Ranking Query Results using Context-Aware Preferences

Arthur H. van Bunningen, Maarten M. Fokkinga, Peter M.G. Apers
Centre for Telematics and Information Technology
University of Twente, The Netherlands
{bunninge,fokkinga,apers}@cs.utwente.nl

Ling Feng
Department of Computer Science & Technology
Tsinghua University, China
fengling@tsinghua.edu.cn

Abstract

To better serve users' information needs without requiring comprehensive queries from users, a simple yet effective technique is to explore the preferences of users. Since these preferences can differ for each context of the user, we introduce context-aware preferences. To anchor the semantics of context-aware preferences in a traditional probabilistic model of information retrieval, we present a semantics for context-aware preferences based on the history of the user. An advantage of this approach is that the inherent uncertainty of context information, due to the fact that context information is often acquired through sensors, can be easily integrated in the model. To demonstrate the feasibility of our approach and current bottlenecks we provide a naive implementation of our technique based on database views.

1. Introduction

Nowadays, more and more information becomes available in digital form. To be able to guide users through this wealth of information, a possibility is to adapt the provided information to the current situation (i.e., *context*) of the user. Research in this area of *context-awareness* originates from the vision of Mark Weiser that the machines should fit human environment instead of forcing humans to enter theirs [18]. From a database perspective this implies, among others, adapting query results to the context of the user at the time of querying [8]. Challenges raised by this implication are to express the preferences of the user in different contexts and to score tuples based on these preferences. These challenges become especially difficult since most context information results from sensors and is therefore uncertain. Moreover, the preferences of the user might differ in strength.

In the field of context-awareness, previous research focused on simple preferences in small scale systems using little context information, whereas in the database field, most research on preferences focused on hard-coded preferences, without addressing the demands of context-aware systems such as reasoning, traceability, uncertainty etc. [16]. In this paper we investigate how to score the results of a database query based on the context of the user, leading to a ranked list of results. To deal with the demands of context-awareness, we base our approach on our previous work to represent contextual features as concept expressions in Description Logics. Furthermore, we use event expressions to deal with the uncertainty of context [17]. The approach results in an explanatory scoring method for documents, which we relate to traditional, non context-aware, information retrieval.

As a an example, consider a new kind of media player, called *TVTouch*, which is able to play both (recorded) television programs and movies. The system is context-aware in a sense that it can provide the user with a list of suggested programs based on the current context of the user. We imagine a usage scenario in which a user Peter uses *TVTouch* to provide him each morning with a list of suggested programs containing traffic bulletins, weather bulletins, news, entertainment etc. based on his activities that day and the importance of the separate items (e.g., if there is news about a subject in which Peter is interested, it should be recommended). To be able to realize such a scenario, many research questions have to be resolved, such as automatically tagging the news items with subjects, sensing context information, and scoring programs. The contribution of this paper is to deal with the third research question and to present a technique which is able to provide a motivated score for the programs based on the preferences and context of Peter. In this exam-

ple we could imagine the following typical query from the *TVTouch* system to the underlying database:

```
SELECT name, preferencescore
FROM Programs
WHERE preferencescore > 0.5
ORDER BY preferencescore DESC
```

where the underlying context-aware database would dynamically assign a preference score to each program, based on the context of the user. The goal of our system is to provide a motivated value of the "preferencescore"-attribute for each tuple.

The rest of the paper is organized as follows. First, we explain how context-aware retrieval relates to traditional probabilistic information retrieval in Section 2. In Section 3 we then present our probabilistic model of context-aware relevance. To realize this model we need a way to describe the relation between the context and preferences of the user, which we achieve using *scored preference rules* which are explicated in Section 4. Using our probabilistic model we present a naive implementation in Section 5. We end with discussions on improvements in Section 6 and conclude the paper in Section 7.

1.1. Related work

Research in the area of ranking query results in databases can be divided into three levels. On the lowest level, we find research on the optimization of query ranking techniques using special database operators, such as the work of Ilyas *et al.* in their paper on rank-aware query optimization [12].

On a higher level, we find research on explicit specification of the way in which tuples are ranked. For example, the work on preferences in databases from Kießling [13] and Koutrika and Ioannidis [14].

On the highest level we find techniques to estimate the ranking of tuples automatically, for example by looking at database and/or access log statistics [2, 5].

The research presented in this paper extends upon the second level as it addresses a way to make explicit, not only the preference of the user, but also the dependency of the preference on contextual factors.

The most closely related research to the work presented is the work of Holland and Kießling [11] on situated preferences. Their approach is founded on preferences as strict partial ordering whereas we picture preferences as scores. Our scores can be expressed as a score-function in their framework but in this paper we will only elaborate and focus on preferences as scores since it allows us to more easily relate preferences to probabilities of choices of the user in the past. Furthermore the two approaches differ in their modeling of situations where they have an ER-based metamodel for context and we chose a model based on Description Logics for reasons we have explicated elsewhere [16].

2. Traditional information retrieval

2.1. An ideal document

To be able to address ranked retrieval in relational databases based on context information, we first look at traditional query based ranking methods and how context information could influence the results of these methods. For this we choose the conceptual model of information retrieval from Berger and Lafferty [3] who in their model generalize the language modeling approach to information retrieval from Ponte and Croft [15]. The main reason why we chose to extend on this approach is because it not only delivers good results for traditional information retrieval but also presents a conceptually simple probabilistic model, that already incorporates some hints on the influence of context information.

Berger and Lafferty imagine that when a user uses an information retrieval system, he starts with an information need. This information need could be presented as a fragment of an "ideal document". The user translates this ideal document fragment (in his head) into a compact query which is posed to the system. The task for the retrieval system is, given the query and a model of how the information need is translated into a query, to retrieve back the most likely document given the query. In other words, to determine for which document d, $P(D{=}d|Q{=}q \wedge U{=}u)$ is the highest. Here $P(D{=}d)$ is the probability that d is the ideal document, $P(Q{=}q)$ is the probability that query q is observed, and $P(U{=}u)$ is the probability that the user is (described by) u. By Bayes' law $P(D{=}d|Q{=}q \wedge U{=}u)$ can be rewritten as:

$$\frac{P(Q{=}q|D{=}d \wedge U{=}u)P(D{=}d|U{=}u)}{P(Q{=}q|U{=}u)} \tag{1}$$

Since the retrieval process should result in a ranked list of documents and the denominator, $P(Q{=}q|U{=}u)$, is for each document the same, it can be ignored, resulting in the following equation consisting of a query-dependent and a query-independent part:

$$\underbrace{P(Q{=}q|D{=}d \wedge U{=}u)}_{query-dependent} \cdot \underbrace{P(D{=}d|U{=}u)}_{query-independent}$$

Berger and Lafferty suggest that the query-independent part can be used to adapt the result to the users needs and interests, but take it to be uniform for all documents. In their paper they focus on the query-dependent part; the probability that query q was posed by the user, given that document d was the ideal document.

2.2. The relevance of a document given its features

To be able to determine the probability $P(Q{=}q|D{=}d \wedge U{=}u)$ we assume that the query and document can be both

represented using features. Suppose function F returns the features for a document:

$$F(d) = \{f_1, \ldots, f_n\}$$

Since it could be the case that we do not know if a document has a certain feature, we can talk about the probability that $F(d)$ returns a certain feature:

$$P(f \in F(d)) \tag{2}$$

Now suppose we assume that the query that the user generates from a document is equal to the set of *all* the features of this document. In this case we could rewrite $P(Q{=}q|D{=}d \wedge U{=}u)$ as:

$$\begin{aligned} &P(Q{=}q|D{=}d \wedge U{=}u) \\ &= P(q{=}F(d)) \\ &= P(\bigwedge_{f \in q} f \in F(d) \wedge \bigwedge_{f \in F(d)} f \in q) \end{aligned}$$

A more realistic assumption would be that the user generates a query from a document by only enumerating *part of* the features of the document. In this case a query could be generated from a document if all features of the query are contained in this document, but they do not have to span the whole document:

$$P(Q{=}q|D{=}d \wedge U{=}u) = P(\bigwedge_{f \in q} f \in F(d))$$

The language modeling approach from Ponte and Croft[15] is concerned with retrieving text documents, so they consider their features to be terms. Furthermore they make two extra assumptions:

- The probability that the user generates a feature from a document is related to the frequency of the term in the document; to model this, they introduce a probability distribution $P(\cdot|d)$ over terms.
- The probability that a certain term is contained in the document is independent of the other terms in the document.

Since we already have a probability distribution for features given a document in equation (2), we assume we can adapt this model to incorporate the frequency of the terms. If we then make the same assumptions as Ponte and Croft we can simplify the calculation of the probability of a query given a document to:

$$\begin{aligned} P(Q{=}q|D{=}d \wedge U{=}u) &= P(\bigwedge_{f \in q} f \in F(d)) \\ &= \prod_{f \in q} P(f \in F(d)) \end{aligned}$$

Given that we have the probability distribution $P(\cdot \in F(d))$ this is enough to rank documents based on a query, which is the core of the language modeling approach to information retrieval.

2.3. The effects of context

Since we not only want to rank documents based on the user query but also based on the context of the user, we shall revisit equation (1) and look how context plays a role in influencing the different terms of the equation. Since the user was already present in the original model and we are only interested in how the context influences the information need for the user of the system, we incorporate the context of the user in the model by considering the user as being *situated* (i.e., having a certain context). We indicate this by replacing u with u_{sit}, by which our new equation becomes:

$$\begin{aligned} &P(D{=}d|Q{=}q \wedge U{=}u_{sit}) = \\ &\qquad \frac{P(Q{=}q|D{=}d \wedge U{=}u_{sit})P(D{=}d|U{=}u_{sit})}{P(Q{=}q|U{=}u_{sit})} \end{aligned} \tag{3}$$

As can be seen, context has its effect in three places. First, it affects the denominator, suggesting that if a user poses a query which is less likely considering its context, the documents matching this result should be considered more relevant. Since this does not affect the ranking of the documents we do not address it in this paper. We imagine, however, that this term could be used to determine the general relevance of query results in a context-aware system, for example to determine how query results should be displayed (since it is important not to flood the user with results).

Second, context affects the "query generation" term $P(q|d \wedge \mathcal{U}_{sit})$, this could be used to account for different input methods under different contexts leading to a different formulation of a query even though the "ideal document" is the same. Notwithstanding the fact that we can for example imagine a user posing different query terms when he uses his phone rather than his desktop PC, we assume in this paper that the influence of context on this term can be neglected compared to the effect of context on another term: the query-independent part.

In the previous section it was assumed that the query-independent part, $P(D{=}d|U{=}u_{sit})$, was the same for each document. However, the essence of context-awareness is that documents have different relevance, depending on the context of the user, which is why we will look in the rest of this paper, how to determine this probability. In other words; we will try to determine the probability of a document being the ideal document (for the user) in a specific context.

3. A model for context-aware relevance

3.1. An ideal document

To be able to calculate the probability of a document being the ideal document in a specific context, we first have to define the notion of an ideal document for a context. If the user did not specify any preferences for documents in contexts our best guess would be to consult the history of the user to determine the ideal document. The thought behind this reasoning is that in many cases the user is not willing to specify preferences for each context, so looking at his history might be the only way to acquire his preferences. Furthermore, this reasoning allows us to integrate preferences, history and uncertainty of context in a natural way.

Considering that both documents and context can be described by features, we define the ideal document for the user in the current context as: *The document which has the same features as documents that the user chose before in contexts with the same features as the current context.*

Since a specific context will never occur more than once, and probably the user will most of the time not be interested in *exactly* the same document as the last time when a similar context occurred, we should be careful how to choose our features and be sure that they are able to generalize well over different documents and contexts. For example, suppose a user watches each workday morning the traffic bulletin. In this case we may assume that on the next workday morning he is again interested in traffic bulletin. The user is, however, probably not interested in watching the same traffic bulletin every day but only in most recent bulletin. Hence, it is important that each recorded traffic bulletin has at least a feature which identifies it as such. How to determine the right kind of features is out of scope for this paper, since this is specific for each retrieval problem, but in many cases features might be provided by data suppliers (e.g., television guides, IMDB) or determined from sensor readings (e.g., localization techniques, activity recognition).

3.2. Finding the most relevant document

The definition of the ideal document above still assumed that, on a high level, a user would each time choose the same kind of document in the same kind of context. In many cases, however, an ideal document cannot exist, simply because the user did not choose a document with the same features each time in the same context. For example, some workday morning the user might decide not to watch the traffic bulletin but watch the weather bulletin instead.

For this reason we define the probability that a document d is the ideal document as *the probability that if we take a random context in the past with the same features as the current context, the user chose a document with the same features as d.* The motivation is that a document would be ideal if it was chosen each time in the same context. If, however, the features of the chosen documents varied, there can exist multiple ideal documents, all with a specific probability. For example, in Figure 1 we graphically displayed an abstraction of the history of a user who watched on workday mornings, in 80% of the cases, programs containing the traffic bulletin and, in 60% of the cases, programs containing the weather bulletin. Now, the probability that on a new workday morning a program which neither contains the traffic bulletin nor the weather bulletin is the ideal program is $(1-0.8) \cdot (1-0.6) = 0.08$.

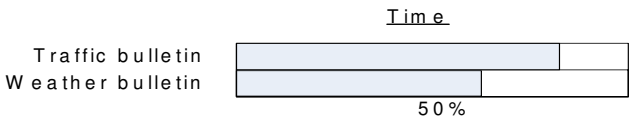

Figure 1. Graphical display of the distribution of video features on a workday morning.

To calculate the probability that this document is ideal we assume that the choices of the different features of a document by the user are independent of each other. This seams reasonable for most cases. An example of an unlikely violation would be that a user likes thriller movies and likes the actor Guy Pearce, but does not like Guy Pearce if he plays in thriller movies. If, however, in a certain domain the dependency of choices is very strong, one could consider designing a model which takes features of the document as context features, but this is out of scope for this paper.

Because we determine the probability that a document is ideal in a certain context based on the features of the context and the document, what we want to calculate is the probability that a document is ideal given the situated user *and* the knowledge that the document can be represented by feature vector $\underline{f}$ and the context can be represented by feature vector $\underline{g}$. For this we introduce, similar to the function F from Section 2.2, also a function G to get from contexts to features.

Now, we can introduce a relation H ("History"), which indicates which document features in past have been chosen in which context; it contains pairs (g, f) with document feature f and context feature g. Furthermore we introduce a score function σ, which gives for each pair (g, f) in H a score, $\in 0..1$. Using this relation, we define the probability that a document is the ideal document in a certain context, given the feature vectors, as follows:

$$P(D{=}d|U{=}u_{sit}, F(d){=}\underline{f}, G(u_{sit}){=}\underline{g}) \tag{4}$$
$$= \prod_{(g,f)\in H} \begin{cases} 1 & \text{, if } g \notin \underline{g} \\ \sigma(g,f) & \text{, if } g \in \underline{g} \wedge f \in \underline{f} \\ 1-\sigma(g,f) & \text{, otherwise} \end{cases}$$

Because of the reasons we explained above, the score function $\sigma(g, f)$ is defined as the probability that if we take a random context in history with feature g, the user chose a document with feature f.

In the example of Figure 1 we assumed that the containment of a traffic and/or weather bulletin were properties of a television program but we could also imagine that they are two disjoint properties of television programs in general; a television program is either a traffic bulletin, or a weather bulletin, or something else. This is an example of a situation where the independence assumption for the choice of the user is clearly violated, since the user *is not able* to choose traffic and weather independently. For dealing with this disjointness we extend the definition $\sigma(g, f)$ as the probability that if we take a random context in history with feature g *and the user was able to choose a document with feature* f *given the other features of the document*, the user actually chose a document with feature f.

Another situation which often happens in context-aware systems is that the user is not so much interested in a single document but in a group of documents. For example, in the case of *TVTouch* a person could choose to watch both the weather and the traffic bulletin at the same workday morning. In this case one should take the whole workday morning as one context where the user chose two documents. Therefore, the features of both documents should be related to this context.

3.3. Incorporating uncertainty of context and document features

Now consider that we are not certain about the features of the documents we have to score and neither we are certain about the contextual features.

For the calculation of the relevance of a document given a situated user, $P(D{=}d|U{=}u_{sit})$, this means looking at all possible combination of features that the current context can have (given the measurements) together with their probabilities. Similarly, we have to look for all possible features that the document can have. This leads to the following computation:

$$
\begin{aligned}
&P(D{=}d|U{=}u_{sit}) \\
&= \sum_{G(u_{sit})=\underline{g}} P(G(u_{sit}){=}\underline{g}) \cdot P(D{=}d|U{=}u_{sit}, G(u_{sit}){=}\underline{g}) \\
&= \sum_{G(u_{sit})=\underline{g}} P(G(u_{sit}){=}\underline{g}) \\
&\quad \cdot \Big(\sum_{F(d)=\underline{f}} P(F(d){=}\underline{f}) \\
&\qquad \cdot P(D{=}d|U{=}u_{sit}, F(d){=}\underline{f}, G(u_{sit}){=}\underline{g})\Big) \\
&= \sum_{G(u_{sit})=\underline{g}} P(G(u_{sit}){=}\underline{g}) \\
&\quad \cdot \Big(\sum_{F(d)=\underline{f}} P(F(d){=}\underline{f}) \\
&\qquad \cdot \prod_{(g,f)\in H} \begin{cases} 1 & \text{, if } g \notin \underline{g} \\ \sigma(g,f) & \text{, if } g \in \underline{g} \wedge f \in \underline{f} \\ 1-\sigma(g,f) & \text{, otherwise} \end{cases} \Big)
\end{aligned}
$$

If we combine this formula with a way to extract features from contexts and documents, and information about their relation (by means of H and σ), we have enough information to score documents based on the context of the user. In the next section we will provide a method to express features of contexts and documents together with a way of providing a relation between them which leads to an example application of our formula.

4. Modeling the relation between context and preferences

4.1. Scored preference rules

The computation method described in the previous section could be used for all kinds of features (in more or less efficient manner). Nevertheless, to be able to deal with structured information in a way that focuses on reasoning and traceability and has close relation to the database world, we choose to model our features using Description Logics (DL) as argued in our previous work [16].

Furthermore, to address the uncertainty of both context and document features we developed in [17] a context uncertainty model. Since, in the scope of our study, we regard correlations and constraints that exist among concept and roles highly desirable (e.g., a person can only be at a single place at one moment), it is important to capture and model these correlations without approximations. For this, we assign each context measurement a probability and a basic event expression. Calculation of the probability of high level context events (e.g., a certain activity) can be done by combining event expressions from measurements attributing to this event as described in [9]. Another advantage of the use of event expressions is that they provide data lineage which could help making the system more traceable.

With this model to represent features of contexts and documents, which allows for the calculation of probabilities, we can define a way to couple context and preferences in a way that is traceable for the user. For this we will use preference rules as we introduced in [16] which consist of a tuple of the form ($\mathcal{C}$ontext,$\mathcal{P}$reference) where both $\mathcal{C}$ontext and $\mathcal{P}$reference are DL concept expressions. However, to be able to incorporate the ideas presented in this paper we

extend the tuple with a score σ. We will call rules of the extended form *scored preference rules*.

As an example consider a scored preference rule $R1$ of a user called Peter:

$$\begin{aligned} \mathcal{C}ontext &: Weekend \\ \mathcal{P}reference &: TvProgram \sqcap \\ &\quad (\exists hasGenre.\{HUMAN\text{-}INTEREST\}) \\ \sigma &: 0.8 \end{aligned}$$

The semantics of the score σ is exactly the same as the function $\sigma(g, f)$ in Section 3.2. In other words, the semantics of this rule is that the probability that whenever we take a random context in the past during the weekend, if Peter was able to choose a television program on human interest, the chance that in this context he would actually choose a television program on human interest was 0.8.

The idea behind the scored preference rules is that they form an abstraction/generalization of the history of the user. They could really be mined from the history but also defined by the user or system designer. Together, they can form the basis for a context-aware application. And, because the semantics of the scoring function for all rules is the same as defined above, they provide an explanatory way for scoring documents.

To determine a score for a document based on the available preference rules we consider only those features important for relevance that are mentioned in the preference rules. In this way, our relation H and score function σ, as introduced in Section 3.2, are completely defined by the preference rules and we could, if we are certain about all features, apply equation (4) to calculate the "abstracted" probability that a document is ideal in a certain context.

Equation (4) however, only provides useful results if there is at least one rule in H applicable in the current context (otherwise it results in a score of 1). For this reason, in case the context of querying is not covered by any preference rule, the retrieval system is unable to return any meaningfull probability for the relevance of the query answers. As a solution, one could cover these situations by having "default" preference rules, which are valid in any context.

4.2. An example

In this section we will provide an example of how the scored preference rules can lead to a probability per document.

Suppose our *TVTouch* system has, next to the rule $R1$ as defined in Section 4.1, a second rule $R2$ which states that, if Peter is having breakfast, he prefers to watch television programs about the news:

$$\begin{aligned} \mathcal{C}ontext &: Breakfast \\ \mathcal{P}reference &: TvProgram \sqcap \\ &\quad (\exists hasSubject.\{News\}) \\ \sigma &: 0.9 \end{aligned}$$

Now, suppose we have to score the four television programs in Table 1 when the context is that the user is having breakfast during the weekend. For simplicity, we assume that the context is certain (and hence the factor $\sum_{G(u_{sit})=\underline{g}}$ in the formula of Section 3.3 equals 1 and will be left out below). Furthermore we assume that features of documents are independent so we are able to construct all probabilities from Table 1.

In this case, based the formula from Section 3.3, the probability that *Channel 5 news* is the ideal document is:

$$\begin{aligned} &\sum_{F(d)=\underline{f}} P(F(d)=\underline{f}) \\ &\qquad \cdot \prod_{(g,f)\in H} \begin{cases} 1 & \text{, if } g \notin \underline{g} \\ \sigma(g,f) & \text{, if } g \in \underline{g} \wedge f \in \underline{f} \\ 1-\sigma(g,f) & \text{, otherwise} \end{cases} \\ &= P(F(d)=\{\text{Humaninterest, weather}\})R1_\sigma R2_\sigma \\ &\quad + P(F(d)=\{\text{Humaninterest}\})R1_\sigma(1-R2_\sigma) \\ &\quad + P(F(d)=\{\text{weather}\})(1-R1_\sigma)R2_\sigma \\ &\quad + P(F(d)=\{\})(1-R1_\sigma)(1-R2_\sigma) \\ &= (0.95\cdot 0.85)\cdot(0.8\cdot 0.9) + (0.95\cdot 0.15)\cdot(0.8\cdot 0.1) \\ &\quad + (0.05\cdot 0.85)\cdot(0.2\cdot 0.9) + (0.05\cdot 0.15)\cdot(0.2\cdot 0.1) \\ &= 0.6006 \end{aligned}$$

Similarly, we can calculate the probabilities for the other documents. This results in the following probabilities for them being the ideal document:

Channel 5 news		$= 0.6006$
Oprah	$(0.85\cdot 1)\cdot(0.8\cdot 0.1)$	
	$+(0.15\cdot 1)\cdot(0.2\cdot 0.1)$	$= 0.071$
BBC news	$(1\cdot 1)\cdot(0.2\cdot 0.9)$	$= 0.18$
MPFS	$(1\cdot 1)\cdot(0.2\cdot 0.1)$	$= 0.02$

If, without considering context, all documents where equally relevant to the user's query, these scores also determine their context-aware ranking.

5. Naive implementation

To show how our ideas could be used in practice, we implemented a naive system that is able to rank results based on the preference rules of the user. To do this we extended PostgreSQL with a datatype for event expressions. Furthermore, since the basic elements of Description Logics are concepts and roles, we view each concept as a table, which

Program	Genre	Probability	Subject	Probability
Oprah	human interest	0.85	-	-
BBC news	-	-	weather bulletin	1.0
Channel 5 news	human interest	0.95	weather bulletin	0.85
Monty Python's Flying Circus	-	-	-	

Table 1. Available television programs.

uses the concept name as the table name and has an *ID* attribute and an event expression attribute. Similarly, we view each role as a table, with the role name as its table name and containing three attributes; *SOURCE*, *DESTINATION*, and an event expression. For each tuple of the table, the role relates the *SOURCE* individual with the *DESTINATION* individual. We adapted the approach of [4] to express DL concept expressions using SQL queries and added support for the propagation of event expressions.

An important remark here is that we provide a uniform tabular view towards both static and dynamic contexts, even though the later (e.g., location, surrounding people, etc.) must be acquired real-time from external sources/services like sensor networks. This is in line with the efforts of the sensornet community which has embraced declarative queries as a key programming paradigm for large sets of sensors [7]. Here, we take the SQL query language as a uniform interface to the contexts. Another reason for doing this is that we can later use the context history for analysis purposes to achieve smartness [1].

With the mapping mechanism introduced in [16], we can construct a database view for each concept expression containing all tuples that are included in the concept expression, together with an event expression as a measure of the probability by which they are included. Since both context and preference in a preference rule are concept expressions we can both represent them as views on the database.

All preference rules together are stored as rows in a repository table consisting of the name of the preference view, the name of the context view, and the score of the rule.

Furthermore, to calculate the probability $P(D{=}d|U{=}u_{sit})$ for each tuple, we use the formula from Section 3.3 to provide a big preference view. This view contains all preferred tuples together with the probabilities that they are ideal based on the current context and preference rules in the repository. The nice part of having such a view is that, as the current context develops, the probabilities of containment of tuples in the view changes accordingly. To get an idea of the complexity, one can look at our manual calculation in Section 4.2, which even was a simplification since context was considered to be certain and document features independent.

Finally we have to adapt the query results of the user by ordering the tuples in the result, based on the probability from the big preference view. This is done by doing a union of the preference view and the results of query of the user, where the results are ordered by the probabilities in the preference view. This step brings us back to the integration of contextual relevance and query relevance from which we started in Section 2.3, where in this naive approach, the probability of the query-dependent part is either 1, if the tuple was contained in the user query, or 0 if it was not.

We have to make one important remark for this implementation. Since for each new rule, both the amount of possible combinations of context features and the amount of possible combination of tuple features (for which the probability should be calculated) are doubled, this leads to highly exponential query times. To verify this we generated a test database of context and documents containing around 11000 tuples; around 1000 persons, 300 TV programs, 12 genres, 6 subjects, 4 activities, 5 rooms and their relations. We created a series of rules on this test database where we measured query times for an increasing number of rules. Our results confirm our suspicion; for one till four rules, query times are still acceptable (query time less than 1 second). Five to six rules take 4-20 seconds, but as we we arrive at seven rules, our query did not finish within half an hour. For this reason our implementation shows, next to the applicability of the approach, also its current bottleneck.

6. Discussions

This paper covers a first step towards a ranking method for query results based on context information. As became clear from the previous sections, there are still many challenges to overcome. In this section we will highlight the ones which we think are the most salient.

- *Performance:* If anything became clear from the implementation, it is that the naive implementation will not scale to a useful number of rules. There are possible ways to address this challenge, if we can prune the amount of applicable rules and candidate documents in early stages.

- *Evaluation of ranking:* Next to an improvement in speed, it should be tested whether the assumptions we made about the preferences of the user hold in practice by conducting user studies. An interesting area

to explore in this light, would be the weighting of the query-independent and query-dependent part of equation (3), using smoothing methods.

- *Explanation of results:* One of the reasons for the introduction of preference rules based on Description Logics, was to make our approach traceable (i.e., to make it easy for the user to understand the choices of the system). To put this into practice we should preferably not require the user to look at the preference rules himself, but provide the user with a motivation for the "context based" answer. An interesting research area would be to determine what kind of explanation (such as rules, features, or scores) would give the user a good insight to the system while still being unobtrusive. Related research in this area is the ordering of attributes of query results by Das *et al.* [6].

- *Mining/learning preferences:* Since our scoring method gives a semantics for the score of a document based on the history of user choices, a legitimate question to ask is, how well the actual user preferences would be predicted by mining the history of the user using exactly these semantics. Related research in the area of mining preferences can be found in [10].

- *Modeling multiple users:* In our case we only dealt with the preferences of one user. In some cases we might have to deal with ranking results for multiple users (for example if multiple users want to watch TV together). We conjecture that this could be naturally addressed with the model presented here, but this is still open for research.

7. Conclusions

We have presented a novel explanatory approach of looking at context-aware relevance by defining the context-aware relevance of features as a probabilistic function of past choices.

We showed that this approach goes well together with traditional probabilistic information retrieval and uncertainty of context information and demonstrated a naive implementation.

The main outstanding issues are the improvement of the scalability of the approach and the measurement of the effect on retrieval performance. We believe that these issues can be solved and will lead to an approach to context-aware ranking that is both explanatory and results in answers that better serve the information needs of users.

References

[1] Echise. first international workshop on exploiting context histories in smart environments, May 2005.

[2] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *CIDR*, 2003.

[3] A. L. Berger and J. D. Lafferty. Information retrieval as statistical translation. In *SIGIR '99*, pages 222–229. ACM, 1999.

[4] A. Borgida and R. J. Brachman. Loading data into description reasoners. In *SIGMOD '93*, pages 217–226, New York, NY, USA, 1993. ACM Press.

[5] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *VLDB '04*, pages 888–899. Morgan Kaufmann, 2004.

[6] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. In *SIGMOD '06*, pages 395–406. ACM, 2006.

[7] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB '04*, pages 588–599, 2004.

[8] L. Feng, P. M. G. Apers, and W. Jonker. Towards context-aware data management for ambient intelligence. In *DEXA '04*, pages 422–431, 2004.

[9] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[10] S. Holland, M. Ester, and W. Kießling. Preference mining: A novel approach on mining user preferences for personalized applications. In *Knowledge Discovery in Databases: PKDD 2003*, pages 204–216. Springer, 2003.

[11] S. Holland and W. Kießling. Situated preferences and preference repositories for personalized database applications. In *23rd International Conference on Conceptual Modeling (ER 2004)*, pages 511–523. Springer-Verlag, November 2004.

[12] I. F. Ilyas, R. Shah, W. G. Aref, J. S. Vitter, and A. K. Elmagarmid. Rank-aware query optimization. In *SIGMOD '04*, pages 203–214. ACM, 2004.

[13] W. Kießling. Foundations of preferences in database systems. In *VLDB '04*, pages 311–322, 2002.

[14] G. Koutrika and Y. E. Ioannidis. Personalized queries under a generalized preference model. In *ICDE '05*, pages 841–852. IEEE Computer Society, 2005.

[15] J. M. Ponte and B. W. Croft. A language modeling approach to information retrieval. In *SIGIR '98*, pages 275–281. ACM, 1998.

[16] A. H. van Bunningen, L. Feng, and P. M. G. Apers. A context-aware preference model for database querying in an ambient intelligent environment. In *DEXA '06*, pages 33–43. Springer, 2006.

[17] A. H. van Bunningen, L. Feng, and P. M. G. Apers. Modeling Uncertain Context Information via Event Probabilities. In *Proceedings of the 2nd Twente Data Management Workshop on Uncertainty in Databases*, pages 25–32. CTIT, 2006.

[18] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.