# Exploiting Perceptual Similarity: Privacy-Preserving Cooperative Query Personalization

Christoph Lofi, Christian Nieke
Technische Universität Braunschweig
Mühlenpfordtstr. 23, 38114 Braunschweig, Germany
{lofi, nieke}@ifis.cs.tu-bs.de

**Abstract:** In this paper, we introduce privacy-preserving query personalization for experience items like movies, music, games or books. While these items are rather common, describing them with semantically meaningful attribute values is challenging, thus hindering traditional database query personalization. This often leads to the use of recommender systems, which, however, have several drawbacks as for example high barriers for new users joining the system, the inability to process dynamic queries, and severe privacy concerns due to requiring extensive long-term user profiles. We propose an alternative approach, representing experience items in a perceptual space using high-dimensional and semantically rich features. In order to query this space, we provide query-by-example personalization relying on the perceived similarity between items, and learn a user's current preferences with respect to the query on the fly. Furthermore, for query execution, our approach addresses privacy issues of recommender systems as we do not require user profiles for queries, do not leak any personal information during interaction, and allow users to stay anonymous while querying. In this paper, we provide the foundations of such a system and then extensively discuss and evaluate the performance of our approach under different assumptions. Also, suitable optimizations and modifications to ensure scalability on current hardware are presented.
(Keywords: Personalized Query Processing, Privacy in Information Systems)

## 1 Introduction

Effective personalization techniques have grown to be an integral and indispensable part of current information systems, and are essential to support users when faced with a flood of different choices. Here, two major approaches are common: a) Using SQL-style personalized queries on meta-data, which unfortunately require users to have extensive domain knowledge in order to formulate precise and efficient queries. Additionally, SQL-style queries are difficult for the large domain of *experience items* like movies, books, or music, as the commonly available meta-data is often not describing the items in a suitable fashion (e.g. if they are suspenseful, funny, or romantic.) b) Adapting recommender systems which proactively suggest items to users based on their user profile, and which became particularly popular in systems like Amazon or Netflix [1]: While many recommender systems provide recommendations of high quality [2], they have several shortcomings. Especially, for *each user* an elaborate user model needs to be built and stored, requiring up to hundreds of ratings until a user can get meaningful recommendations. This creates a high barrier for new users to join the system. But more severely, this user model contains exhaustive personal information on a user's preferences, her reaction to different

items, or her general likes and dislikes. In order to query or use the system, this information must be *clearly associated* with the respective user and needs to be *stored long-term*. Such profiles are highly valuable, and can easily be commercialized, abused, or even stolen. This situation raises many privacy concerns, and repels privacy-conscious users.

In this paper, we therefore present an alternative approach combining advantages of both recommender systems and SQL-based database personalization techniques, while at the same time avoiding many of the associated privacy risks. We realize this with privacy-preserving *query-by-example personalization*, which allows users to query for items fitting their current preferences easily without providing explicit feedback on attributes or their values. In order to obtain meaningful attribute values for each database object, we rely on *perceptual spaces* [3] which encode the implicitly perceived properties of each item, allowing to measure the consensually *perceived similarity* between given items. Similar to recommender systems, this perceptual information is mined from user-item ratings. However, we avoid the drawbacks of recommender systems: no user profiles are necessary to query the system, allowing situative, personalized, and anonymous ad-hoc queries.

In summary, our contributions in this paper are as follows:

- We present privacy-preserving queries for experience products using example-based queries on *perceptual spaces*. Our query-centered personalization is based on an adaption of *Bayesian navigation*.
- We discuss the advantages of our approach over SQL-style-based personalization and recommender systems focusing specifically on *privacy concerns* and *ease-of-use*
- We address the *performance and scalability* issues resulting from adapting Bayesian Navigation with *significant improvements* to the query execution process
- We evaluate the effectiveness of our approach in an *extensive user study*
- We evaluate the effects of our proposed algorithmic optimizations on the system's *performance*, *scalability* and *result quality*, and show that our approach can indeed be scaled to satisfy the demands of modern information systems.

## 2 Foundations & System Design

Experience products like movies, games, but also restaurants or hotels are items which are mostly characterized by non-hard attributes, i.e. they are subjectively experienced by their consumers and it is hard to find explicit attributes describing this resulting experience properly (e.g., a movie is considered humorous in a dry sense, as opposed to slapstick). Our approach is intended to allow users to easily explore a database with experience products by using personalized and privacy preserving query-by-example (QBE) navigation in order to avoid directly interacting with attribute values.

The intended semantics of our approach are complementary to both SQL-style personalization as well as to recommender systems: SQL-style personalization offers powerful queries using the usually available meta-data (like actors or genres for movies) and is suitable for users *who know exactly what they are looking for* (i.e. users need good domain knowledge and must be able to formalize a precise query). SQL queries will provide exact results and do not require any form of user profiling. In contrast to that, recommender systems or cross-selling systems focus on the case that *a user does not know what she is looking for*, and the system proactively presents suitable database items. In order to realize

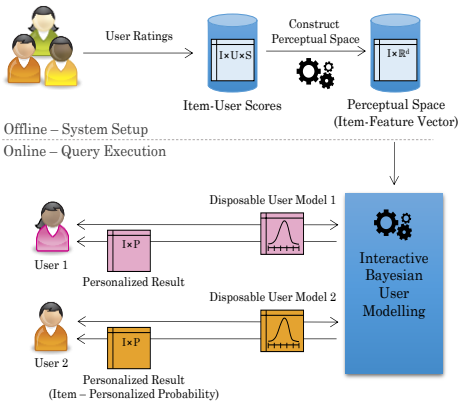


**Figure 1.** Basic System Design

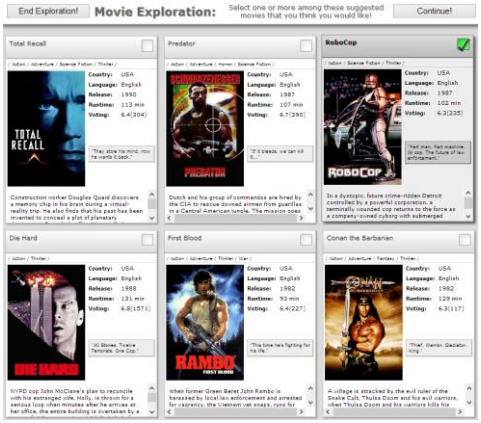


**Figure 2**. Screenshot of Prototype
2nd display after user provided "The Terminator (1984)" as start example

this, these systems need to "get to know" each user, i.e. they need to learn each user's likes, and dislikes. A scenario where a group of friends spontaneously decides to watch a movie would require to create an account for the group and vote on up to hundreds of movies, while our system allows to create a temporary profile on-the-fly while browsing. Additionally, the information aggregated in permanent user profiles can become very extensive and poses a serious threat to privacy. Furthermore, systems run into the risk of over-personalization by recommending only those items the system believes a user will like, which in turn makes it more likely that users consume and provide feedback only on similar items, locking them tightly into a "filter bubble" [4].

Our approach is in the middle-ground between both: the user poses a simple query by giving a vague example of what she is looking for, and can navigate through items selected and displayed due to their perceptual similarity to the example, by simply pointing out good suggestions in the display. It adopts a browsing behavior similar to physical book or video stores with "I know what I am looking for when I see it"-semantics. No direct interaction with attribute values is necessary. This type of query occurs quite often naturally, i.e. "There is a movie which I found interesting (because I liked it, or it was interesting in a way, or me and my friend both liked it, etc.), and now I am interested in more like that".

Accordingly, two major challenges are discussed in this paper: a) How can experience items be represented in a high-dimensional feature space such that meaningful similarity measurements and QBE navigation are possible? b) How can we personalize an example-based query in such a way that it respects the user's feedback actions and privacy?

Most popular QBE approaches in multimedia databases tried to operate on features extracted from the actual multimedia file itself, which could be low-level features (e.g. color histograms or pattern-based features), or so-called high-level features as for example in scene composition [5] or content-based semantic features [6] (e.g., presence of explosions or a flag, etc.). Here, our approach takes a completely different route, as our features result from external user ratings instead of being extracted from the media. Such information has been shown to be very informative, and semantically more meaningful to users than traditional meta data as, e.g. information about the director or actors (as shown in e.g. [7] for

movies). In this paper, we demonstrate how such semantically rich rating data can represent each item of an experience product database within a high-dimensional feature space. The idea is that the resulting space implicitly encodes how users perceived a movie, e.g., if it was funny, or if certain plot elements or tropes were present. For this task, we adapt *perceptual spaces*. Perceptual spaces have been introduced in [3], and are built on the basic assumption that each user who provides ratings on items has certain personal interests, likes, and dislikes, which steer and influence her rating behavior [8]. The resulting general system design of our approach is shown in Figure 1: in an offline system initialization phase, a large number of user ratings is processed into a perceptual space, and then our adapted Bayesian Navigation approach is used to personalize user queries by eliciting short term user profiles which are discarded after the query.

### 2.1 Personalization and Privacy

Privacy concerns severely impact a user's overall satisfaction with a Web-based system (as argued in [9]), and might even prevent them from using it altogether, if the balance between privacy concerns and perceived system utility becomes unfavorable. The focus of our system in terms of privacy is to allow all users to use the personalized query capabilities without requiring a user profile or pre-query preference elicitation. Especially, this means that browsing or querying our system requires no *long term user profiles* (unlike recommender systems), but only *temporary query profiles*, thus removing the need to store and protect this sensitive information. A single query profile will usually not be enough to extract a sufficiently distinctive pattern to identify a user, as it is not connected to other profiles (or a user id) and is only of the form: "an (anonymous) user wanted to start his query with 'The Terminator', and then selected 'Conan' out of a small set of movies proposed by the system" – it provides little insight into the wider preferences of a user.

But still, our system will require a small group of enthusiast users to provide identifiable rating data in order to construct the perceptual space. However, this construction process is completely decoupled from executing queries, and the perceptual space itself does not contain any user related information, not even in an anonymized or masked form. Even just the number of users that participated in its creation are not included. It is basically just a matrix of movie ids and their major perceptual dimensions (n=100 in our case). Therefore, approaches de-anonymizing ratings similar to the ones detailed in [10] cannot be applied. This fact could also allow a "trusted platform" like MovieLens to use its users' ratings to construct a perceptual space, which then could be used by another system like ours. In contrast to publishing anonymized rating data, publishing a perceptual space carries only minimal risks to the user's privacy. But in any case, even if users did decide to contribute ratings to build the space, all users can use the query capabilities of our system without leaving trails of personal information in an ad-hoc fashion.

### 2.2 Related Work

Content-based retrieval [11] and Query-by-example-based approaches [12, 13] have been very popular during the late 90s in the context of multi-media databases. The paradigm's main selling point is that querying high-dimensional data becomes very simple as it only requires the user to give a starting example and some easy-to-elicit feedback. However, QBE became less popular in recent years as, according to [14], the features used were often unsufficient, as discussed before. Also the Bayesian Retrieval approach [15]

adapted to our system was originally designed to work on image color histograms. Here, our approach takes a completely different route, as our feature space results from the reactions of users after consuming the media instead of being extracted from the media itself. Feature spaces constructed from ratings, like our perceptual space [3], have also been explored, as for example in [16]. However, in contrast to these works, we complement such feature spaces with personalized query capabilities, and discuss and address the design issues resulting from integrating such an approach into an information system. Approaches based on different flavors of Bayesian modeling have also been employed in recommender systems. In [17], users rate a small set of movie trailers in order to generate a user mood profile using Bayesian modeling, which later is integrated into a long-term user profile. However, they do not offer QBE functionality.

Approaches for privacy preservation in recommender systems mostly focus on the protection of the potentially vulnerable rating data. Here, many different approaches have been developed to either anonymize or encrypt the rating data to protect it from misuse by either the recommender platform itself, or from malicious attacks by 3rd parties. In most of these solutions, there is an inherent trade-off between privacy, accuracy, and efficiency [18]. For example in [19], user data is perturbed by adding fixed-distribution random values to each user rating, therefore hindering subsequent user identification, but also decreasing the quality of recommendations. A similar notion is followed by differentially-private recommender systems like [20], which add noise to the item similarity matrix in order to obfuscate the ratings originally provided.

### 2.3 Perceptual Spaces

Perceptual spaces have been introduced and formalized in [3], we therefore only briefly summarize the most relevant aspects in this section. Perceptual Spaces exploit the bias of users when rating items, which is influenced by personal likes and dislikes towards properties of the rated item. They heavily rely on factor models, a technique popular in recommender systems research [2]. Factor models have originally been developed to estimate the value of non-observed ratings for the purpose of recommending new (yet unrated) items to existing users, but can also be beneficial beyond recommendation tasks [21]. We assume that a perceptional space is a $d$-dimensional space as follows: each user and each item is represented as a $d$-dimensional numeric vector. The vector of a user represents her personality, i.e., the degree by which she likes or dislikes certain characteristics. Likewise, item vectors represent the degree to which an item shows the same characteristics. Items which are perceived similarly in some aspect have similar vectors. As we are only interested in items, user vectors are not stored and are discarded in later stages. Furthermore, each user rating can be seen as a function of the user vector and item vector. This assumption reflects established models of human preferences and is well-accepted in recommender systems research [2]. All model parameters are estimated by minimizing a cost function measuring the deviation between the actual observed ratings and those predicted by the model. By formulating this as an optimization problem, user and item vectors can be found that fit the given rating data best.

Formally, a large and sparse user-item-rating matrix is given, containing only rating for around 1-2% of all user-item pairs. The goal is to find a matrix $A = (a_{m,k}) \in \mathbb{R}^{n_M \times d}$ representing movies as $d$-dimensional coordinates. To achieve this, we also need a helper matrix $B = (b_{u,k}) \in \mathbb{R}^{n_U \times d}$, representing user in the same space. Then, we use a factor

model representing a rating function $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. Basically, this function can predict missing ratings given user and item vectors. We approximate this function and the involved vectors/matrices, we use Euclidian Embedding (as in [22]), and we want the distance between a movie vector $a_m$ and user vector $b_u$ to be small if user $u$ likes movie $m$; otherwise, it should be large. To account for general effects independent of personal preferences, for each movie $m$ and user $u$, we introduce the model parameters $\delta_m$ and $\delta_u$, which represent a generic movie rating bias relative to the average rating $\mu$. Then, a rating of a movie $m$ by a user $u$ can be predicted by $\hat{r}_{m,u} = \mu + \delta_m + \delta_u - dis_E^2(a_m, b_u)$, i.e. the average rating of all movies (e.g., $\mu$=6.2 out of 1..10) plus the user bias (e.g., $\delta_u$=-1.6 representing a critical user always rating worse than others) and the movie bias (e.g., an overall good movie with an average rating of 8.4, so $\delta_m$=2.2). The last term, $dis_E(\cdot,\cdot)$, represents the distance of the movie vector and the user vector in a $d$-dimensional space. Finally, all movie vectors (and therefore the matrix $A$) are approximated by solving a least squares optimization problem with all instances of the above equation for which a rating is known (including a correction for noise). The result represents our perceptual space.

Unfortunately, the resulting features in this space are implicit and have no direct real-world interpretation, and therefore not suitable for SQL-style queries. However, they allow for measuring perceived similarity effectively (i.e. the distance between the feature vectors). This now allows using the query-by-example paradigm, which provides simple query formulization without the need to explicitly refer to any features.

### 2.4 Basic Bayesian Retrieval

For approaching query-by-example personalization, we will adapt Bayesian Retrieval as shown in e.g. [15]. Such approaches have been successfully used in multimedia databases research, but have also been adapted to general database retrieval [23]. In short, our approach aims at computing for each database object the probability of the user being interested in it considering the feedback she has provided on a selection of items during a cooperative preference elicitation. We will only briefly summarize the basic theory of Bayesian retrieval and user modeling in this subsection, before highlighting the modifications necessary to adapt the model to our context in the next section, and discussing techniques for improving the computational and memory performance of the approach in section 4. From a user's perspective the interaction style of this approach is similar to non-personalizing similarity navigation (i.e. repeatedly navigating from one item to similar ones), but we will show that incorporating Bayesian personalization will lead to significantly better and quicker interaction performance, outshining similarity navigation.

Elicitation of user preferences is performed interactively over several *steps*. In each step $t = 1,2, \ldots$, the user will be shown a set of objects $D_t$ (the so-called *display*) from the database containing our previously computed perceptual space with $n_M$ objects. The choice of objects in $D_t$ depends on a *selection strategy* (discussed later), which relies on analyzing the probability estimates for each database object representing the belief that it is the one the user is looking for given the current interaction history. This is formalized as follows: the database objects are denoted as $O_1, O_2, \ldots, O_n$. Each object $O_i$ is annotated with a probability of being the user's target $O$. Here, the target is the best suited object in the database to fulfill the user's current needs, which is of course yet undefined.

After reviewing the objects, the user will provide feedback on the display's items in form of a user action $A_t$. In our case, this is simply selecting any items that "look right"

(similar to judging covers in a video store). The initial *a priori estimate* before starting the user interaction for each object $O_i$ will be denoted as $P(O = O_i)$. Again, several strategies are viable for providing this *startup distribution* of the a priori estimates (discussed later). After the interaction has started, the probabilities are updated to *a posteriori estimates* respecting the current *interaction history* denoted as: $H_t = A_0,\ D_0,\ A_1,\ D_1,\ \dots,\ A_t,\ D_t$. Since $D_t$ is deterministically given by the selection strategy and $H_{t-1}$ is known, we arrive at the following formula (see [15] for details):

*Simplified Bayesian Update:* For each $O_i$ in the perceptual space:

$$P(O = O_i \mid H_t) = P(O = O_i \mid D_t,\ A_t, H_{t-1}) = \frac{\boldsymbol{P(A_t \mid O = O_i,\ D_t, H_{t-1})}\ P(O = O_i \mid H_{t-1})}{\sum_{j=1}^{n_M} P(A_t \mid O = O_j,\ D_t, H_{t-1})\ P\big(O = O_j \mid H_{t-1}\big)}.$$

After each user feedback, this update operation has to be performed for all items in the perceptual space, thus forming the *user model* representing in which items the current user is likely interested. This is represented in the model by all a-posteriori probabilities of all items $P(O = O_i \mid H_t)$ with $1 \le i \le n_M$.

The term $P(O = O_j \mid H_{t-1})$ can be computed recursively until the a priori approximation given by the start distribution is reached. But the central term within the previous equitation $P(A_t \mid O = O_i,\ D_t,\ H_{t-1})$ remains difficult, i.e. the probability that the user will actually perform the current action $A_t$ given that the current database object $O_i$ is indeed the target $O$ given the history $H_{t-1}$. This term predicts a user's action considering all currently known information, and is provided via a *user prediction model* (next section).

## 3 Adapting Bayesian Retrieval

For adapting Bayesian retrieval to our usage scenario, we need to create a suitable user prediction model, startup distribution, and selection strategy.

The **user prediction model** provides an implementation for evaluating the term $P(A_t \mid O = O_i,\ D_t,\ H_{t-1})$ from the previous section. This makes up the "semantic core" of the calculation and is essential in determining whether the calculated probabilities will actually correctly represent the users' preferences. To allow an easy and intuitive interaction with the system, we opt for item-based feedback during a user feedback cycle, meaning that a user simply selects any number of items from the display $D_t = \{X_1, \dots, X_{n_D}\}$ she wants to use as positive examples for further personalization. $n_D$ is a system parameter denoting how many items are shown in each display, and can be adjusted to the type of items and display device which is used (we used $n_D = 9$ in our Web-based prototype). Then, following [15] we also take a soft-min approach for item-based feedback, assuming that the user behaves time invariant (i.e. her decisions are based on her implicitly known target object and we can drop $H_{t-1}$). Accordingly, the probability for each decision on each single object $X_a$ from the display can be modeled as:

$$P_{soft}\big(A = a \big| X_1, \dots, X_{n_D}, O_j\big) = \frac{\exp(-d(X_a, O_j))}{\sum_{i=1}^{n_D} \exp(-d(X_i, O_j))}$$

In this formula, $d(X_a, O_j)$ denotes the *distance* between item $X_a$ and the target object $O_j$ and the approach yields highest values for those items $X_a$ closest to the target $O_j$. The quality of this approach is of course highly dependent on choosing a meaningful metric for the distance, which corresponds to the similarity of the objects. We use the *Euclidian*

*distance measured in the perceptual space*, which is a significant contribution as it provides a semantically meaningful high-dimensional feature space for augmenting regular database objects, and distances measured in this space represent the *consensual subjective semantic similarity* between the objects elicited from a large number of users. These measures are significantly more meaningful than similarity measured on typical meta-data usually available in information systems [3]. Using the assumption of independent decisions, a combined decision can be calculated by multiplying the probabilities of each single decision, e.g. $P_{soft}(A = 1, 2) = P_{soft}(A = 1) * P_{soft}(A = 2)$.

The **selection strategy** decides which objects are included in a display $D_i$ for each feedback cycle $t_i$ and is important to allow a satisfying interaction with the system. Basically, two major approaches are possible here:
a) *Most-probable strategies* select objects with the highest a posteriori probability $P(O = O_i | H_t)$. These selection strategies tend to favor similar objects at the beginning of the interaction, and at a first glance might resemble similarity search for the first one or two displays. However, after few feedback cycles, this strategy will be able to cross larger distances in the space depending on user input. But still, in its naïve form, this strategy is prone to getting stuck in clusters repeating the same objects over and over.
b) *Most-informative strategies* aim at maximizing the information gain in each feedback step [15] by diversifying the display (i.e. displaying those items which would impact the a posteriori probabilities most). This usually leads to higher navigation speeds, and users can traverse the space quickly with only few feedback steps.

While the most-informative strategy shows superior performance (i.e. it needs less user interactions to find a certain target in the space), we found in our pre-study that users were easily confused by the resulting displays, as the "long distance links" to yet untouched areas of the space were perceived as mistakes of the algorithm (e.g. presenting the family movie "Finding Nemo" after a user selected the action movie "The Terminator").

We therefore opted for adapting the most-probable strategy to our needs, sacrificing some interaction speed for increased user satisfaction. In order to avoid being caught in local maxima or clusters, we only consider objects which have not yet been displayed to the user, resulting in the *most-probable unseen strategy*.

The **startup distribution** defines the a-priori probability $P(O = O_i)$ for each database object $O_i$ of being the target object $O$ before the user starts interacting, which can alleviate the cold start problems of the not yet personalized system. Without any additional assumptions, the naïve approach is a uniform startup distribution with $P(O = O_i) = 1/n$ for each $O_i \in \{O_1, \dots, O_n\}$. Alternatively, additional information on the database objects could be incorporated into the startup distribution as for example average user ratings or popularity measures. In our prototype implementation, we initialized the system with a simple uniform startup distribution, modified by an explicit, user-provided example. For this, a single feedback-step is transparently executed during startup (i.e. the user does not see this first feedback step). Here, we evaluated two approaches:
*Free Example*: The user freely provides an example to start the query and we simulate the first user action $A_0$ as selecting the example from a display $D_0$ which contains only two items: the example and the movie in the dataset with the maximal distance to that selected movie. Therefore, the first display the user will see is $D_1$ which is already strongly influenced by the start example.
*Supported Free Example*: In our pre-studies, we found that some users had a hard time

thinking of a good example for starting the query. We therefore presented a set of 12 popular example movies hand-picked from different genres, allowing the user to either provide her own example or pick one of ours. $D_0$ is then made up of these 12 examples plus the optional freeform example.

## 4 Mastering Performance Issues

For performance evaluation, we focus only on costs for executing individual user queries, as the costs for maintaining the perceptual space are negligible. The perceptual space is not impacted enough by adding single ratings to justify the effort for continuous updates.

In contrast, query performance raises several demanding issues: Bayesian retrieval as introduced in the last sections requires storing the a posteriori probabilities for each database object individually in the user model of each user. Furthermore, for each user interaction, all probabilities in that user's model need to be updated, and each update of a single probability requires considering all other probabilities. Clearly, this situation presents severe challenges to scalability with respect to the number of objects in the perceptual space $n_M$, but also to the number of concurrent system users $n_{UC}$. Both memory space for storing the user models as well as computation time are negatively impacted. Here, the required memory is in $O(n_{UC} * n_m)$, while the computation effort per query is in $O(n_{UC} * n_m^2)$. By simply caching the denominator term in the Bayesian update formula in 2.4, the resulting algorithm can be brought to linear complexity in $O(n_{UC} * n_m)$.

Therefore, we introduce *locality-restricted Bayesian updates* in this section. While these efforts do not improve the theoretical worst-case complexity, they drastically improve the actual time and memory needed for executing a query, making our approach feasible on currently available hardware, even for larger perceptual spaces and many concurrent users. The basic idea is to restrict the probability updates to the *relevant* parts of the perceptual space, i.e. those objects that are close to those selected before. If a user started her query with a family comedy, we can ignore horror movies unless her feedback indicates otherwise by steering toward that direction. This allows us to extend the area under observation slowly and in a directed fashion, without affecting semantics too much.

We implement this by introducing, for each user, the set $M_j$ for memorizing the relevant part of that user's user model, i.e. the objects that are close to formerly chosen objects, which therefore have a high a posteriori probability after interaction step $t_j$. As the remaining objects are far from the objects that are currently considered to be likely, they will have a low probability and can be ignored until the exploration leads to their section of the space. The set $M_{-1}$ is initialized with the example object selected by the user and its $\delta_N$ nearest neighbors in the perceptual space. The parameter $\delta_N$ can be freely chosen during system setup, and we will show the effects for different $\delta_N$'s in the evaluation section. After each user interaction $t_j$, the previous set $M_{j-1}$ is expanded by adding the $\delta_N$ nearest neighbors in the perceptual space of each display object $X$ that was selected by the user in $t_j$. As the probabilities of the newly added objects are still unknown, we will heuristically assume that their probability is similar to the closest neighbor already in the set $M_{j-1}$ and use the known probability of this neighbor to initialize the new object.

After all new objects were added to the set $M_{j-1}$, the locality-restricted Bayesian update computes the new set $M_j$ by calculating the a posteriori probabilities of each object and their respective probabilities in $M_{j-1}$ similar to its original version, but ignoring all objects not being in $M_{j-1}$. After that, $M_{j-1}$ can be discarded and we continue with the extended set $M_j$. *Locality-Restricted Bayesian Update at $t_j$:*

For each $O_i \in M_{j-1}$: $$P(O = O_i \mid H_t) = \frac{P(A_t \mid O = O_i,\ D_t, H_{t-1})\ P(O = O_i \mid H_{t-1})}{\sum_{O_k \in M_{j-1}} P(A_t \mid O = O_k,\ D_t, H_{t-1})\ P(O = O_k \mid H_{t-1})}$$

with $M_{-1} = \{\delta_N NN(selected(A_0))\}$ and $M_i = M_{i-1} \cup \delta_N NN(selected(A_i))$

In our locality-restricted variant, the actual effort needed to store or update the user model after each interaction is clearly reduced as the set $M_i$ is significantly smaller than the whole perceptual space. But this also incurs a penalty on the semantics of the approach, as relevant objects, which would have been assigned a high probability using a non-restricted approach, might not yet be part of the latest set $M_i$, and are therefore ignored.

Of course finding nearest neighbors can be very expensive as well, but in our current prototype, we simply materialize a full index of the $\delta_N$-nearest neighbors of each object offline when importing a new perceptual space, which even for 100k objects is easily feasible even on low-end machines. This provides us with near instant access to the nearest neighbors independent of the size of the perceptual space.

## 5 Evaluations

In this section, we present extensive evaluations of our privacy-preserving query-by-example personalization on perceptual spaces. In the first set of evaluations focusing on usability and semantics, we use a real-life dataset with movie ratings for initializing the perceptual space, and have real users interact with a prototype implementation. In addition to this study, simulations focusing on performance aspects of the approach under different assumptions and parameters were performed. Also, these simulations are run on increasingly larger datasets to analyze scalability issues. We close the section with a discussion of the results and their adaptability to other domains.

The perceptual space used in our real-word data experiment is based on the dataset released during the Netflix Prize challenge [1], and consists of 103M ratings given by 480k users on 17k video and movie titles. This dataset is still one of the largest user-item-rating datasets which are available to the community. All titles are from 2005 and older. We filtered out all TV series and retained only full feature movies for our evaluation, leaving 11,976 movies. The initial construction of the 100-dimensional space took slightly below 2 hours on a standard notebook computer.

For simulations on synthetic data, we randomly generated spaces with 100-dimensions and a varying amount of database items assuming an independent uniform distribution of the attribute values. Although real perceptual spaces are usually not uniformly distributed, our experiments comparing the real perceptual space with the uniformly generated ones

showed that the results are close enough to allow drawing some conclusions about performance from simulations to real-world behavior (see section 5.2). Our prototype was implemented in Java 7, and the perceptual space as well as all probabilities computed for the user models were held using the H2 Main Memory DBMS. Our prototype ran on a notebook computer with a Core-i7 at 2.4GHz and 10GB of main memory usable for Java.

### 5.1 User Study

Our user study was performed using the CrowdFlower.com crowdsourcing platform to recruit participants. The users were redirected to our Web-based prototype implementation (screenshot in Figure 2 in section 2), answered a survey afterwards, and were finally paid for their efforts. 179 users participated, and as we allowed users to perform additional (unpaid) interactions, this resulted in 288 completed queries. In the survey concluding the evaluation, we asked users for feedback on a number of statements, on a scale from 1: strongly disagree, 3: neutral, to 5: strongly agree, and we counted 4 and 5 as explicit agreement and 1 and 2 as explicit disagreement. Our test users were fairly evenly distributed over different age classes, gender, movie knowledge and experience with existing movie subscription system like Netflix or Amazon Instant.

We asked the users to test the system in an explorative fashion with the task of deciding on a suitable movie for an imaginative movie night in mind. We allowed them to start the interaction with an example of their choice, but additionally supported this with displaying a selection of 12 manually selected examples for inspiring the choice of a start example. Users were then able to explore the movie space using Bayesian Navigation as they saw fit, and could end the interaction whenever they felt they had seen enough.

After querying, 74% of all users explicitly claimed to have found a suitable movie, but only 6% explicitly disagreed (the other users were neutral). However, this statement has to be taken carefully as our users often had certain expectations of what movie they should discover when certain examples and feedback were provided. As our dataset was restricted to movies released in or before 2005, many users complained that they expected newer movies in the display which were simply not part of our dataset (e.g. they expected to find "The Hobbit (2012)" when providing "The Lord of the Rings 1 (2001)" as a start example).

To assess the semantic quality of each feedback cycle's display, we asked our test users if the screens presented by the system were a good fit for the feedback they had given, and got 60% explicit agreement and only 12% explicit disagreement to this statement. While this does not allow to quantify how close our displays and query results are to the "best possible ones", it shows that users felt that they were "good enough".

When asked if they prefer our approach over the regular query approaches as used by e.g. Amazon Instant or Netflix (recommendations with SQL-style querying and most-popular-in-category lists), 54% of all participants stated that they explicitly prefer our approach, while only 9% explicitly disagreed. However, when being asked whether they think if this approach is a valuable addition to current state-of-the-art interaction paradigms, the explicit agreement increased to 73% of all users, while only 9% think our approach is not a useful extension. This result underlines our notion that personalized example-based queries are a very valuable addition to SQL-style filtering, categorization, and recommender systems, but of course are not a full-fledged replacement. SQL-style queries cover the case where a user knows exactly what she is looking for, while recommender systems proactively recommend an item, often even without a query. Our approach is in

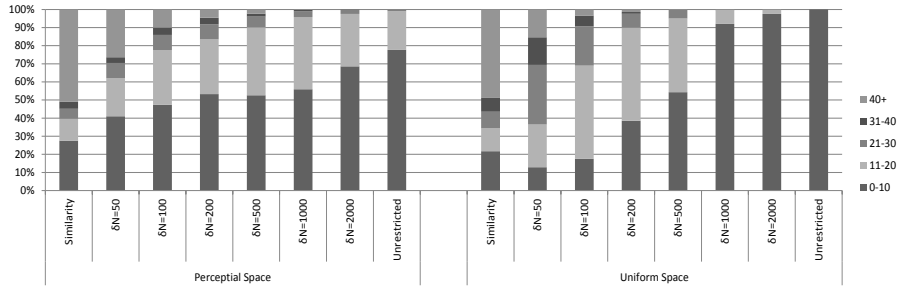


**Figure 3.** Simulated Query Runtimes for $\boldsymbol{n_m = 11,976}$
in number of displays that needed feedback to reach a randomly selected target

the middle ground, where users have some intuition about their preferences, but still cannot precisely formulate them as required for an SQL query.

### 5.2 Performance Simulation

In this section, we evaluate our approach using simulated user interactions on the Netflix perceptual space and on different artificially created ones. As it is rather hard to evaluate the semantic usefulness of an explorative query paradigm in simulations, the main purpose of this section is to showcase the impact of different factors on the systems performance. Especially, we are interested in the scalability of our approach and the impact of our proposed heuristic on memory and CPU consumption to show the expected runtime performance when applied to big datasets.

Therefore, as natural browsing behavior is hard to model, we chose a similar approach as in [23]. In this type of simulation, a particular movie has to be found as quickly as possible, starting from the initial example. While this setting does not represent a typical user's behavior, in addition to assessing performance and scalability, this setup is also able to provide some insight into the speed in which the space could theoretically be traversed.

In order to perform the experiments, we randomly select a movie from the database to be a user's target movie, i.e. the intended perfect match for the user's current preferences which has to be found. The simulation algorithm then picks the best choice of a given feedback display, which is the movie with the smallest distance to the target in the perceptual space, and continues until the target appears in the feedback display. The simulation only had a limited set of start example movies to choose from, depending on the experiment either the 12 handpicked movies we used for our supported free example startup when using the Netflix perceptual space, or the 12 centroid objects of the clusters resulting from a k-means-clustering of the artificial spaces. For all experimental results presented in the following, we performed at least 1,000 independent simulation runs.

To assess the effectiveness of our approach in this simulation setting, we measured how many displays (with 9 choices each) a simulated user needed to asses before finding the target. In Figure 3 we show this for the Netflix perceptual space, and a uniformly generated one of the same size, each with locality-restricted updates with varying $\delta_N$ and an unrestricted version. Furthermore, we included an approach based only on similarity search without Bayesian modeling, i.e. a user provides a start example and is shown a display of the 9 yet-unseen nearest neighbors in the perceptual space, selects one, and obtains a new display with that movie's yet-unseen closets neighbors. Here, we can see that Bayesian retrieval approaches outperform similarity search, as they need significantly less displays,

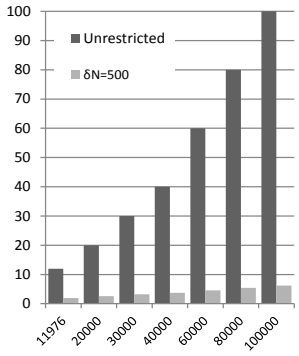


**Figure 4**. Memory for User Profiles
*y*: #1000 items stored; *x*: size $n_M$

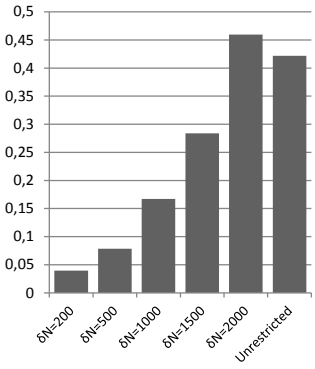


**Figure 5.** Time per Update with $n_M = 100k$
y: time in sec

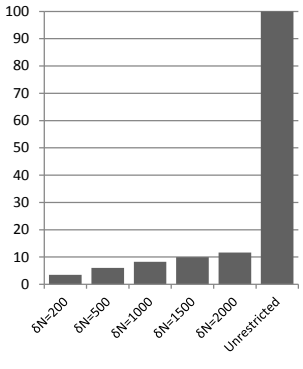


**Figure 6.** Memory for User Profiles $n_M = 100k$
*y*: #1000 items stored

while the locality restricted Bayesian update deteriorates towards similarity search for very small $\delta_N$ and converges towards unrestricted Bayesian retrieval for sufficiently large $\delta_N$.

For $\delta_N = 1{,}000$ the target is found after 10.7 displays on average in the perceptual space, while additional experiments using a uniform space with 100k items resulted in a slightly higher average of 17.7 displays. The parameter $\delta_N$ affects the uniform space more strongly than a real perceptual space, but it can be seen that the general behavior of the algorithm in a perceptual space can roughly be approximated by the uniform space. Especially for values of $\delta_N = 500$ the observed query behavior is similar; therefore we use this value when evaluating the scalability of our algorithm in the next experiments.

Figure 4 shows the memory usage of our locally restricted Bayesian update approach for storing the final user model after a query found its intended target for increasing dataset sizes. It can clearly be seen that the locality restriction scales favorably: the user model requires only about 16% the size of the unrestricted approach for the uniform dataset with 11,976 items (the size of the original perceptual space) while for the larger space containing 100k items, this percentage even decreases down to 6%.

In Figure 5, we investigate the average time needed per single locality-restricted update operation on a large synthetic perceptual space with 100k objects. Using larger $\delta_N$ unsurprisingly increases the updates times, which can become higher than the time needed for unrestricted updates when using very high values of $\delta_N$ (but still, all times are well below 0.5 seconds which is perfectly fine for Web applications). However, for $\delta_N = 1{,}500$ which provides semantically very similar results to the unrestricted approach, there is still a computation time advantage of 138ms. Also, the number of items to be held in each user model increases with $\delta_N$ (see Figure 6). However, here it can clearly be seen that locality restriction significantly affects the amount of memory needed, and even for $\delta_N = 2{,}000$, only 11.7% of the memory required by the unrestricted variant is consumed.

### 5.3 Adapting to different domains

While we used movies as a running example to demonstrate our system design, our approach can be adapted to any experience product which is frequently consumed and rated. Unfortunately, the free availability of user-item-rating information is limited, especially after the scandals following the release of the Netflix dataset (which is still the most

extensive dataset up to now). Therefore, we resorted to rating data crawled from the Web in 2012 [3] to show the applicability in other domains. The first data set contains restaurant ratings in the San Francisco area obtained from yelp.com (3,811 restaurants; 128,486 users; 626,038 ratings). It is only a small item set with a large but rather inactive user base (i.e. only few ratings per user). In contrast, the second dataset consists of ratings of board games from boardgamesgeek.com (32,337 games; 73,705 users; 3,536,455 ratings). Here, a rather small user base is a highly active, rating a large number of items each.

For both datasets, we performed simulations similar to those described in the last subsection and found that the yelp.com data set could be traversed in 8.5 screens on average, while the board games data set needed 11.1 screens on average. No further user studies have been performed so far, as users recruited from crowd sourcing platform most likely lack the domain knowledge to evaluate a system using such data, but real user experiments with these datasets will be part of future works. While we currently do not provide experimentally supported insight into the resulting user experience, we can claim that the adaptation to different domains is at least technically possible and the results are comparable to movies for the simulations.

## 6 Summary & Discussion

In this paper, we demonstrated privacy-preserving query personalization using Bayesian query-by-example techniques on perceptual spaces. This allows users to query an information system with experience products in an anonymous and ad-hoc fashion, not requiring profiling or explicit preference elicitation. Our system is in the middle ground between SQL-style queries, which need the user to be able to formulize a precise query, and recommender systems which proactively suggest items to users. It allows explorative queries with "I know what I am looking for when I see it" semantics, which also helps users to break out from the "filter bubble" created by recommender systems.

We have shown how to build such a system, and how perceptual spaces can be combined with, and adapted to our proposed Bayesian query-by-example framework. Furthermore, we demonstrated how the resulting performance and scalability issues can be overcome by introducing locality-restricted Bayesian updates which provide tremendous advantages in terms of memory consumption, and saves a significant amount of computation time. Finally, in our extensive evaluations we have presented a user study with over 175 participants, and obtained their opinion on the effectiveness of the system. Furthermore, we have performed several experiments using simulations on synthetic data, and showed that the approach is indeed scalable and adaptable even on lower-end hardware like a laptop.

While we achieved semantically meaningful and simple to formulate queries without requiring user profiling during the query process, our current approach still relies on aggregating user-item-ratings from a group of enthusiast users to construct the perceptual space, which still carries certain privacy concerns. One of the future directions of research is hence the development of representations of experience items which are similarly expressive as perceptual spaces, but can be constructed using data not exhibiting any privacy concerns, like for example using Linked Open Data sources or anonymized product reviews. Furthermore, while system usability is already quite good as is, it could be further improved by explaining the system behavior to users. This especially covers describing

why the presented items are a good match to the query, e.g., “The items shown have been selected because they have settings similar to movie X, but share the humor of movie Y – and you used both movies as positive examples”.

## References


1. Bell, R.M., Koren, Y., Volinsky, C.: All together now: A perspective on the Netflix Price. CHANCE. 23, 24–24 (2010).
2. Koren, Y., Bell, R.: Advances in Collaborative Filtering. Recommender Systems Handbook. pp. 145–186 (2011).
3. Selke, J., Lofi, C., Balke, W.-T.: Pushing the Boundaries of Crowd-Enabled Databases with Query-Driven Schema Expansion. Proc. VLDB. 5, 538–549 (2012).
4. Pariser, E.: The Filter Bubble: What the Internet Is Hiding from You. Penguin Books (2011).
5. Sundaram, H., Chang, S.-F.: Computable scenes and structures in films. IEEE Trans. Multimed. 4, 482–491 (2002).
6. Neo, S.-Y., Zhao, J., Kan, M.-Y., Chua, T.-S.: Video Retrieval Using High Level Features: Exploiting Query Matching and Confidence-Based Weighting. Lect. Notes Comput. Sci. 4071, 143–152 (2006).
7. Pilászy, I., Tikk, D.: Recommending new movies: even a few ratings are more valuable than metadata. ACM Conf. on Recom. Systems (RecSys). , New York, USA (2009).
8. Kahneman, D., Tversky, A.: Psychology of Preferences. Sci. Am. 246, 160–173 (1982).
9. Knijnenburg, B.P., Willemsen, M.C., Gantner, Z., Soncu, H., Newell, C.: Explaining the user experience of recommender systems. UMUAI. 22, 441–504 (2012).
10. Narayanan, A., Shmatikov, V.: Robust De-anonymization of Large Sparse Dataset. IEEE Symposium on Security and Privacy. , Oakland, USA (2008).
11. Yoshitaka, A., Lchikawa, T.: A survey on content-based retrieval for multimedia databases. IEEE Trans. Knowl. Data Eng. 11, (1999).
12. Kato, T., Kurita, T., Otsu, N., Hirata, K.: A sketch retrieval method for full color image database-query by visual example. Pattern Recognit. (1992).
13. Niblack, C.W., Barber, R., Equitz, W., Flickner, M.D., Taubin, E.H.G.D.P., Yanker, P., Faloutsos, C., Taubin, G.: QBIC project: querying images by content, using color, texture, and shape. Storage and Retrieval for Image and Video Databases. , San Jose, USA (1993).
14. Santini, S., Jain, R.: Beyond query by example. ACM Multimedia. , Bristol, UK (1998).
15. I.J. Cox, Miller, M.L., Minka, T.P., Papathomas, T.V., Yianilos, P.N.: The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments. IEEE Trans. Image Process. (2000).
16. Slaney, M., White, W.: Similarity Based on Rating Data. 8th Int. Conf. on Music Information Retrieval (ISMIR). , Vienna, Austria (2007).
17. Babas, K., Chalkiadakis, G., Tripolitakis, E.: You Are What You Consume: A Bayesian Method for Personalized Recommendations. RecSys ’13. , Hong Kong, China (2013).
18. Jeckmans, A.J.P., Beye, M., Erkin, Z., Hartel, P., Lagendijk, R.L., Tang, Q.: Privacy in Recommender Systems. Social Media Retrieval. pp. 263–281. Springer London (2013).
19. Polat, H., Du, W.: SVD-based collaborative filtering with privacy. ACM Symp. on Applied Computing (SAC). , Santa Fe, USA (2005).
20. McSherry, F., Mironov, I.: Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD). , Paris, France (2009).
21. Selke, J., Balke, W.T.: Extracting Features from Ratings: The Role of Factor Models. Workshop on Advances in Preference Handling. , Lisbon, Portugal (2011).
22. Khoshneshin, M., Street, W.: Collaborative filtering via euclidean embedding. 4th ACM Conf. on Recommender Systems (RecSys). , Chicago, Illinois, USA (2010).

23. Lofi, C., Nieke, C., Balke, W.-T.: Mobile Product Browsing Using Bayesian Retrieval. IEEE Conf. Commerce and Enterprise Comp. (CEC). , Shanghai, China (2010).