A Model-Based Customer Inference Engine

Sven Radde and **Andreas Kaiser** and **Burkhard Freitag**[1]

Abstract. In complex and frequently changing product domains, customers need qualified consultation services that allow them to make well-informed purchasing decisions without requiring excessive technical knowledge about the product domain. In this paper we present an intuitive customer metamodel with an associated inference engine which can be integrated into conversational recommender systems to provide comfortable reasoning about customer needs. The inferred knowledge can be used to manage the recommendation dialogue and to map non-technical customer statements into preferences for technical attributes.

1 Introduction

To be able to offer high quality assistance in complex product domains, recommender systems have to move away from their usually strictly feature-centric recommendation approaches towards customer-oriented models. Any good natural salesperson does not primarily ask technical questions to a customer but rather tries to elicit the customer's needs and expectations about his/her new product. From there, the salesperson uses his/her own technical expertise to map these "soft" statements to technical attributes that are most likely to satisfy the customer's needs.

Furthermore, customers expect a quality of recommendation when shopping online that is comparable to visiting a store – particularly if the product domain is highly complex or changes frequently, so that even technically savvy users would require assistance in their purchasing decisions. Electronic recommender systems that surpass the common simple "configurators" in functionality are a necessity to increase acceptance of online sales in these domains.

The contribution of this paper is an industrial strength customer metamodel, coupled with a Bayesian inference engine which is automatically derived from an instance of the customer model. The Bayesian engine allows a recommender system to classify its users with respect to different stereotypes, to assess their needs and, finally, to obtain information about the most recommendable products by inferring likelihoods for the different possible technical characteristics of the products-to-be-sold. Apart from being used to obtain product recommendations, a conversational recommender system can use the inference engine to decide on a dynamic course of its dialogue. The presented customer metamodel was designed in cooperation with an industry partner, to closely resemble the natural recomendation procedures in a wide range of imaginable business domains. When using the metamodel, the mathematical complexity is hidden, which enables intuitive model maintenance by non-programmers, i.e. marketing experts.

In summary, we present: (1) an efficiently maintainable, industrial strength *customer metamodel*, (2) an application of *Bayesian networks* as an inference engine for reasoning about customer profiles in recommender systems, (3) a *generation-method* to automatically derive the inference engine from an instance of the metamodel.

The rest of this paper is organized as follows: In section 2 we detail a representative use case before we describe our customer metamodel in section 3. We give a detailed description of the inference engine and its use in a recommender system in sections 4 and 5. We briefly cover model maintenance in section 6 and review some related work in section 7, before concluding with an outlook in section 8.

2 Use Case

Today's market for new automobiles is characterized by a huge number of choices, extended by different variants per vehicle, numerous optional features and special equipment often available only in packages combined with other extras. Customers need qualified consultation to match their (often vague) preferences with these complex product models. However, when visiting the web sites of major car manufacturers, we find that these offer so-called "configurators" only, that completely lack high quality recommendation functionality.

The course of action of "natural" salespersons is notably different, according to our personal experiences and interviews with vendors: Initially, the customer is classified into a set of broad stereotypes, such as "business customer", "young", or "male". Based on this classification, the vendor determines the further course of the dialogue. Different stereotypes of customers likely have different needs and expectations about their future car and the salesperson tries to assess those based on his/her experience in the field. Those needs are then mapped into suggestions for technical features and, consequently, available products. The customer is only asked technical questions when it is necessary for the recommendation process. Note that this concentration on *soft criteria* is the key difference between a natural sales dialogue and the common technology-oriented online car configurators.

Apart from its complexity, the product domain changes frequently and often radically. Updates of the product domain may stem from anything ranging from a temporary marketing campaign to the introduction of new vehicle variants or even technical innovations which often require significant adjustments to a recommendation dialogue to accommodate pre-

[1] Institute for Information Systems and Software Technology, University of Passau, Germany, email: {radde, kaiser, freitag}@uni-passau.de

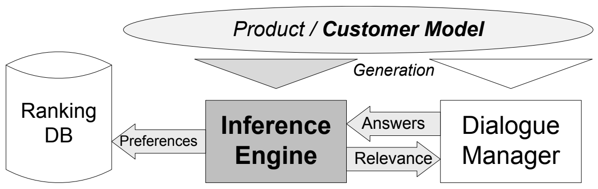

Figure 1. Architecture overview

viously unknown functionality. As an example, imagine the recent boom of GPS navigation assistants and the currently emerging approaches to location-based services where salespersons have to familiarize themselves with entirely new technologies, services and even business models. Therefore, maintainability of the domain model plays an important role, too.

We are cooperating with a local industry partner to realize a model-based conversational recommender system for this use case and similarly structured business domains. In [11], a dialogue structuring method was presented, into which the inference engine described here can be embedded. A ranking-enabled database querying technique was presented in [3]. Fig. 1 illustrates how the approach described here is integrated with those components to form a complete recommender system architecture (cf. section 5 for more detail).

The metamodel structure was designed by interviewing domain experts to closely resemble actual sales practice, recommendation processes and product catalog structures. Apart from being used in our prototypical "car domain", it was successfully instanced for another, similarly structured business domain in cooperation with a local industry partner. A detailed market study was conducted to supplement the experts' opinions and to build a solid understanding of the currently applied recommendation methodologies in that market to enable evaluations of our approach and to validate our notions of the used catalog and customer models. In the further course of our ongoing project, we will roll-out prototypes that build upon the metamodel to enable us to do field-tests to judge 1) the ability of domain experts to model their business domain and 2) the quality and precision of the produced predictions.

3 Customer Metamodel

The central part of the model is a domain-dependent description of the (prospective) customers. To represent the course of action shown in section 2, a domain expert defines a number of "interesting" *stereotypes* that are deemed relevant for an initial classification of the customer. Furthermore, a number of customer *needs* are defined that are assumed to be the driving force for the customer choosing certain products (for the product domain at hand). Finally, the model includes the technical attributes of the products to be sold, along with their possible values, as is shown in Fig. 2.

Owing to our usage of Bayesian networks in the inference engine, *stereotypes* are assigned a fixed a priori probability as part of the model. As detailed in section 4, the *stereotypes* will be represented as nodes in a Bayesian network modeling the event that a customer is of that particular stereotype.

To represent the interrelations between *stereotypes*, *needs* and *attributes*, we introduce *influences* and *matches*. These

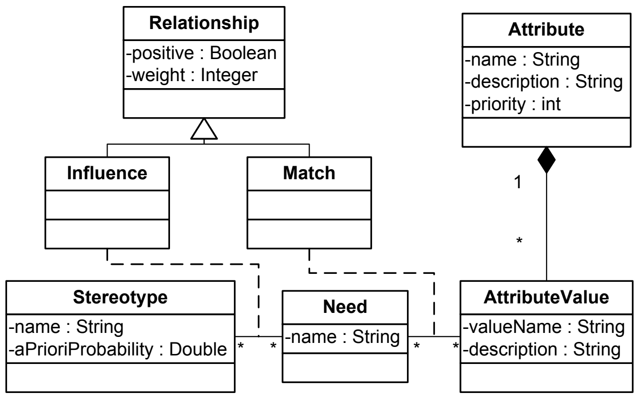

Figure 2. UML diagram for the customer metamodel

signify what kind of *influence* the fact that a customer belongs to a certain *stereotype* has on the likelihood that a customer will or will not have a particular *need* and how likely it is that a certain *attribute value* will satisfy the customer's *needs*.

Example 1 *As a brief and somewhat stripped example in the "car" domain, a domain expert might define the following* stereotypes *as relevant: "male", "young (age under 25)", "senior (age over 55)" and "business customer". Assume that the following* needs *are considered in this domain: "low price", "fast car", "comfortable car", "representative car".*

Furthermore, the domain expert defines the influences *between the* stereotypes *and the* needs *based on his marketing and domain knowledge, as shown in Table 1.*

Table 1. Positive and negative influences for example 1

positive influences		
"male"	→	"fast car"
"young"	→	"fast car"
"young"	→	"low price"
"senior"	→	"comfortable car"
"business customer"	→	"comfortable car"
"business customer"	→	"representative car"
negative influences		
"male"	→	"comfortable car"
"young"	→	"representative car"
"senior"	→	"low price"
"senior"	→	"fast car"
"business customer"	→	"low price"

Influences from *stereotypes* to *needs* have a *type* (cf. Fig. 2): A *positive influence* means that the likelihood that a customer will have a particular *need* increases if he or she is classified as belonging to the corresponding *stereotype*. In this case, being classified as not belonging to that particular *stereotype* decreases the corresponding likelihood. A *negative influence* is to be interpreted the other way round. We also introduce a numerical *weight* to be able to assign some measure of "importance" to different relationships.

In our Bayesian engine described later, the relation will be used to calculate the conditional probability tables for the nodes that represent customer *needs*, based on the a priori probabilities of the *stereotypes* (which would be replaced by evidence once the salesperson begins the dialogue).

Now, to be able to find products that are suitable for a particular customer, we have to establish another relationship that maps our knowledge about the customer to the technical properties of a product as stored in the product database. Our notion of the interaction between a customer's needs and the technical features is that particular values of product features may help to *satisfy* the needs of a customer.To this end, we define positive and negative *matches* between customer *needs* and possible *values* of product features very much the same way as the *influences* between *stereotypes* and *needs* (cf. Fig. 2). The underlying assumption is that having certain *needs* increases (or decreases) the likelihood that a customer prefers products with particular technical attributes, which enables us to retrieve the appropriate products from the catalogue.

Example 2 *When looking at the car painting, the* need *of "low price" has an influence on the possible values of the attribute "price" (e.g., "up to €500", "€500 to 1.000", and "more than €1.000"): The influence on "up to €500" is clearly positive, as it is the cheapest painting available. For the contrary reason it is negative on the price category "more than €1.000". The middle price category is not influenced by the* need *"low price". The rest remaining unchanged, this means that the cheapest paint has the highest likelihood of satisfying the customer's* needs*, followed by the middle price paint. The most expensive paint has the smallest probability.*

Furthermore, the need *"representative car" has a positive match with "metallic paint type" and the colors "black" and "blue", while it has a negative match with colors like "red", "yellow" etc. As an example showing that the relationship does not necessarily have to be based on purely objective criteria, consider a positive match between the* need *"fast car" and the color "red".*

Apart from eliciting the necessary information from a domain expert, it is possible to try to *learn* this relationship, e.g., by conducting dedicated marketing research or by analyzing data from previous successful sales or CRM systems. Those techniques may also be used to verify and refine an initial assessment made by a domain expert.

4 Generating the Inference Engine

4.1 Network Structure

To realize the inference engine with these desired properties, our approach is to represent the user model as a Bayesian network (cf., e.g., [12]). Some of the properties of Bayesian networks come particularly handy in our use case:

- Partial knowledge is transparently integrated into the network: In some cases, we do not have explicit information (= evidence) for, e.g., all the *stereotypes* that the customer may belong to. Evidence may easily be provided at any time to correct or refine the predictions.
- Although the predictions may not be very precise when evidence is scarce, all inquiries about a posteriori probabilities can be answered any time.
- Introducing evidence for a node has influence on the probability distributions of that node's parents. This leads to refined predictions in other parts of the network as well.

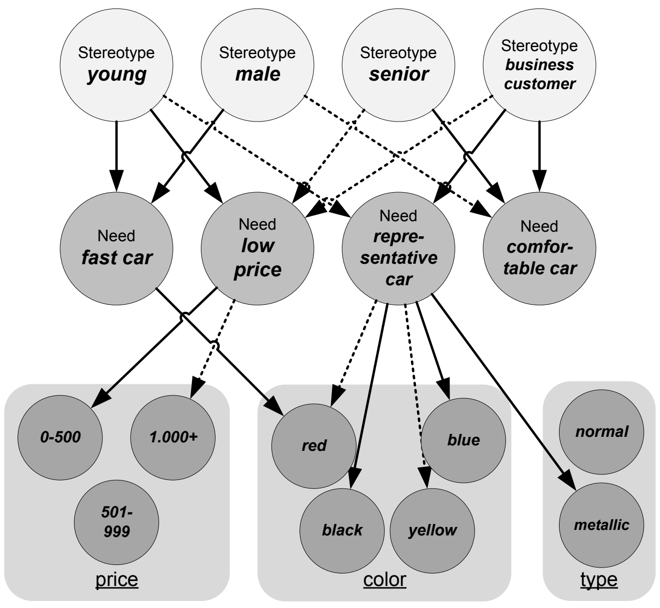

Figure 3. Bayesian network for the car painting *(solid edges denote positive influences/matches, dashed edges denote negative influences/matches)*

As established in section 3, our customer metamodel consists of *stereotypes*, *needs*, *attributes* and relations linking *stereotypes* and *needs*, as well as *needs* and *attributes* respectively. *Stereotypes* and *needs* will be represented as random variables (nodes) in our network, with a Boolean value range $\{false, true\}$. *Influences* and *matches* will form the edges of the network.

The Boolean values model the situation that a customer either belongs to a certain *stereotype* (resp. has a certain *need*) or not. Modeling each *stereotype* as a separate node which may be true independently of the others allows a very detailed classification of the customer in the form of a *"composite" stereotype*, e.g., "young female business customer" (based on the stereotypes introduced in example 1).

Product attributes in our Bayesian network are represented in a way similar to the representation of *needs* and *stereotypes*. However, attention must be paid to the fact that attributes generally do not have a Boolean value range. Instead, an attribute may have one of several values (e.g. the painting may be of several colors), as already detailed in the UML model (cf. Fig. 2). We chose to represent *each possible value* of each feature as a distinct node in our Bayesian network, again with a value range of *true* and *false*. A node in this case therefore represents the probability distribution that a particular value of an attribute is useful or desirable for the customer (i.e., the attribute value *matches* the *needs* of the customer).

Combining the model elements given in examples 1 and 2, a Bayesian network that deals with the car's paint is given in Fig. 3. For a complete, real life use case, the number of nodes at all three "layers" of the graph would grow, with the largest number of nodes to be expected for the attribute values.

An alternative representation of attributes could have used one single node per attribute that has all possible attribute values as its value range (i.e. a node "color" with "blue", "yellow" and "red" as possible values instead of three Boolean nodes "color::blue" etc.). While at a first look this approach seems more intuitive, the Boolean modeling captures some possible situations better than the alternative:

(1) The probabilities that the individual attribute values meet the needs of the customer are independent of each other. In particular, the case that two different attribute values are definitely useful for the customer (i.e. with 100% probability) could not be modeled with a single node per attribute.

(2) Having individual probabilites per attribute value gives an automatic normalization of the value range to [0,1]. With a single node per attribute, the information that one particular value would be useful with a probability of, e.g., 20% would not mean much without knowing at least the cardinality of the value range.

4.2 Conditional Probability Tables

Stereotype nodes will be assigned an a priori probability distribution (see also Fig. 2), based on CRM information about the demographic structure of the potential customers.

Definition 1 r_n *is defined as the* random variable *corresponding to node n in our Bayesian network. As mentioned previously, all* r_n *are binary random variables.*

To denote the set of all random variables corresponding to a set N of nodes, we use R_N.

Example 3 *Using data provided by the company's CRM system, the product manager assigns the following probabilities for the* stereotypes *of example 1:*

n	$p(r_n = true)$	$p(r_n = false)$
"male"	*0.70*	*0.30*
"young"	*0.40*	*0.60*
"senior"	*0.20*	*0.80*
"business cust."	*0.25*	*0.75*

We now have to provide a means to derive the conditional probability tables for nodes that represent *needs* and *attribute values*. To this end, we use a method similar to the "Noisy Add" technique (cf. [5]), which we now present in more detail.

Definition 2 *Let N be the set of nodes in the Bayesian network. An* edge *is a tuple* $e = (s, d, t, w)$ *with* $s \in N$ *the source node,* $d \in N$ *the destination node,* $t \in \{pos, neg\}$ *the* "type" *and w a positive real number representing the weight.*

Note that there are only edges between stereotypes *and* needs*, and* needs *and* attribute values*, respectively, as illustrated by the "three-layered" shape of the example network in Fig. 3.*

We define E as the set of all edges in the network. To access the individual components of a single edge, we use the common dot-notation, e.g., e.w to denote the weight w of edge e.

The semantics of an edge $e = (s, d, t, w)$ are as follows: If $t = pos$, the probability of r_d is increased if and only if r_s is *true*, "weighted" by w. For the case $t = neg$, the probability is decreased correspondingly. Intuitively, e.g., for a positive influence between a stereotype s and a need n, the probability that the customer feels the need n is increased iff. the customer belongs to s.

Definition 3 *For a* node $n \in N$ *we define the set* E_n *of* incoming edges *of n by* $E_n := \{(s, n, t, w) \in E\}$ *and the set* P_n *of* parents *of n by* $P_n = \{s \mid (s, n, t, w) \in E\}$.

Definition 4 $V(P_n)$ *is defined as a* valuation *of the random variables in* R_{P_n}*. We use* $V_s(P_n)$ *to denote a valuation of a single random variable* $r_s \in R_{P_n}$.

Using these defined valuations, the conditional probability $P(r_n = true \mid R_{P_n})$ that n occurs provided that the parent nodes of n occur is calculated using a *weighted average* of the parent random variables. For ease of notation, we first introduce the *weighted value* of an edge $e \in E$ given a valuation $V(P_{e.d})$ as:

$$wv(e, V(P_{e.d})) =$$

$$\begin{cases} e.w, & \textit{iff}\, V_s(P_{e.d}) = \textit{true} \wedge e.t = \textit{pos} \\ e.w, & \textit{iff}\, V_s(P_{e.d}) = \textit{false} \wedge e.t = \textit{neg} \\ 0, & \textit{otherwise} \end{cases}$$

Intuitively, $wv(e, V(P_{e.d}))$ returns the weight of the edge e in two cases: 1) The type of e is *pos* and $r_{e.s}$ is *true* for the valuation at hand or 2) the type of e is *neg* and $r_{e.s}$ is *false*. In all other cases, $wv(e, V(P_{e.d}))$ returns 0.

Now, to calculate the conditional probability distribution $P(r_n \mid V(P_{e.d}))$ of node n for a given valuation $wv(e, V(P_{e.d}))$ as a weighted average of the parents, we have to sum up the weighted values of all edges in E_n and divide by the sum of all weights in E_n:

$$\begin{aligned} P(r_n = true \mid R_{P_n})) &= \frac{\sum_{e \in E_n} wv(e, V(P_n))}{\sum_{e \in E_n} e.w} \\ P(r_n = false \mid R_{P_n})) &= 1 - P(r_n = true \mid R_{P_n})) \end{aligned}$$

This equation enables us to calculate the complete conditional probability table of the random variable r_n of node n, when the calculation is executed for all possible valuations $V(P_n)$. As an example, we detail the calculation of the conditional probability for node "comfortable car" (cf. Fig. 3):

Example 4 *For ease of notation, let n be the* need *"comfortable car" and* s_1, s_2, s_3 *the* stereotypes *"male", "senior" and "business customer", respectively.*

Using sample weights of 1, 2, *and* 3*, respectively, we get the following influences for n:*

$E_n = \{(s_1, n, neg, 1),\ (s_2, n, pos, 2),\ (s_3, n, neg, 3)\}$

We can now calculate the conditional probability distribution of r_n *for a given valuation* $V(R_{P_n^E})$*, e.g.:*

$p(r_n = true \mid r_{s_1} = true, r_{s_2} = false, r_{s_3} = true) = \frac{0+0+3}{6} = 0.5$

$p(r_n = false \mid \ldots) = 1 - p(r_n = true \mid \ldots) = 0.5$

Intuitively, this means that a customer who is characterized as a "male business customer that is not a senior" has a 50% probability of feeling the need for a comfortable car. Table 2 shows the complete conditional probability table for all possible valuations (i.e. all possible customers).

Given the a priori probability distributions of the *stereotype* nodes, an initial assessment of the *needs* that the "stereotypical" customer will or will not have can be done. When conducting an actual recommendation process with a customer, a recommender system would now assign evidence to any or all of the *stereotypes* nodes based on its initial assessment of the customer to obtain a personalized *needs* profile. Naturally, it is possible to re-state this evidence later or, instead of providing evidence, to just modify the a priori probability distribution if a more fine-grained assessment is desired.

Table 2. Conditional prob. for the need "comfortable car"

s_1	s_2	s_3	$p(n = \mathit{true} \mid \ldots)$	$p(n = \mathit{false} \mid \ldots)$
t	t	t	$(0 + 2 + 3)/6$	$1/6$
t	t	f	$(0 + 2 + 0)/6$	$4/6$
t	f	t	$(0 + 0 + 3)/6$	$3/6$
t	f	f	$(0 + 0 + 0)/6$	1
f	t	t	$(1 + 2 + 3)/6$	0
f	t	f	$(1 + 2 + 0)/6$	$3/6$
f	f	t	$(1 + 0 + 3)/6$	$2/6$
f	f	f	$(1 + 0 + 0)/6$	$5/6$

5 Using the Inferred Knowledge

In [3] an approach to rank items based on weight-annotated boolean conditions has been presented which was implemented as an extension of the standard SQL syntax allowing for a ranked result retrieval in databases. In this case, the calculated likelihoods can be directly used as weights defining the desired ranking, thereby balancing all of the customer's preferences against each other. When seen as an application of Multi-Attribute Utility Theory (MAUT) [13], this approach defines a utility function and profits from the fact that the single-attribute evaluations are not done on the attributes themselves but rather on the conditions, implying a normalized (i.e. boolean) value range.

As an alternative to utility methods, PreferenceSQL [8] could be used to query the product database, if the calculated likelihoods are transformed into suitable terms of its pareto-optimality based preference algebra.

Apart from creating product recommendations, the primary task of a conversational recommender system as presented in [11] is to choose the next question to ask the customer in an intelligent manner. Our inference engine allows to estimate the probability that a customer will have a particular *need* n. This probability is interpreted as how relevant n is to the customer, as the following definition details.

Definition 5 *Let* $p(n) \in [0, 1]$ *be the calculated probability that the current customer has the* need n*. Then,* $rel(n) = 2 * |p(n) - 0.5|$ *is the* relevance *of* n *to the customer. The higher the numerical value of* $rel(n)$*, the more relevant is* n *to the customer.*

The definition of $rel(n)$ is based on the assumption that a user will have stronger feelings about a particular *need* when we predict either a very high or a very low probability for that *need*. The dialogue manager of a recommender system should make sure to verify these predictions first, before attempting to clarify any *needs* that the user may have no clear opinion about (i.e. those with a predicted probability close to 0.5). Those *needs* may be examined later, when the customer has begun to build up trust in the recommender system.

Should it become necessary that detailed questions about the customer's technical preferences are required, it is necessary to define a relevance measure for *attributes* as well. We do so in a way similar to *needs*, taking into account that a question about an *attribute* will try to elicit answers about all possible values at once. The likelihoods that the various values of an attribute are useful for a customer are combined into a single relevance indicator.

Definition 6 *Given attribute* a *with* $dom(a) = \{v_1, \ldots, v_n\}$ *and the probabilities* $p(v_1), \ldots, p(v_n)$ *that value* v_i *matches the preferences of the customer, we define the* relevance *of* a *as* $rel(a) = \frac{\sum_{v \in dom(a)} |p(v_i) - 0.5|}{n/2}$.
Intuitively, the "distances" of the probabilities to the "undecided" value of 0.5 for each v_i *are summed up and divided by* $n/2$ *returning a value in* $[0, 1]$*. The higher the numerical value of* $rel(a)$*, the more relevant is* a *to the customer.*

Again, as with definition 5, $rel(a)$ is based on the assumption that attributes the customer has a clearly stated opinion about have a higher importance than those he or she is indifferent about. Consequently, these should be verified first and also be given a higher influence when computing the product recommendations than those for which the predicted probabilities are closer to 0.5.

Depending on the way the dialogue manager combines the relevance information for *needs* and *attributes*, using a common scale for both may be in order. Therefore, we chose to normalize $rel(n)$ and $rel(a)$ to the interval $[0, 1]$.

6 Model Maintenance

The approach was specifically designed to accommodate frequent model changes as efficiently as possible. Apart from major revisions of the model, all changes, such as extending the list of considered stereotypes are rather local modifications that leave the rest of the model intact. After the model has been modified to satisfaction, the Bayesian network can be rebuilt automatically from the model.

Example 5 *Regarding the current trend towards "green" technologies, a product manager decides to include the new* need *"environmentally friendly" into the customer model. First, he or she must decide which of the considered stereotypes influence this need. Second, the same must be done to find technical attributes that match this need, e.g., fuel consumption.*

Of course, developing friendly user interfaces that efficiently *guide* a non tech-savvy domain expert (say, e.g., a marketing manager) through the model maintenance process is still a task to be done. However, the stability of the model and inference layers allows for a separation of concerns in the development process and the model was specifically designed to allow an intuitive representation of a domain expert's marketing knowledge. When properly integrated into the existing business processes of catalogue creation and maintenance, these tools have the potential for even further optimizations, as they can be re-used, e.g., for the creation of printed marketing material in a natural way.

7 Related Work

Bayesian networks are used by Ji et al. in [7] to obtain recommendations in the commodities market. In contrast to our approach, they do not rely on a domain or customer model, but focus on learning the structure of the network and all probabilities from history data. Based on evidence provided by the current customer's purchases, other commodities are recommended depending on their posteriori probabilities. This kind of evidence is not available in our application scenario, as the customer generally will make a single purchase and leave.

Park et al. use Bayesian networks for a very detailed user representation in [10]. They use an expectation maximization algorithm to learn the conditional probability tables on their network. However, the structure of the network itself has been designed by a domain expert and is intended to remain fixed. Therefore, their approach requires extensive work when the underlying model changes.

Ardissono et al. [1, 2] 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 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), thus requiring repair actions. Our approach does not directly take elicited preferences as constraints but rather uses them as inputs to ranking-enabled database queries, returning a list of product recommendations which is ordered according to the customer's preferences. Giving a pre-sorted list of products instead of a single "optimal" product is intended to improve the customer's freemdom of choice. By evaluating the relevance measure, our approach can also suggest personalized default answers. It does not need to rely on extensive domain knowledge or a set of business rules as is the case in [1].

An approach similar to the one proposed in this paper is presented by Jameson et al. [6]. However, their utility estimations (the "value tree") do not seem to be built on an explicit model of the currently served customer but rather on an average user of their system. Hence, the recommendations are not personalized as strongly as in our approach which allows an adaption even to atypical customers by setting the appropriate stereotypes. 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 a single need. Furthermore, it is not completely clear how informal statements (i.e., "I am a law student.") can be interpreted as relevant knowledge (i.e. an increased interest in politics) by the system unless a domain expert models this association directly within the Bayesian Network.

In [4] Cao and Li develop a recommender system for consumer electronics by using a fuzzy-based approach for the inference process which involves reasoning about a product's features. The approach does not include an explicit customer model and therefore is limited to reason only about the technical features of products. Therefore, its potential for a conversational recommender system that aims at complex product domains appears limited.

A domain model based on dynamic logic programming was introduced by Leite and Babini in [9]. Both customer and user model are represented using a massive set of declarative rules which allows a detailed and powerful specification of the business domain – possibly even extended by user-supplied personalized rules. However, the complex formal models appear difficult to maintain and use even by domain experts, let alone customers.

8 Conclusion

We presented an application of Bayesian networks as an inference engine for reasoning about customer profiles in conversational recommender systems, enabling intelligent dialogue-management and preference-elicitation. The inference engine is generated from instances of an industry strength customer metamodel which improves the maintainability of the recommender system in complex and frequently changing product domains.

Initial evaluation of the approach based on interviews with domain experts has been quite positive. As a continuation of our work, a full-scale field test in a real-life application domain will be conducted in cooperation with a business partner. Furthermore, although model creation and maintenance are deliberately easy and supposed to be feasible by domain experts, we investigate how techniques to learn the Bayesian network structure from available sales data can be integrated to refine and verify the experts' assumptions.

REFERENCES

[1] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, M. Meyer, G. Petrone, R. Schaefer, W. Schuetz, and M. Zanker, 'Personalizing online configuration of products and services', in *Proc. of the 15th European Conference on Artificial Intelligence (ECAI)*, (2002).

[2] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, G. Petrone, R. Schaefer, and M. Zanker, 'A framework for the development of personalized, distributed web-based configuration systems', *AI Magazine*, **24**, 93–110, (2003).

[3] M. Beck and B. Freitag, 'Weighted boolean conditions for ranking', in *Proc. of the ICDE-08 Workshop on Ranking in Databases (DBRank'08)*, (2008).

[4] Y. Cao and Y. Li, 'An intelligent fuzzy-based recommendation system for consumer electronic products', *Expert Systems with Applications*, **33**, 230–240, (2007).

[5] P. Dagum and A. Galper, 'Additive belief-network models', in *Proc. of the 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, (1993).

[6] A. Jameson, R. Schaefer, J. Simons, and T. Weis, 'Adaptive provision of evaluation-oriented information: Tasks and techniques', in *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, (1995).

[7] J.Z. Ji, Z.Q. Sha, C.N. Liu, and N. Zhong, 'Online recommendation based on customer shopping model in e-commerce', in *Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence (WI)*, (2003).

[8] W. Kießling and G. Köstler, 'Preference SQL – Design, Implementation, Experiences.', in *Proc. of the 28th International Conference on Very Large Data Bases (VLDB)*, (2002).

[9] J. Leite and M. Babini, 'Dynamic knowledge based user modeling for recommender systems', in *Proc. of the ECAI-06 Workshop on Recommender Systems*, (2006).

[10] M.-H. Park, J.-H. Hong, and S.-B. Cho, 'Location-based recommendation system using bayesian user's preference model in mobile devices', in *Proc. of the 4th International Conference on Ubiquitous Intelligence and Computing (UIC)*, (2007).

[11] S. Radde, M. Beck, and B. Freitag, 'Generating recommendation dialogues from product models', in *Proc. of the AAAI-07 Workshop on Recommender Systems in E-Commerce*, (2007).

[12] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall International Editions, 1995.

[13] D. von Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research*, Cambridge University Press, 1986.