Exploiting Preference Queries for Searching Learning Resources

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

L3S Research Center and Leibniz University of Hannover, Hannover, Germany
{abel,herder,kaerger,olmedilla,siberski}@L3S.de

Abstract. While the growing number of learning resources increases the choice for learners, it also makes it more and more difficult to find suitable courses. Thus, improved search capabilities on learning resource repositories are required. We propose an approach for learning resource search based on preference queries. A preference 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'). Such queries always return the set of optimal items with respect to the given preferences. We show how to exploit this technique for the learning domain, and present the Personal Preference Search Service (PPSS) which offers significantly enhanced search capabilities compared to usual search facilities for learning resources.

1 Introduction

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 adding into the query hard constraints representing the user wishes (e.g., from the user profile), that is, conditions that must be fulfilled. Examples of these hard constraints are "results must be either in English or German and must provide a certification". There are two choices how to incorporate these additional constraints into a given query, both leading to suboptimal answer sets. Either, we use a conjunctive query, i.e., the additional constraints are connected with an 'and'. In this case, the danger is high that we end up with an empty result set because of the query becomes too specific. Or, we add the constraints disjunctively, i.e., all constraints connected with an 'or'. But then, the size of such a result set grows significantly, and will contain many scarcely relevant results.

In order to solve the problem of large number of returned results, ranking mechanisms try to sort the results showing to the user the best matches first, but this notion of relevancy is typically a score computed out of i.e. number of occurrences of a keyword, TF/IDF[1], proximity of keywords, popularity of the resource, etc., elements that do not necessarily represent the user wishes.

[1] term frequency / inverse document frequency

A closer look reveals that in most cases additional constraints are not hard constraints. Typically a user may want to express that she wants "courses preferably in English but if there are not, also in German would suffice and which take place on Mondays better than Tuesday or Fridays". These "preferably" and "better-than" indicate soft constraints in which a user specifies *what she prefers*, that is, her wishes as preferences. These preferences can then be used in order to filter out non-relevant results. For example, if two courses are found, both on Mondays and one is in English and the other one in German, intuitively the latter can be discarded since given the same (or worse) conditions, the user prefers English over German. This way, only optimal results according to 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.

It is important to note that the term *user preferences* has been extensively used in the field of user modeling [1] and adaptive hypermedia [2, 3]. Typically, these user preferences are a set of properties for which learners express interest (and which are added in the queries as hard constraints). By contrast, our method is more expressive since it does not only allow such interests to be modelled but also allows users to indicate which properties they prefer to another by allowing for a *preference order*.

This paper describes how preference-based queries can be used in order to a) increase the expressivity of queries, helping users describing more accurately their wishes and interests and b) retrieve efficiently optimal matches according to the user preferences discarded the rest. The paper is organized as follows: in Section 2 our approach is motivated with a running scenario. Theoretical background about preferences and its use in query processing is provided in Section 3. Section 4 applies this theory to our running example in order to show how preferences may be applied to search of learning resources. Section 5 describes our prototype implementation and presents some experiments. Finally, Section 6 compares our approach with existing initiatives, Section 7 discusses some important issues regarding to user interface and Section 8 concludes the paper.

2 Motivation Scenario

In the following, we picture a scenario to demonstrate how preference-based search supports learners in finding suitable courses. We will use this example throughout the paper to illustrate 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. 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. Bob is not a professional in photography: so he does not insist on gaining a certificate. But should there be a course with a certificate at the same or better conditions (price, etc.), he would prefer to take the one with the certificate. However, he does not want to pass

an exam for gaining the certificate. Bob believes that he will like doing image processing with his computer. Hence, he also wants a course comprising some kind of homework.

Bob would prefer a course offered in the evening on working days, except on Monday; then 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. Bob would like to have the course taking place once a week, in a period of about three months. 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. However, since he just got his new camera he wants the course to start as soon as possible as not to lose any time.

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

With current search interfaces, it is not possible to specify such a complex search request. A platform providing extended search capabilities to take into account all given 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 mother language. It also uses Bob's preferences gained from his past interactions, such as his fondness for meeting people, the location where he lives, his regular meeting on Fridays. By taking all these constraints into account, the system is able to perform a query comprising most of the particularities in Bob's idea of a course. Probably there will be no course matching all 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.

3 Preferences and Preference-based Queries

In order to model the kinds of hard and soft constraints 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, advanced search for suitable courses is needed. Searching is not well supported with a query model where users can only specify hard constraints on course characteristics. To provide more effective search capabilities in such cases, query languages like SQL over relational databases and, recently, SPARQL over RDF graphs have been extended to facilitate preference-based retrieval algorithms [4, 5].

These approaches assign a degree of match with respect to user-specified soft constraints to each object and then aggregate this degree to compute the set of *best matching answers*. Under the common exact match paradigm too specific query predicates often lead to empty result sets, while too unspecific hard constraints may yield huge numbers of results. The notion of best matches fits much better to typical user's search requests, because it automatically adapts query specificity to the available objects. Our proposed solution to achieve best matches is exploiting preference orders for querying.

The notion of preference-based querying in the context of databases has been formalized independently by Kießling [6] and Chomicki [7]. To describe user's preferences in a way exploitable for querying, we rely on the preference query formalization proposed by Chomicki in [7]. In this extension to relational algebra, preferences are expressed as binary relations over a set of objects R.

Definition 1. *Let $A = \{a_1, \ldots, a_n\}$ be the set of available attributes of the elements in R, and U_i, $1 \leq i \leq n$ the respective set of possible values of 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 a preference relation over R.*

For combining several preference relations, *uni- or multidimensional composition* of the preference relations is needed. Unidimensional composition is applicable if the relations are defined over the same attribute subset. If the relations are imposed over different sets of attributes, we need a multidimensional composition imposing 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 [7], two multidimensional compositions are common:

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

In most of the cases, imposing a priority to the dimensions is difficult for a user. For example, in our given scenario it is difficult for the user to decide what is more important, the schedule of a photography course or its location. It is best to consider them as equally important, and then let the user do the final choice given on the found courses. Therefore, we use pareto composition as default to combine preference relations.

Pareto composition yields a new preference relation following the 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 2. *Given the preference relations* $\succ_1,\ldots,\succ_n$ *over the sets of attributes* $A_1,\ldots,A_n$*, the pareto composition* $\succ_P$ *of* $\succ_1 \ldots \succ_n$ *is defined as:*
$x \succ_P y \Leftrightarrow (\forall i : x \succ_i y \vee x =_i y) \wedge \exists j : x \succ_j y.$

For instance, in our scenario a low-cost course X dominates an expensive course Y only iff in terms of all other preference relations (e.g., 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* [8–10]. 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 which 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*.

Pareto composition can be combined with lexicographic composition in the following way: on some of the pareto-combined dimensions a hierarchy can be imposed such that only if two objects are equal in terms of the first preference relation, the second one will be considered. We call the resulting preference expression a *cascaded preference*.

In the next section, we show how these preference expressions are applied to effectively search for learning resources.

4 Preferences on Metadata of Learning Resources

With the preferences at hand, we are now able to specify the constraints in the scenario in Section 2 in a formal way. For each preference Bob provides, a preference relation is imposed upon the corresponding attribute. Preference relations can be expressed over a single attribute (such as Bob's preferences concerning the weekday of the course) or over several attributes (such as Bob's preference relation about the venue of the course: it depends on two attributes, the location (north or south) and the distance from his home). According to that, we can formally define Bobs 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:

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

In a similar way we can define $\succ_{type\_of\_learning}$, $\succ_{is\_homework}$, $\succ_{cycle}$, $\succ_{price}$, etc.

Preference relations build partial orders on the values of the attributes they are imposed on. In some cases, a preference correspond to a total order (such

as Bob's preference on price), but usually a total order is too restrictive and do not allow for indifferences (such as Bob's indifference concerning Tuesday, Wednesday, or Thursday). Figure 1 shows the partial orders representing Bob's preference relations.

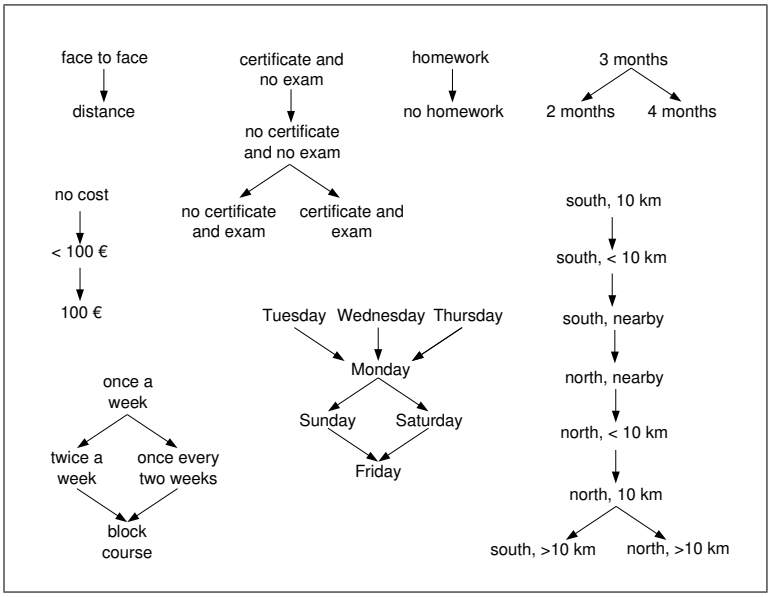

Fig. 1. Partially Ordered Sets representing the preference relations according to Bob's preferences

These single preferences build up 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 according to the attributes preference relations to C_2 and in at least one attribute C_1 is better than (and not equal to) C_2.

Considering the relation $\succ_{Bob}$, the optimal course would be the one fulfilling all the 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 desired features. Unfortunately, in most of the cases this course does not exist, and it is a challenge to find out which of the courses available provide an optimal trade-off between desired and existing features. We will now show by the hand of the dataset depicted in Figure 2 that the pareto composition $\succ_{Bob}$ provides exactly the intended best match result, i.e., the courses in the skyline, or, more precisely, the courses which are not dominated by any other course.

As stated above, a course C is considered a best match according to Bob preferences if there is no course C' such that $C' \succ_{Bob} C$, i.e. there is no other

course that dominates C. Given this, we can conclude, that course B in Figure 2 is irrelevant since it is dominated by A: A is equal to B according to the dimensions price, distance, and location; but 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. So Bob will not be interested in B since A provides a better alternative. Let us have a look at A and C: A is better than C concerning $\succ_{weekday}$ but otherwise C $\succ_{venue}$ A holds. Given the pareto composition of these preferences, A and C are not comparable since none of them dominates the other. Hence, Bob is probably interested in both since they are orthogonal alternatives.

For attending course D, Bob has to ride to the north of the city what he really dislikes. But D is for free, so he may accept to drive to the north because he saves money. $\succ_{Bob}$ ensures that also this alternative will be included into the result set since it is not dominated (although it is the last option in terms of $\succ_{venue}$).

Course	**Weekday**	**Price**	**Distance**	**Location**
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

From the courses depicted in Figure 2, the preference based search with the query described in the scenario presents the courses A, C, and D. It prunes the courses B and E. B is dominated by A because on Monday Bob prefers to attend the jogging with his friend, and A is equally good in all other dimensions. E is dominated by D, because it is more expensive and not better in any other dimensions.

5 Preference Search Prototype

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 currently held at the University of Hannover. 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 Service called *Personal Preference Search Service (PPSS)*[2] integrated as a Web Service into the Personal Reader Framework [11].

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

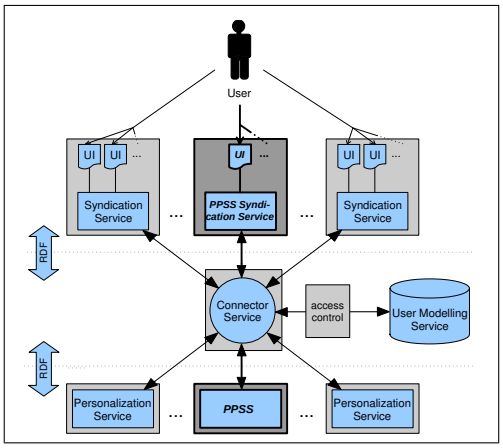

Fig. 3. The PPSS integrated into the Personal Reader Framework

5.1 The Personal Reader Framework

The *Personal Reader Framework* [11] enables developers to create web service based Semantic Web applications. Such applications are composed of different kinds of services as illustrated in Figure 3: *Syndication Services* implement the application logic and utilize RDF data that is provided by the *Personalization Services* which themselves are called via a *Connector Service* by specifying a goal. Based on this goal the Connector Service detects the best suiting Personalization Services. Both Syndication and Personalization Services are able to access user data managed by a central *User Modelling Service*.

As shown in Figure 3 the Personal Preference Search Service is integrated into the Personal Reader Architecture as a Personalized Service including the following components:

1. User Interface which enables the user to formulate preference queries and visualizes the results of a search
2. Syndication Service which preprocesses the preferences, initiates the search, and processes the results
3. Personalization Service called *Personal Preference Search Service* offering the core search engine for Learning Resources

Given this setting, the PPSS is able to benefit from the shared user model while other services of the Personal Reader Framework will benefit from the functionality of the PPSS for their part. For example the *Personalization Service for Curriculum Planning* [12] and the *MyEar Music Recommender* [13] can utilize the PPSS to offer an improved search for adequate courses and music files respectively. The *Personal Publication Reader* [14], 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. Such integration issues are current research topics.

5.2 The Personal Preference Search Service

Querying with preferences in the context of the Semantic Web is a relatively new field. In [5], 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 [15] part of the Jena Framework.

To specify preferences, the SPARQL language has been extended by the `PREFERRING`-construct, two atomic preference expressions, and two facilities for combining preference dimensions. For atomic preferences, the following expression types are offered:

- *Boolean preferences* are specified by a boolean condition. Results satisfying that condition are preferred over results which do not satisfy it.
- *Scoring preferences* are specified by a `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 (resp. vice versa).

These atomic preference expressions can be composed to two types of multidimensional preferences (c.f. Section 3):

- 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.
- In a *cascading preference*, two preference expressions are connected by a `CASCADE`; the first preference is evaluated first; only for objects which are equally good with respect to the first preference, the second preference is considered.

The PPSS operates on top of the extended ARQ engine. If the PPSS receives an RDF description of preference definitions, it creates a SPARQL query, passes it to the engine, collects the result set, and returns an RDF description of that result set. The separation of functionalities in the PPSS (i.e., the separation of SPARQL query generation, the query processing, and the assembly of the result set) as well as the architecture of the Personal Reader enables the system to query each RDF-based data set of learning resources.

The current user interface (shown in Figure 4) allows the user to define his preferences. Currently we provide total ordered single-attribute preferences (c.f. Section 3). Due to the complexity of a user interface allowing the definition of partial orders and dependend dimensions, we currently do not allow for these kinds of preference structure, although our implementation is able to handle them. Considerations concerning the user interface are discussed in Section 7.

5.3 Experiments

We have performed experiments with the lecture database of the learning management system of the University of Hannover. That system currently comprises

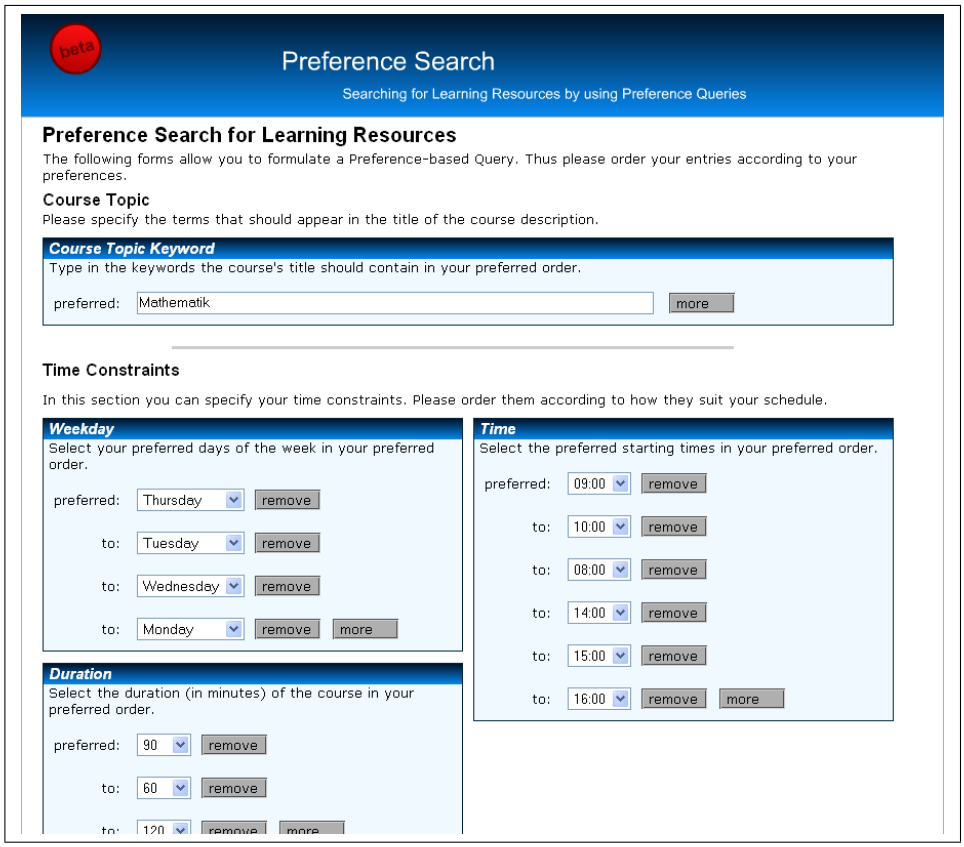

Fig. 4. The user interface of the Personal Preference Search Service

9829 lectures. As an example, given the following preference query, we show how preference 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 and seminars. If possible, I would like to attend a 90 minutes lecture. 60 minutes are also fine, but 120 minutes 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, then Tuesday. Wednesday would also be acceptable and is preferred to Monday, where I am usually still at my parents.*

The SPARQL query according to this desired course is shown in Figure 5. Its corresponding result set is shown in the table in Figure 6. Obviously, none of the returned courses matches all the desired attributes: the first lecture is held too late, on Tuesday, and it is not a reading; the second is too long, and

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX j.0: <http://www.l3s.de/studip#>
PREFIX fn: <java:com.hp.hpl.jena.query.function.library.>

SELECT ?name ?starttime ?type1 ?weekday ?duration ?faculty
WHERE {
    ?x j.0:name ?name.
    ?x j.0:type1 ?type1.
    ?x j.0:weekday ?weekday.
    ?x j.0:start_time ?starttime.
    ?x j.0:duration ?duration.
    ?x j.0:faculty ?faculty.
    FILTER (fn:contains(?name,"Mathematik")).
}
PREFERRING
    ?type1 ='Vorlesung'
        CASCADE ?type1 ='Uebung'
        CASCADE ?type1 ='Seminar'
AND
    ?weekday='Thursday'
        CASCADE ?weekday ='Tuesday'
        CASCADE ?weekday ='Wednesday'
        CASCADE ?weekday ='Monday'
AND
    ?starttime='09:00'
        CASCADE ?starttime ='10:00'
        CASCADE ?starttime ='08:00'
        CASCADE ?starttime ='14:00'
        CASCADE ?starttime ='15:00'
        CASCADE ?starttime ='16:00'
AND
    ?duration ='90'
        CASCADE ?duration ='60'
        CASCADE ?duration ='120'
```

Fig. 5. Preference-extended SPARQL query

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, these 6 results are optimal: the remaining 58 courses are worse in terms of at least one preference relation.

Without the possibility to define preference orders, there are two alternative approaches in classic, i.e., best match search interfaces: The first is to conjunctively connect all preferred attributes and do several queries by going step by step down according to the preference order. This manner of querying returns to few and - in most of the cases - no results. After some queries with no results the user gets frustrated, and 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 since non of the courses in Figure 6 bear each of the most preferred properties.

The second approach is to disjunctively put all the possible desired outcomes into a single query. This query usually returns a huge result set containing the desired optimal courses but also a lot of non optimal results which are dominated by better ones. In our example, this querying yields to 25 courses (see an excerpt of the results in Figure 5.3), including courses with suboptimal attribute

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.
Mathematics (Chemistry)	09:00	Reading	Thursday	120	Chemistry
Mathematics and Physics	10:00	Reading	Tuesday	90	Chemistry

Fig. 6. Optimal courses at University Hannover

combinations. For instance the lecture "Mathematics (Engineers)" held at the Faculty for Algebra is suitable but obviously worse than "Mathematics (Geography)" held by the Faculty for Analysis (third item in Fig. 6) because the latter dominates the former since it is a 90 minutes lecture which is preferred to a 120 minutes one. For that reason, the longer lecture is not worth to be included into the result set. By doing that the PPSS reduces the number of results from 25 to 6.

Both, the conjunctive as well as the disjunctive approach are not satisfactory: the first one comes up with no results whereas the second alternative bothers the user with many non relevant courses.

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. 7. Courses at University Hannover matching the disjunctive query

6 Related Work

Beyond the theoretical achievements in preferences (as summarized in [7, 6]), several applications of this theory have been realized to support search. In [16], preference-enhanced search engines for e-shopping are presented. In any of these systems, preferences are hard-wired into the search mask and cannot be easily specified or refined by the user. Preference-based search in a digital library is provided in [17]. There, preferences are defined for one single dimension, i.e., over keywords of the desired object. Due to that fact, the preferences are used

to sort the results and can not be exploited to filter irrelevant objects. [18] compares different approaches for catalog search and shows that the preference-based alternative is the most promising. However, the opportunities for defining preferences in the search form of the compared preference approach 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.

7 Discussion

In the previous sections we have seen that preference queries provide a powerful means for accessing large e-learning repositories, allowing the users to specify their preferences without having to give priority to one preference or another. As only optimal results are returned, all recommended items may be considered equally relevant. However, some user interface issues should be considered to maximize the benefits for the users.

It is a well-known phenomenon that 'first things' are perceived as being most important. As an example, Web users almost never look beyond the first page of search results [19]. This implies that the further from the start of a result list, the less likely it is that an item will be selected - even if they are as relevant as the results shown first. The same yields for the order in which preferences are elicited. Ideally, preferences should be elicited in such a way that any preference can be given in any order, with immediate and preferably visible feedback [20].

The closed-world assumption of skylining - that only those preferences explicitly stated by the user are relevant - is not always correct. When planning ahead, users typically have an idea of what they want, but are unable to directly express their needs. Only after having seen the first result set and inspecting its contents, they are able to communicate more advanced preferences. Such a process of *orienteering* [21] helps the users in finding what they want rather than what they ask for. As skylining only returns the results that are deemed the perfect solution, this process of orienteering might become disrupted: the user will not be aware of results that are initially second-best, but that might turn out to be better, based on preferences not yet expressed. For this reason, a careful combination of preference eliciting and critiquing [22] should be chosen.

8 Conclusions and Further Work

Search capabilities in existing educational repositories typically allow users to specify hard constraints a query must fulfill. However, users typically do not think on hard constraints but rather soft constraints such as "Monday is better but Tuesday would also be fine". Preferences allow users to specify their wishes in a way that can be processed by engines in order to return only the best matches based on such wishes, that is, those results that *dominate* the rest of potentially relevant ones. In this paper, we describe how such preferences and preference-based queries can be used in the context of search of learning

resources. We show how our approach is more expressive than existing ones and returns optimal result sets. In addition, we present our implementation as a web service in the Personal Reader Framework and demonstrate its value via some experiments in the Hannover University learning management system.

Our future work focuses on the improvement of our current prototype with optimized algorithms based on latest results on skylining research and investigate an enhancement to our user interface in order to allow the more expressive preferences that our implemented engine already support. In addition, preferences can also be used for automatic course generation (e.g., curriculum planning) and for recommendation algorithms [23], directions that we are currently exploring.

Acknowledgements

The authors would like to thank Mohammad Alrifai and Ingo Brunkhorst who provided the data set and helped to preprocess it. The authors' efforts were (partly) funded by the European Commission in the TENCompetence project (IST-2004-02787) (`www.tencompetence.org`).

References

1. Kobsa, A.: Generic User Modelling Systems. In: 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)
2. Brusilovsky, P.: Adaptive hypermedia. User Modeling and User-Adapted Interaction, Ten Year Anniversary Issue 11 (Alfred Kobsa, ed.), pp 87-110. (2001)
3. P, B., Peylo, C.: Adaptive and intelligent web-based educational systems. International Journal of Artificial Intelligence in Education, Special Issue on Adaptive and Intelligent Web-based Educational Systems **13** (2003) 159–172
4. Kießling, W., Köstler, G.: Preference sql - design, implementation, experiences. In: Proceedings of 28th International Conference on Very Large Data Bases (VLDB). (2002) 990–1001
5. Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: Proceedings of the 5th International Semantic Web Conference (ISWC), Athens, GA, USA (2006) 612–624
6. Kießling, W.: Foundations of preferences in database systems. In: Proceedings of the 28th International Conference on Very Large Data Bases, Hong Kong, China (2002) 311–322
7. Chomicki, J.: Preference formulas in relational queries. ACM Trans. Database Syst. **28**(4) (2003) 427–466
8. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany (2001)
9. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: Proceedings of the 27th International Conference on Very Large Databases (VLDB), Rome, Italy (2001)

10. Papadias, D., Tao, Y., Fu, G., Seeger, B.: 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) 467–478
11. Henze, N., Kriesell, M.: 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)
12. Baldoni, M., Baroglio, C., Brunkhorst, I., Marengo, E., Patti, V.: A personalization service for curriculum planning. In: ABIS 2006 - 14th Workshop on Adaptivity and User Modeling in Interactive Systems. (October 2006)
13. Henze, N., Krause, D.: Personalized access to web services in the semantic web. In: SWUI 2006 - 3rd International Semantic Web User Interaction Workshop, Athens, Georgia, USA (Nov 2006)
14. Abel, F., Baumgartner, R., Brooks, A., Enzi, C., Gottlob, G., Henze, N., Herzog, M., Kriesell, M., Nejdl, W., Tomaschewski, K.: The personal publication reader, semantic web challenge 2005. In: 4th International Semantic Web Conference. (November 2005)
15. Seaborne, A.: An open source implementation of sparql. In: WWW 2006 Developers track presentation, http://www2006.org/programme/item.php?id=d18 (2006)
16. Kießling, W., Köstler, G.: Preference SQL - design, implementation, experiences. In: 28th International Conference on Very Large Data Bases (VLDB 2002), Hong Kong, China. (2002)
17. Spyratos, N., Christophides, V.: Querying with preferences in a digital library. In: Federation over the Web. Volume LNAI 3847. Dagstuhl Seminar (N 05182) (May 2005)
18. Dring, S., Fischer, S., Kießling, W., Preisinger, T.: Optimizing the catalog search process for e-procurement platforms. deec **0** (2005) 39–48
19. Spink, A., Wolfram, D., Jansen, B.J., Saracevic, T.: Searching the web: The public and their queries. Journal of the American Society for Information Science and Technology **52 (3)** (2001) 226–234
20. Pu, P., Faltings, B., Torrens, M.: User-involved preference elicitation. In: Proc. IJCAI 2003 Workshop on Configuration. (2003)
21. Teevan, J., Alvarado, C., Ackerman, M.S., Karger, D.R.: The perfect search engine is not enough: a study of orienteering behavior in directed search. In: CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (2004) 415–422
22. Viappiani, P., Faltings, B., Pu, P.: Evaluating preference-based search tools: A tale of two approaches. In: Proceedings of The Twenty-First National Conference on Artificial Intelligence, July 16-20, 2006, Boston, Massachusetts, USA. (2006)
23. Satzger, B., Endres, M., Kießling, W.: A Preference-Based Recommender System. In: E-Commerce and Web Technologies. Springer Berlin / Heidelberg (2006)