 4 results as shown in the table in Figure 7. Obviously, none of the returned courses matches all of the desired attributes. The first lecture is held too late during the day, takes place on Tuesday, and it is not a reading either; the second lecture is too long, and so on. Mind that the order in the table does not correspond to a ranking: all six results are equally relevant. However, concerning all the 64 courses about Mathematics that are contained in the database, these 4 results are optimal: the remaining 58 courses are worse in terms of at least one preference relation.

In Section III-C we already pointed out that it is difficult to express soft constraints in a best match search approach that does not allow for preferences. In this section, we show by means of the given search request that the two alternative approaches,

Course	Start time	Type	Weekday	Duration	Faculty
Mathematics Exercises	10:00	Tutorial	Tuesday	120	Applied Math.
Mathematics (Economics)	09:00	Reading	Thursday	120	Algebra
Mathematics (Geography)	08:00	Reading	Thursday	90	Analysis
Mathematics (Engineers)	10:00	Reading	Tuesday	60	Applied Math.

Fig. 7. The 4 optimal courses at University of Hannover retrieved via a preference-enabled query to the course repository.

Course	Start time	Type	Weekday	Duration	Faculty
Math. in Physics	14:00	Seminar	Wednesday	120	Theor. Physics
Math. in Assurances	08:00	Reading	Monday	90	Mathematics
⋮	⋮	⋮	⋮	⋮	⋮
Mathematics (Engineers)	10:00	Reading	Thursday	120	Algebra
Math. for Beginners	08:00	Tutorial	Wednesday	120	Algebra

Fig. 8. The 25 courses at the University of Hannover that match the disjunctive query. The 4 optimal courses included.

conjunctive and disjunctive queries, do not provide a satisfactory search result.

Conjunctive. For this approximation to preferences, the user would need to conjunctively connect all preferred attributes and to execute several queries by going step by step down in the preference order, by applying a trial and error strategy. This way of querying is cumbersome and, moreover, returns too few and—in most of the cases—no results. After some queries with no results the user gets frustrated. Even if some results are returned, the user needs to create queries with all different alternatives in order to be able to select the best match. In our current example, the conjunctive query yields an empty result, as none of the courses in Figure 7 has all the most preferred properties.

Disjunctive. The second approach is to disjunctively put all the possible desired outcomes into a single query. This query usually returns a large result set that does contain the desired optimal courses, but also a lot of non-optimal results that are dominated by better ones. In our example, this query yields 25 courses (see an excerpt of the results in Figure 8), including courses with suboptimal attribute combinations. For instance, the lecture "Mathematics (Engineers)", offered by the Faculty for Algebra, is suitable but obviously worse than "Mathematics (Geography)", offered by the Faculty for Analysis (third item in Figure 7). The latter dominates the former, as it is a 90 minutes lecture, which is preferred to a 120 minutes one. For this reason, it is not worthwhile to include the longer lecture into the result set. By following this procedure, filtering out non-optimal results, the PPSS reduces the number of results from 25 to 4.

Both the conjunctive and the disjunctive approaches are not satisfactory. The former comes up with no results and the latter puts the burden on the user with many non relevant courses. Therefore, our preference approach solves this problem by returning to the user not too many results (leaving optimal ones out) and not too few (including sub-optimal ones), but returning exactly the ones that are optimal according to the user's preferences given in the query.

VI. Related Work

To the best of our knowledge, using preference handling to support the learning process has not been considered thus far. Personalization that is based on single-value preferences (preferences not providing an order but a single value) have been subject of manifold research. For some examples we refer the reader to [3], [5], [38]. The work described in this paper focuses explicitly on how learner preferences look like and exploits the structure of the user-given alternatives.

Some research has been carried out in area of quantitative preferences. In [39] a framework for expressing and combining quantitative preferences is described. Two other quantitative approaches are already mentioned in Section III-C, such as query relaxation and top-k. The main drawback of any quantitative solution is that the user is forced to either define a utility function (or a relaxation function) or to use a predefined one. Both alternatives do not sufficiently fit a learning scenario in which users are not willing to specify detailed numeric functions. Furthermore, aslearners have different preferences, learning styles, and constraints, predefine solutions do not fit as well.

Preference-based search in the domain of digital libraries is provided in [40]. In this work, preferences are defined for one single dimension: over keywords of the desired object. Due to this fact, the preferences are used for sorting the results and cannot be exploited to filter irrelevant objects. In [41], different approaches for catalog search are compared. It was found that the preference-based alternative is the most promising. However, the opportunities for defining preferences in the search form of the compared preference approach, as presented in [41], are limited to the identification and prioritization of dimensions, but do not allow for preferences between the values of the dimensions. This is crucial for complex domains, such as learning resources where most of the dimensions are discrete.

VII. Conclusions and Further Work

Learner models as well as search capabilities in existing educational systems typically allow for hard constraints that learning material should fulfill. However, in many cases, users do not just think in terms of hard constraints, but rather have soft constraints in their mind, such as "Monday is better but Tuesday would be fine as well". Preferences allow users to specify these wishes in a way that can be processed by engines in order to return only the best matches based on such wishes: those results that *dominate*

```
 1 PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
 2 PREFIX rdf:<http://www.w3.../22-rdf-syntax-ns#>
 3 PREFIX j.0:<http://www.l3s.de/studip#>
 4 PREFIX fn:<java:...jena.query.function.library.>
 5
 6 SELECT ?name
 7        ?starttime
 8        ?type1
 9        ?weekday
10        ?duration
11        ?faculty
12 WHERE {
13     ?x j.0:name ?name.
14     ?x j.0:type1 ?type1.
15     ?x j.0:weekday ?weekday.
16     ?x j.0:start_time ?starttime.
17     ?x j.0:duration ?duration.
18     ?x j.0:faculty ?faculty.
19     FILTER (fn:contains(?name,"Mathematics")).
20 }
21 PREFERRING
22   HIGHEST ?type1
23           (('Vorlesung','Uebung')
24           ('Uebung','Seminar'))
25 AND
26   HIGHEST ?weekday
27           (('Thursday','Tuesday')
28           ('Tuesday','Wednesday')
29           ('Wednesday','Monday'),
30           ('Monday',*))
31 AND
32   HIGHEST  ?starttime
33           (('09:00','10:00')
34           ('10:00','08:00')
35           ('08:00','14:00')
36           ('14:00','15:00')
37           ('15:00','16:00')
39           ('16:00',*))
40 AND
41   HIGHEST ?duration
42           (('90','60')
43           ('60','120')
44           ('120',*))
```

Fig. 6. Preference-extended SPARQL query for preferred courses at the University of Hannover.

the rest of potentially relevant ones. These preference-enabled queries are easy to process and provide intuitive semantics of what is meant with a preference order.

In this paper, we described how such preferences and preference-based queries can be used for selecting for suitable learning resources. We showed that our approach is more expressive than existing approaches, which are either restricted to a single value preference or require a quantitative representation of preferences. By considering the learner's preferences, the learning objects that are selected by the system are optimal for the learning process. Moreover, suboptimal learning objects are not shown to the user, which saves the user from the burden of selecting from a huge set of possibilities. Our approach allows for both hard and soft constraints. We presented a very broad notion of soft constraints that allows for preferences including prioritized as well as non-prioritized composition, partial order preferences and conditional preferences.

We presented the implementation of our approach as a Web service in the Personal Reader Framework. We exploited the preference extension of a widespread Semantic Web query language in order to query a large dataset of learning objects in a user-friendly way. Further, we demonstrated the value of our approach via an experiment at the University of Hannover's learning management system.

Our future work focuses on the improvement of our current prototype with optimized algorithms that are based on recent results on skylining research. We currently investigate enhancements to our user interface in order to allow more expressive preferences, which are already supported by our implemented engine. The storage and reuse of preferences is another topic of future research: a preference repository [9] may store some of the preferences that tend to be static (such as the preferred language, location, etc.) and reuse them automatically for future search requests. The central user modeling service of the Personal Reader Framework may serve as such a repository. In addition, preferences may also be used for improving existing automatic course generation algorithms (such as [42], [36]) and recommendations (first ideas in [43]). We are currently exploring these directions of research.

ACKNOWLEDGEMENTS

The authors' efforts were (partly) funded by the European Commission in the TENCompetence project (IST-2004-02787) (www.tencompetence.org).

REFERENCES

[1] A. Jameson, *Adaptive interfaces and agents*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 2003, pp. 305–330.

[2] G. Cheetam and G. Chivers, *Professions, Competence and Informal Learning*. Edgard Elgar Publishing Limited, 2005.

[3] P. Brusilovsky, "Adaptive hypermedia," User Modeling and User-Adapted Interaction, Ten Year Anniversary Issue 11 (Alfred Kobsa, ed.), pp 87-110., 2001.

[4] H. Hills, *Individual Preferences in E-Learning*. Brookfield, VT 05036, USA: Gower Publishing Co., 2003.

[5] A. Kobsa, *The Adaptive Web: Methods and Strategies of Web Personalization, Brusilovsky, P., Kobsa, A., Nejdl, W., eds., Lecture Notes in Computer Science, Vol. 4321*. Springer-Verlag, Berlin Heidelberg New York, 2007, ch. Generic User Modelling Systems.

[6] P. Brusilovsky and C. Peylo, "Adaptive and intelligent web-based educational systems," *International Journal of Artificial Intelligence in Education, Special Issue on Adaptive and Intelligent Web-based Educational Systems*, vol. 13, pp. 159–172, 2003.

[7] F. Abel, E. Herder, P. Kärger, D. Olmedilla, and W. Siberski, "Exploiting preference queries for searching learning resources," in *EC-TEL*, 2007, pp. 143–157.

[8] S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.* Addison-Wesley, 1990, ch. 5.4.2 Hasse Diagrams, pp. 163, 169–170, and 206–208.

[9] S. Holland and W. Kießling, "Situated preferences and preference repositories for personalized database applications," in *ER*, 2004, pp. 511–523.

[10] W. Kießling, "Foundations of preferences in database systems." in *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China, 2002, pp. 311–322.

[11] J. Chomicki, "Preference formulas in relational queries." *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 427–466, 2003.

[12] W.-T. Balke, U. Güntzer, and W. Siberski, "Restricting skyline sizes using weak pareto dominance," *Informatik—Forschung und Entwicklung*, vol. 21, no. 3-4, pp. 165–178, 2007.

[13] C.-Y. Chan, P.-K. Eng, and K.-L. Tan, "Stratified computation of skylines with partially-ordered domains," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2005, pp. 203–214.

[14] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

[15] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, Rome, Italy, 2001.

[16] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries." in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Diego, CA, USA, 2003, pp. 467–478.

[17] A. Motro, "Vague: a user interface to relational databases that permits vague queries," *ACM Trans. Inf. Syst.*, vol. 6, no. 3, pp. 187–214, 1988.

[18] T. Gaasterland, "Cooperative answering through controlled query relaxation," *IEEE Expert*, vol. 12, no. 5, pp. 48–59, 1997.

[19] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica, "Relaxing join and selection queries," in *Proceedings of the 32nd international conference on Very large data bases (VLDB)*. VLDB Endowment, 2006, pp. 199–210.

[20] I. Muslea, "Machine learning for online query relaxation," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. New York, NY, USA: ACM, 2004, pp. 246–255.

[21] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis, "Automated ranking of database query results," in *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.

[22] M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, Eds., *Proceedings of 25th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, 1999.

[23] C. Li, M. A. Soliman, K. C.-C. Chang, and I. F. Ilyas, "RankSQL: Supporting ranking queries in relational database management systems." in *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway, 2005, pp. 1342–1345.

[24] R. Fagin, "Combining fuzzy information from multiple systems," in *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 1996, pp. 216–226.

[25] U. Güntzer, W.-T. Balke, and W. Kießling, "Optimizing multi-feature queries for image databases," in *Proceedings of 26th International Conference on Very Large Data Bases (VLDB)*, 2000, pp. 419–428.

[26] M. Theobald, H. Bast, D. Majumdar, R. Schenkel, and G. Weikum, "TopX: efficient and versatile top-k query processing for semistructured data," *VLDB J.*, vol. 17, no. 1, pp. 81–115, 2008.

[27] W. Kießling and G. Köstler, "Preference sql - design, implementation, experiences." in *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, 2002, pp. 990–1001.

[28] W. Siberski, J. Z. Pan, and U. Thaden, "Querying the semantic web with preferences." in *Proceedings of the 5th International Semantic Web Conference (ISWC)*, Athens, GA, USA, 2006, pp. 612–624.

[29] R. Koper and B. Daniel, "Developing advanced units of learning using ims learning design level b," *International Journal on Advanced Technology for Learning*, 2005.

[30] N. V. Stash, A. I. Cristea, and P. M. D. Bra, "Authoring of learning styles in adaptive hypermedia: problems and solutions," in *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA: ACM, 2004, pp. 114–123.

[31] S. Y. Chen and R. D. Macredie, "Cognitive styles and hypermedia navigation: development of a learning model," *Journal of the American Society for Information Science and Technology*, vol. 53, no. 1, pp. 3–15, 2002.

[32] N. Henze and M. Kriesell, "Personalization functionality for the semantic web: Architectural outline and first sample implementations, semantic web challenge 2005," in *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)*, August 2004.

[33] A. Seaborne, "An open source implementation of sparql," in *WWW 2006 Developers track presentation*, http://www2006.org/programme/item.php?id=d18, 2006.

[34] B. McBride, "Jena: a semantic web toolkit," *Internet Computing, IEEE*, vol. 6, no. 6, pp. 55–59, Nov/Dec 2002.

[35] N. Henze and D. Krause, "Personalized access to web services in the semantic web," in *SWUI 2006 - 3rd International Semantic Web User Interaction Workshop*, Athens, Georgia, USA, Nov 2006.

[36] M. Baldoni, C. Baroglio, I. Brunkhorst, E. Marengo, and V. Patti, "A personalization service for curriculum planning," in *ABIS 2006 - 14th Workshop on Adaptivity and User Modeling in Interactive Systems*, October 2006.

[37] F. Abel, R. Baumgartner, A. Brooks, C. Enzi, G. Gottlob, N. Henze, M. Herzog, M. Kriesell, W. Nejdl, and K. Tomaschewski, "The personal publication reader, semantic web challenge 2005," in *4th International Semantic Web Conference*, November 2005.

[38] P. Dolog, N. Henze, W. Nejdl, and M. Sintek, "Personalization in distributed e-learning environments," in *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA: ACM, 2004, pp. 170–179.

[39] R. Agrawal and E. L. Wimmers, "A framework for expressing and combining preferences," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2000, pp. 297–306.

[40] N. Spyratos and V. Christophides, "Querying with preferences in a digital library," in *Federation over the Web*. Dagstuhl Seminar (N 05182), May 2005, vol. LNAI 3847.

[41] S. Dring, S. Fischer, W. Kießling, and T. Preisinger, "Optimizing the catalog search process for e-procurement platforms," *deec*, vol. 0, pp. 39–48, 2005.

[42] C. Ullrich, "Course generation based on htn planning," in *Proceedings of 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems*, Saarbrcken, Germany, 2005.

[43] B. Satzger, M. Endres, and W. Kießling, *E-Commerce and Web Technologies*. Springer Berlin / Heidelberg, 2006, ch. A Preference-Based Recommender System.

Philipp Kärger is a Ph.D. student at the University of Hannover's Computer Science Department and a research scientist at the L3S Research Center. He received his master degree in computer science from Saarland University, Saarbrücken, Germany.

Daniel Olmedilla is a project leader at L3S Research Center and the University of Hannover since 2005. Before joining L3S as researcher in 2002, he worked as consultant and project manager in IT companies. He received his master degree and Ph.D. in computer science from Universidad Autnoma de Madrid, Spain.

Fabian Abel started his Ph.D. in 2007 at the L3S Research Center. He received his master degree in computer science at the Leibniz University Hannover, Germany. In the context of his Ph.D. he is concerned with Social Media, Semantic Web, user modeling, and personalization techniques in social systems.

Eelco Herder received his Ph.D. in Computer Science at the University of Twente, the Netherlands. A long-term Web usage study, in collaboration with the University of Hamburg, was awarded at the WWW2006 conference. Currently, he works at the L3S Research Center in Hannover, Germany. His main research interests include user modeling, Web personalization, usability, interaction design and user studies.

Wolf Siberski since 2005 project leader at L3S Research Center of University of Hannover. Before joining L3S as researcher in 2001, he worked as software architect and internal consultant in IT companies. He received his PhD in computer science from University of Hannover and his master degree in computer science from University of Hamburg, Germany.