Using Bayesian Networks To Infer Product Rankings From User Needs

Sven Radde and Burkhard Freitag

Institute for Information Systems and Software Technology
University of Passau, 94030 Passau, Germany
{sven.radde,burkhard.freitag}@uni-passau.de
http://www.ifis.uni-passau.de/

Abstract. Customers are commonly not able to provide preferences that are technical enough to be used in the internal algorithms of knowledge-based recommender systems. In this paper, we present an approach to use a Bayesian network to infer technical preferences from customer answers obtained through a conversational elicitation process. The inferred preferences can be used in conjunction with a variety of common database ranking technologies to generate product recommendations.

Key words: Bayesian inference, ranking databases

1 Introduction

When recommending complex products to customers, a knowledge-based recommender system has to base its reasoning on the mostly technical product properties that can be obtained using datasheets or similar sources of information. To this end, several well-known techniques exist, such as database ranking based on multi-attribute utility-theory or preference-based databases. However, as a prerequisite for applying these techniques, customers would need to be able to specify their wishes in technical terms. Experience reveals that this is far more than what can be expected from the average customer, in particular for complex technical product damains.

In this paper, we present a way to elicit preferences from customers by asking them "soft" questions about their needs and expectations. From their answers, preferences that serve as inputs to the afore-mentioned recommendation algorithms can be inferred by using a Bayesian network. We describe a utility-based approach to obtain product recommendations and provide a method to use the inferred knowledge as pareto-preferences.

The rest of this paper is organized as follows:

In section 2, we present a use case and describe the requirements arising from it. The process of inferring utility values for technical product attributes from customer answers is detailed in section 3. In section 4, we elaborate on how to make use of the inferred knowledge by employing a MAUT-based approach as well as other techniques such as pareto-based preferences. We review some related work in section 5 before concluding in section 6.

2 Use Case

Today's cellphones are complex technical devices, fulfilling a wide range of functions beyond the classic "make a phone call away from home" scenario, such as navigation aid, MP3 player, digital camera, web / messaging client and many more. Very few customers are able to make completely well-informed buying-decisions in this field without a significant need for consultation. The situation is made even worse by the frequently changing product domain. Retailers are forced to permanently release new products to expand their share in a rather saturated market. Also, the domain itself sees frequent technical innovations, creating new products and sometimes even new business models (think of, e.g., location-based services made possible by the recent integration of GPS receivers into many mobile phones). Therefore, even once-acquired domain knowledge ages rapidly, for customers and salespersons alike.

Cellphones differentiate themselves primarily through their *technical* properties, whereas customers' whishes commonly consist of "softer" *needs* that the desired phone is supposed to satisfy. Salespersons bridge this discrepancy in a natural way, translating the customers' wishes into appropriate technical requirements before recommending cellphones based on these requirements.

However, many current online-recommenders for mobile phones, despite being aimed at end-users, e.g., in web stores, cannot handle this abstraction. Commonly, many online applications are merely product configurators that require their users to specify technical constraints which are then used to restrict the set of available products accordingly. To bridge the gap between a customer's wishes and technically evaluable preferences, an electronic recommender system must be prepared to accept fuzzy user input and use an internal model to infer the technical criteria. At the same time, the inference mechanism must be flexible enough to be used in a dynamic dialogue environment, i.e., it cannot rely on a particular order answers are given in. Some questions may well remain unanswered throughout the dialogue, and for some questions the customers may change their opinion at a later point in the dialogue.

3 Eliciting Preferences

Our approach to infer technically useful information from customer answers is based on a Bayesian network (cf., e.g., chapter 14 in [12]) which is automatically generated from a metamodel-based description of the targeted market domain. Bayesian networks can be applied to model causal relationships between observations and their possible effects based on conditional probabilities. Once having evidence of (part of) the observations, a Bayes engine can infer the probabilities of the possible outcomes by probability propagation through the network. In the work reported on here, we use a Bayesian network to model the relationships of user needs (observations) and technical properties of products (outcomes).

Fig. 1 shows a UML diagram of a portion of the domain metamodel which is relevant to the process of eliciting preferences. The *Articles* in the targetted

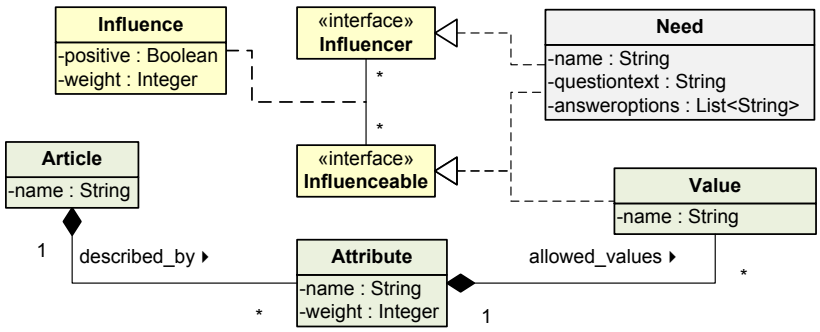

Fig. 1. UML diagram of the domain metamodel

domain are described by an explicit aggregation of their technical *Attributes*. To be able to distinguish the relative importances of *Attributes* during the recommendation process, they are assigned a numerical weight (cf. definition 4 for a detailed explanation). Also, each *Attribute* explicitly lists its allowed *Values*, defining the possible characteristics of the products in the target domain. Recommendations will then be obtained based on the modeled description of the technical properties of the target domain. The implied restriction to finite, discrete value ranges for *Attributes* has proven to be not a significant limitation, since, for our purpose, the value ranges of typically continuous *Attributes* (e.g., prices or sizes) can easily be classified into discrete ranges.

The dialogue used to obtain information from the customer is specified by modelling a number of *Needs*. The *Need* entity contains properties that are used to formulate a specific question that can be asked during the course of the elicitation process, including the possible answer options.

Connecting the answers about *Needs* with the desired technical product properties that are required to obtain recommendations is accomplished by modelling a number of *Influences* that represent the causal interdependencies between *Needs* and other *Needs*, as well as between *Needs* and attribute *Values*. *Influences* can be positive, i.e., have a strengthening effect, or not, i.e., have an attenuating effect. The numerical weight of an *Influence* denotes the "strength" of the *Influence* as we will detail further below.

Linking *Needs* with each other by means of an *Influence* is based on the notion that answers to questions regarding one *Need* may influence answers that the customer is likely to give when asked questions about other *Needs* in the model, as can be observed in example 1. *Influences* between *Needs* and *Values* model the influence an answer to a question has on the perceived utility of a product, as we will show in definition 1.

Note that the diagram in Fig. 1 displays a *metamodel*, which must be instantiated to a domain model with concrete *Needs*, *Attributes*, and *Influences*. Examples 1, 2, and 3 will illustrate this process for a simplified mobile communications domain.

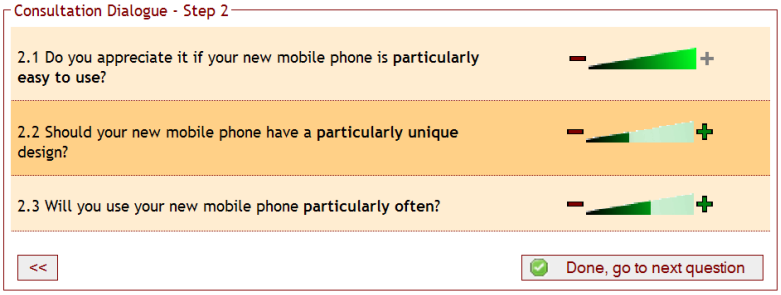

Fig. 2. Screenshot of the dialogue in our web application

Example 1 (Needs). Let us consider a simplified example from the mobile communications domain where the metamodel could have been instantiated by specifying the following exemplary *Needs*:

Need name	**Question text**
	"Will you use your new phone to...
`multimedia`	...play multimedia content?"
`business`	...use business functions?"
`music`	...listen to music?"
`internet`	...access the internet?"

It is immediately apparent that causal influences exist between some of these *Needs*, which can be represented using *Influences* (cf. Fig. 1), as we show below.

For many questions it turns out in practice that they should be written in a way so that they can be answered by customers on a Likert scale [9] (i.e., on a scale between "strongly agree" and "strongly disagree"). See Fig. 2 for a screenshot that illustrates how questions are displayed in a web application that implements the approach described in this paper.

Example 2 (Attributes and Values). A current real-life model instantiation for the mobile telecommunications domain comprises about 70 technical *Attributes* with about 500 possible attribute *Values*. In this paper, we focus on three exemplary attributes and their possible values: Does the cellphone have an integrated MP3 player, does it have broadband internet connectivity via UMTS, and what size has its internal memory?

Attribute	**Allowed values**	**weight**
`mp3`	`yes` / `no`	2
`umts`	`yes` / `no`	1
`memory`	`small` / `medium` / `large`	1

While `mp3` and `umts` are examples of boolean value ranges, `memory` illustrates a discretized value range where the actual amount of memory has been mapped to discrete values (using, e.g., <1GB / 1-4GB / >4GB as a classification rule).

Table 1. Exemplary Influences

from	**to**	**type**
business	internet	positive
business	Memory_small	negative
business	Memory_medium	positive
internet	umts_yes	positive
internet	umts_no	negative
multimedia	music	positive
multimedia	Memory_medium	negative
multimedia	Memory_large	positive
music	mp3_yes	positive
music	mp3_no	negative

Example 3 (Influences). Let our sample domain model contain *Influences* as shown in table 1 connecting the model elements of examples 1 and 2. For simplicity, assume further that all of them have an equal weight of 1.

Using *Influences*, arbitrary hierarchies of *Needs* may be specified, although a real-life evaluation revealed that the hierarchy remains rather flat in the considered use case. In fact, there are merely three tiers, i.e., one tier of *Needs* that influence other *Needs* on a middle tier which, in turn, influence the technical *Attributes* that form the bottom tier.

From an instance of the domain metamodel, a corresponding Bayesian network can be automatically generated. *Needs* and attribute *Values* are represented as random variables (i.e., nodes of the graph representing the network). *Need*-variables have outcomes that correspond to the possible answers of the corresponding questions in the elicitations dialogue, so that a customer's answers can easily be represented in the Bayes net by introducing evidence for the appropriate outcomes. On the other hand, each *Value* is represented by a random variable with the outcomes of "*true*" and "*false*" that model the likelihood that a certain *Value* fulfills the customer's *Needs*. Modelling *Values* in this way (and not, e.g., by representing one *Attribute* as a single variable with all *Values* as possible outcomes) is necessary to model the fact that *Values* may be satisfying independently of each other.

The *Influences* form the edges of the Bayesian network. Modeled causal dependencies between source nodes (*Influencers*) and a target node (*Influenceable*) are expressed by the entries in the conditional probability table (CPT) of the target node. The CPTs are constructed in such a way that, for positive *Influences*, if the customer agreed to the question corresponding to the source's *Need*, the likelihood that he/she will also agree to the target *Need* is increased, weighted relatively against other *Influences* by using the specified weight. If the target node corresponds to a *Value*, the probability for "*true*" of that particular *Value* is increased (i.e., it is more likely that products with this attribute value will be acceptable for the customer). Negative *Influences* have the opposite effects.

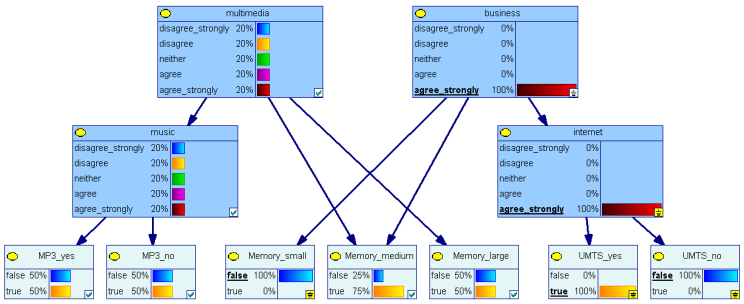

Fig. 3. Inference snapshot (modeled with GeNIe – http://dsl.sis.pitt.edu/)

For a detailed description of the overall generation method see [10]. Fig. 3 shows the Bayes net for the domain model as introduced in examples 1, 2, and 3. The main purpose of the Bayesian network is to derive utility estimations for attribute *Values* from which preferences will be derived. To this end, the notion of utility introduced below is based on the assumption that *Values* which are more likely to satisfy the customer's needs are also more useful (i.e., should have a greater utility).

Definition 1 (Utility of a Value). *We define the* utility u_{av} *of an attribute Value av as the posteriori-probability of the outcome "true" of the random variable* r_{av} *in the Bayesian network that corresponds to av:*

$$u_{av} := p(r_{av} = true \mid \ldots)$$

It immediately follows that all utility values are normalized between 0 *and* 1*. Also, the advantage of representing attribute Values as separate booleans becomes apparent: It is reasonable that more than one Value for an Attribute can get a* utility *of 1.0 (i.e., is useful with 100% probability). This situation can be taken into account in a Bayes net as described above.*

Example 4 (Utility). Consider an (early) point in the preference elicitation dialogue where the customer has given the answer "I strongly agree." to the question for the *Need* `business`. The answer is entered as evidence into the Bayesian network, leading to the posteriori probability-distributions shown in Fig. 3. Evaluating the *Influences* as described above, the elicitation algorithm infers that the customer will need to access the internet using the cellphone which makes UMTS connectivity a necessity, i.e., we have $u_{umts\_yes} = 1.0$ and $u_{umts\_no} = 0.0$

On the other hand, needing business functionality does not allow any conclusions towards multimedia support, which leads to ambivalent utilities for an MP3 player:

$$u_{mp3\_yes} = u_{mp3\_no} = 0.5$$

As for the memory capacity, the system reasons that a too small capacity is inacceptable in a business context, while a medium capacity is probably sufficient (depending on whether additional storage is required for multimedia content).

As we have seen, the primary goal of the inference engine in our context is to infer technical values from user needs according to the influence relations and given conditional probabilities. However, it can also process direct user input to allow for short-cuts by simply inserting a submitted value as evidence for the node of the Bayesian network that corresponds to the appropriate *Value*-entity. Thus, the preference elicitation dialogue is inherently adaptable to different levels of customer expertise. Further uses of the described inference approach, e.g., to control the dialogue flow by estimating the importance of unanswered Needs and to predict a customer's answers are presented in a broader context in [11].

4 Using Preferences

As shown above, the Bayesian network provides us with estimations about the utility of every possible attribute value in the product domain. To use these estimations to elicit preferences, we also need a notion of the relative importance of the attributes relative to each other. In our approach, the importance of an attribute depends on three factors:

(1) Those attributes customers show significant interest in should be regarded as more important than others. In our model, "significant interest" is derived from the fact that more *distinctive* predictions for the attribute values exist.

(2) The dialogue *situation* has influence on the importance of an attribute for the elicitation process. Attributes that are not connected to any already answered question, should not have any influence at all.

(3) A domain expert may assign a static numerical *weight* to each attribute (cf. Fig. 1). Marketing research shows that some attributes are inherently more important than others in a buying decision. Experiments indicate that it is sufficient to classify attributes into a small number of weight "classes", which is considered to be rather simple for suitably knowledgeable experts.

Definition 2 (Distinctiveness of an Attribute). *The* Distinctiveness d_a *of Attribute a is defined as the average of the distances of the utilities of all possible attribute values of a from the "indifferent" utility of* 0.5*, normalized to* [0..1]*:*

$$d_a := 2 * \frac{\sum_{v \in dom(a)}(|u_v - 0.5|)}{|dom(a)|}$$

Example 5 (Distinctiveness). To calculate the distinctiveness d_{memory} of the attribute "memory", assume that the following utilities have been inferred (cf. Fig. 3):

$$\begin{aligned} u_{memory\_small} &= 0.0 \\ u_{memory\_medium} &= 0.75 \\ u_{memory\_large} &= 0.5 \end{aligned}$$

$$d_{memory} = 2 * \frac{|0.0-0.5|+|0.75-0.5|+|0.5-0.5|}{3} = 0.5$$

The result fits our intuition: We are not yet sure about the customer's opinion regarding memory size at this point of the dialogue. Therefore, values derived for this attribute should not be taken too seriously.

Definition 3 (Situation Factor of an Attribute). *Let $Q_{answered}$ be the set of all questions already answered in the current dialogue. For an Attribute a let $P(a)$ denote the set of all Influence-ancestors of a, i.e., the Needs connected directly or transitively with a Value of a via an Influence-path in the network. The* Situation factor s_a *of a is defined as follows:*

$$s_a := \begin{cases} 0 \text{ if } \forall q \in Q_{answered} : q \notin P(a) \\ 1 \text{ otherwise} \end{cases}$$

Definition 4 (Importance of an Attribute). *Let a be an Attribute and w_a be the weight of a as specified in the particular model under consideration. Let d_a be the distinctiveness of a according to definition 2 and s_a be the situation factor of a (definition 3). The* Importance i_a *of a is defined as*

$$i_a := d_a * s_a * w_a$$

Example 6 (Importance). Extending example 5, we determine i_{memory} based on the following parameters:

$d_{memory} = 0.5$ (*example 5*)
$s_{memory} = 1.0$ (*memory is connected to the answered question, Fig. 3*)
$w_{memory} = 2.0$ (*taken from the domainmodel*)

$i_{memory} = 0.5 * 1.0 * 2.0 = 1.0$

Note that distinctiveness and situation factor depend on the preferences learned during the course of the elicitation dialogue. Therefore, they are updated after each dialogue step to account for newly acquired knowledge.

The approach to preference elicitation described here is based on Multi-Attribute Utility-Theory (MAUT – cf., e.g., [13]) which is used to combine the utility estimations of the various technical attributes into one global ordering of the whole product catalogue. To achieve this, we formulate a utility function for products which essentially computes a weighted sum over the utilities of all technical attributes.

The utility function can be formulated as a standard SQL query. For the sake of simplicity, we assume that all relevant data for an article is available in a single table with one column for each technical attribute (cf. Fig. 1) and an additional column containing the product's name. Every tuple in this table represents one concrete product (e.g., in the sample domain, one concrete cellphone). The following `SELECT` query computes a numerical `UTILITY` value for each tuple and orders the answer set accordingly:

```
SELECT   *, ($utilityfunction) AS UTILITY
FROM     cellphones
ORDER BY UTILITY DESC
```

`$utilityfunction` calculates the overall utility of a given cellphone by summing up the utility of each attribute value u_{value} (cf. definition 1), weighted by that attribute's importance $i_{attribute}$ (cf. definition 4). To this end, our implementation uses a set of `CASE-WHEN` clauses for each attribute.

Table 2. Exemplary Product Catalogue

Name	**mp3**	**memory**	**umts**
MobileA	yes	medium	no
MobileB	yes	large	no
MobileC	no	medium	yes
MobileD	no	small	no

Example 7 (Preference Order by MAUT). Assume that the current product catalogue contains the entries as shown in table 2. The utility and distinctiveness values are derived from the dialogue situation shown in Fig. 3:

$u_{umts\_yes} = 1.0$
$u_{umts\_no} = 0.0 \quad d_{umts} = 1.0$
$u_{memory\_small} = 0.0$
$u_{memory\_medium} = 0.75$
$u_{memory\_large} = 0.5 \quad d_{memory} = 0.5$
$u_{mp3\_yes} = 0.5$
$u_{mp3\_no} = 0.5 \quad d_{mp3} = 0.0$

For simplicity, assume situation factors of 1.0 for all attributes. For this constellation, the following ranking of catalogue entries according to the utility values computed as shown above results:
1) MobileC (Utility: `1.375`)
2) MobileA (Utility: `0.375`)
3) MobileB (Utility: `0.25`)
4) MobileD (Utility: `0.0`)

Since UMTS capability is the most important feature for our sample business customer, it is decisive in producing the utility-based rank ("MobileC" is the only UMTS-capable device in table 2). In contrast, support for MP3 does not play a role since the course of the dialogue did not yet allow any conclusions about the customer's wishes in this respect.

The structure of the query is known at the time of domain model design. Therefore, the query can be implemented as a stored procedure that can be called with the current importances and utility values as parameters. In our experiments, the actual query execution times were only a few millliseconds. It should be noted, however, that there are less than 1,000 different cellphones on the market today, leading to a rather small product catalogue and thus a small domain size. It is worth noting that, in general, the calculated utility for a product does not have an absolute meaning (i.e., a statement about *how* useful the product is, cannot be made). The value can only be interpreted in a relative way, i.e., a higher utility means greater usefulness and hence a higher rank.

The method described so far can also be used to provide the necessary inputs for more general approaches to rank query answers using boolean predicates, such as the one presented in [2]. To take full advantage of the more elaborate possibilites offered there, it would be necessary to extend our product modelling

by ways to express the potentially hierarchical structure of complex boolean conditions. Also, preferences for approaches based on pareto-optimality such as PreferenceSQL [7, 8] can be derived using a Bayesian network as described in this paper. In effect, the Bayesian network provides an ordering for all values of a technical attribute by interpreting the calculated utility values as the pareto-preferences serving as an input for PreferenceSQL.

PreferenceSQL supports "`LAYERED`" preferences (cf. [7]) for situations where a domain $dom(A)$ of an attribute can be partitioned into subsets that are ordered according to a "better than" relation. In our approach, all attribute values that have the same utility are grouped together in the same "layer", leading to a straight-forward application of the `LAYERED` preference constructor. Clustering techniques may be used to limit the number of subsets that are to be considered by interpreting some values as equally preferred, despite minimal differences in their numerial utility values.

Since the semantics of our approach relies on the notion that the customer is generally indifferent about attribute values with the same utility (i.e., all values with the same value are mutually substitutable), we annotate each `LAYERED` preference with the additional "`regular`" keyword (cf. section 4 in [7]). These preferences are combined using the pareto "`AND`" operator to form the complete PreferenceSQL query.

Example 8 (PreferenceSQL). We re-use the dialogue situation of example 7 to formulate a query in PreferenceSQL. Using the pattern described above leads to the following statement:

```
SELECT * FROM cellphones PREFERRING
umts   LAYERED (('yes'), ('no'), others) regular                  AND
memory LAYERED (('medium'), ('large'), ('small'), others) regular AND
mp3    LAYERED (('yes', 'no'), others) regular
```

Executing this statement against the product catalogue of table 2 yields "*MobileC*" as the query result, which is the pareto-optimal tuple of the relation and therefore the only result according to the "Best-Matches-Only" semantics of PreferenceSQL. Successively re-executing the query with added `WHERE`-clauses to exclude the already-retrieved tuples yields the following results which are ordered exactly as those obtained using our MAUT-approach in example 7 (NB: in general, the orderings defined by both approaches are not equivalent):

1) MobileC
2) MobileA (`WHERE name <> 'MobileC'`)
3) MobileB (`AND name <> 'MobileA'`)
4) MobileD (`AND name <> 'MobileB'`)

Generated in this straight-forward way, the PreferenceSQL query would consist of a very large number of pareto-preferences. To reduce the number of preferences, the importance of an attribute (cf. definition 4) may be exploited to limit the preferences to a fixed number or to consider only preferences for attributes having an inportance that exceeds a certain threshold.

5 Related Work

The knowledge-based elicitation approach described here is notably different from collaborative filtering methods (cf., e.g., [4, 6]) since it does not require item ratings. Assuming that a user will not interact with the system very frequently, we cannot rely on buying histories to build our model of the user. While eliciting explicit ratings may be acceptable in an online context, it seems not to be an adequate form of interaction between salespersons and customers. Hence, our user-model builds on preferences that can be elicited during the course of a natural sales dialogue.

Ardissono et al. [1] present a personalized recommender system for configurable products. Their approach involves preference elicitation techniques that employ reasoning about customer profiles to tailor the dialogue to a particular customer by providing explanations and smartly chosen default values wherever possible. The customer preferences learned this way are then used as constraints in the configuration problem at hand to generate the recommended product configuration, which might result in empty recommendations (i.e., the specified constraints are not satisfyable), requiring repair actions. Our approach does not directly exploit the elicited preferences as constraints but rather uses them as an input to ranking database queries which return a list of products ordered according to the customer's preferences.

In [3], the dialogue-oriented recommender suite CWAdvisor is presented. Their knowledge-base is similar to ours but it includes an explicit representation of the "recommender process definition", i.e. all possible dialogue paths in a tree-like structure. While obviously able to specify a fine-grained dialogue, the achievable level of detail is limited by the complexity of the dialogue specification. Our approach generates the (equally complex) dialogue specification from a much more compact model and is more flexible by incorporating mixed-initiative selection of questions, easy belief revision and adaptive personalization.

An approach similar to ours is presented in [5]. However, the utility estimations (the "value tree") of Jameson et al. do not seem to be built on an explicit model of the currently served customer but rather on the assumed properties of an average user of their system. Hence, the derived preferences are not personalized as strongly as in our approach. Also, as the value tree is a strictly hierarchical structure, it cannot capture the fact that a technical attribute may be influenced by more than one single need.

6 Conclusion

We presented an approach to inferring utility estimations usable for preference-based or ranking-based database queries, which is accomplished by using a Bayesian network that can be generated from a domain model. The approach described in this paper has been implemented in a joint project together with an industry partner. Their business experts designed a domain model comprising about 25 needs and more than 100 technical attributes. The model is considered

to adequately represent the marketing-relevant aspects of the mobile communications domain from the industry point of view. Although the effort for this first real-life instantiation of the model was significant, the task was regarded as feasible and the routine model maintenance has turned out to be inexpensive under operational conditions. Evaluations of the overall recommendation concept by marketing research experts have shown convincing results concerning the adequacy of the model and the acceptance of the overall approach.

Our current work focuses on conducting a thorough field evaluation of the system with a larger number of volunteers to obtain more statistical evidence about the quality of the derived preferences and related product rankings. Another field of ongoing research is the design of an explanation component which is able to explain the reasoning that led to the generated recommendations and to exploit user feedback about the recommendations to automatically adjust some parts of the domain model.

References

1. Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schaefer, R., Zanker, M.: A Framework for the Development of personalized, distributed Web-Based Configuration Systems. AI Magazine 24, 93–110 (2003)
2. Beck, M., Freitag, B.: Weighted Boolean Conditions for Ranking. In: Proc. of the ICDE-08 Workshop on Ranking in Databases (DBRank'08) (2008)
3. Felfernig, A., Friedrich, G., Jannach, D., Zanker, M.: An Integrated environment for the Development of Knowledge-Based Recommender Applications. Intl. Journal of Electronic Commerce 11(2), 11–34 (2006)
4. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS) 22(1), 5–53 (2004)
5. Jameson, A., Schaefer, R., Simons, J., Weis, T.: Adaptive Provision of Evaluation-Oriented Information: Tasks and Techniques. In: Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI) (1995)
6. Jin, R., Si, L., Zhang, C.: A Study of Mixture Models for Collaborative Filtering. Information Retrieval 9(3), 357–382 (2006)
7. Kießling, W.: Preference Queries with SV-Semantics. In: Proc. of the 11th International Conference on Management of Data (COMMAD). pp. 15–26. Computer Society of India (2005)
8. Kießling, W., Köstler, G.: Preference SQL – Design, Implementation, Experiences. In: Proc. of the 28th Intl. Conference on Very Large Data Bases (VLDB) (2002)
9. Likert, R.: A Technique for the Measurement of Attitudes. Archives of Psychology 22(140), 55ff (1932)
10. Radde, S., Kaiser, A., Freitag, B.: A Model-Based Customer Inference Engine. In: Proc. of the ECAI-08 Workshop on Recommender Systems (2008)
11. Radde, S., Zach, B., Freitag, B.: Designing a Metamodel-Based Recommender System. In: Proc. of the 10th International Conference on Electronic Commerce and the Web (EC-WEB) (2009)
12. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall International Editions (1995)
13. Schaefer, R.: Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests. In: Workshop on Adaptivity and User Modelling (2001)