Semi-automatic Generation of Recommendation Processes and their GUIs

Hermann Kaindl[†], Elmar P. Wach[‡], Ada Okoli[◇], Roman Popp[†], Ralph Hoch[§], Werner Gaulke[$], Tim Hussein[$]
[†]Vienna University of Technology, Austria, kaindl@ict.tuwien.ac.at, roman.popp@tuwien.ac.at
[‡]STI Innsbruck, Austria, elmar.wach@sti2.at
[◇]Smart Information Systems, Austria, a.okoli@smart-infosys.at
[§]Dornbirn, Austria, hoch.ralph@gmail.com*
[$]University of Duisburg-Essen, Germany, werner.gaulke@uni-due.de, tim.hussein@uni-due.de

ABSTRACT
Creating and optimizing content- and dialogue-based recommendation processes and their GUIs (graphical user interfaces) manually is expensive and slow. Changes in the environment may also be found too late or even be overlooked by humans. We show how to generate such processes and their GUIs semi-automatically by using knowledge derived from unstructured data such as customer feedback on products on the Web. Our approach covers the whole lifecycle from knowledge discovery through text mining techniques to the use of this knowledge for semi-automatic generation of recommendation processes and their user interfaces as well as their comparison in real-world use within the e-commerce domain through A/B-variant tests. These tests indicate that our approach can lead to competitive results in terms of click-out rate, while requiring much less manual effort.

ACM Classification Keywords
H.5.2 User Interfaces: Graphical user interfaces (GUI)

General Terms
Design

Author Keywords
E-commerce; recommender systems; dialogue-based recommenders; model-based UI generation; ontologies.

INTRODUCTION
Dialogue-based recommendation processes have proven to be helpful tools in the area of e-commerce. Like content-based recommenders, they are based on explicit information about the items. According to [14], dialogue-based recommenders are especially helpful when users are looking for complex items that they do not buy very frequently (cars or computers, for instance). Manually creating and continuously optimizing such processes and their GUIs (graphical user interfaces), however, is laborious and error-prone.

In order to reduce the costs of creating a content- and dialogue-based recommendation process and its GUI for real-world use, we strived for automation. The recommender lifecycle proposed in this article makes use of knowledge sources on related products on the Web in order to generate recommendation processes semi-automatically.

To illustrate the approach, we use a running example based on a given product domain ontology (PDO). We use ontologies, since such formal representations can be more easily processed automatically than conventional, usually less structured product descriptions. The given PDO for mobile phones contains 74 properties representing various mobile phone features such as "face detection" or "operating system". These properties are referenced in the recommender dialogue to ask the end-users about their preferences (for instance, "Which operating system do you prefer?").

Within the automated recommender lifecycle, customer feedback on the products that are supposed to be recommended can be found as unstructured text on the Web. We use text mining techniques in order to adapt the PDO, including its annotations like recommendation priorities. Subsequently, the adapted PDO is used to generate a high-level abstraction of a recommendation process. Using domain-specific heuristics, this process is operationalized in all its details, including a GUI for its actual use by customers. We consider this an intelligent user interface supporting the customer in the course of browsing and potentially buying products. For instance, this approach ranked features higher that were related to the mobile phone display and the operating systems, while they were neglected or ranked differently in a human-generated version.

We deployed such a semi-automatically generated process in a real-world setting and compared its results to a process that was previously created manually. We use A/B-variant tests for this comparison. Data from real-world deployment of this new and automated approach provide empirical evidence of its usefulness. Deployed at a large e-commerce platform and tested with about 1,500 different customers, this approach increased the rate of customers who followed the recommendations by 14%. In addition to that, it reduced the manual adaptation work to very roughly half as much time. We regard the proposed recommender lifecycle as the major contribution of this paper, combining several mechanisms into an integrated approach that has proven to increase the quality of the recommendations while reducing manual effort.

*Ralph Hoch did this work while being with the Institute of Computer Technology (ICT) of the Vienna University of Technology.

IUI'13, March 19–22, 2013, Santa Monica, CA, USA.

The remainder of this paper is organized in the following manner. After discussing the state of the art, we introduce the recommender lifecycle, describing its components in detail. After that, we present an evaluation of this approach based on data obtained from real-world deployment. We conclude this paper by discussing the approach in a broader context.

STATE OF THE ART

To the best of our knowledge, no previous approach can directly be compared to the recommendation lifecycle proposed in this paper. However, several solutions for certain aspects of it have been proposed in the past. Dialogue- or critique-based recommenders have been in the focus of research for more than a decade [3, 18, 27]. In e-commerce, they have proven to be a good choice for guiding users through a variety of alternatives. While the user has a rather passive role in frequently used algorithms such as collaborative filtering [11], dialogue-based recommenders offer (and require) a higher degree of interaction between user and system. As a result, users tend to trust dialogue-driven recommendations more compared to recommendations generated by the system without interaction [30]. However, it is not trivial to select and order the questions that are supposed to be used in such a dialogue [5].

Explicit customer feedback to recommendations is rather scarce [15], so sources such as customer reviews or blog entries should be taken into consideration for revising the dialogue models according to recent trends or customer opinions. Data mining techniques have been proven to work well in the field of recommender systems [2]. They have, for instance, been used to identify the most salient features of products [12], to approximate user ratings [4], and to use this information directly during the recommendation generation process [16, 24, 32]. Mining and sentiment analysis techniques can be especially helpful, if collaborative filtering techniques do not yield meaningful recommendations due to sparsity problems [31].

In our case, ontologies build the foundation for the recommendations. Middleton et al. have introduced systems that use ontologies as a representation of user interests [19]. Ontological user profiling in recommender systems is also used by Sieg et al. [28]. Systems that use semantic product data [10] have been proposed as well. Hussein and Ziegler use product ontologies in combination with context ontologies for generating context-aware recommendations [13].

Mahmood and Ricci describe a conversational recommendation process as a sequential decision problem and model it as a Markov Decision Process [17]. A model of the user behavior is learned and used to acquire the adaptive strategy through Reinforcement Learning techniques. In this context, the system learns an optimal strategy by observing the effects of its actions on the final outcome of the recommendation sessions. Like in our approach, Aciar et al. use text mining techniques to analyze customer feedback [1]. However, their solution is not intended to be used in combination with conversational recommenders and does not provide a lifecycle like ours.

Website morphing as proposed by Hauser et al. [9] uses Bayesian Inference and Dynamic Programming to adapt layout and presentation of a Website to the user's preferred way of processing information. It focuses on how content is presented, while our approach creates a process and its content. In principle, these two approaches are complementary and could be combined.

THE PROPOSED RECOMMENDER LIFECYCLE

Figure 1 provides an overview of the proposed recommender lifecycle, illustrating the flow between the different steps. Initially, feedback is crawled and analyzed from customer reviews. Based on the results, the Product Domain Ontology (PDO) is updated, for instance by adding properties reflecting certain features that recently emerged for a type of product. The PDO then builds the foundation for discourse-based models, which in turn are used to generate concrete recommendation processes. Finally, the performance results of new recommendation processes are compared to those of existing ones using A/B tests. The dotted arrow indicates that the test results could be used to automatically adjust the PDOs again. This feature, however, is outside the scope of this paper.

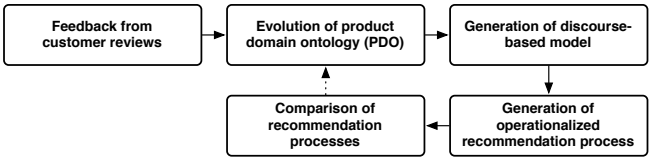

Figure 1. Overview of the recommender lifecycle.

Customer feedback on products

Discussing and reviewing products has become common in the modern Web. Users can express their opinion about a product on various sites. For example, many retailers such as amazon.com offer rating and feedback mechanisms for products offered. The approach presented in this paper analyzes customer feedback on products of a given product domain to identify the importance of certain features.

The underlying assumption for our approach is, that important features are discussed more often than unimportant ones. Hence, we count the number of appearances of features in the reviews. We do not take into account whether the feature is mentioned in a positive or negative way. We assume that the opinion about a feature is tied to the specific product and cannot be applied to the whole domain but the fact that the feature itself was mentioned makes it important.

Analysis of customer reviews is a major goal in the research of text- and opinion mining and several approaches are investigated [20]. Figure 2 illustrates the most important steps for processing customer feedback. The architecture is based on the approach for review analysis proposed by Hu and Liu [12] and was extended to support multiple feature detection techniques.

First, reviews are crawled from previously specified online sources. They are stored along with meta-data such as the respective publication date. Then, text mining algorithms are used to identify features mentioned in the reviews. We call these features *candidates*. These candidates are compared to

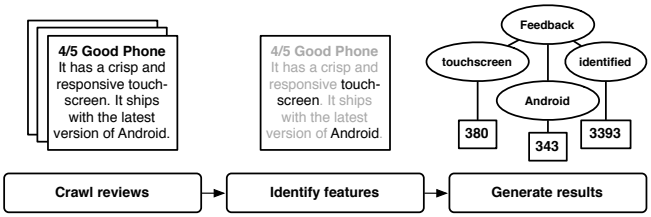

Figure 2. Steps necessary for feedback processing.

the features currently included in the PDO. Both candidates and existing features are counted and stored in a semantic representation for processing them later in the lifecycle.

The current implementation integrates supervised, unsupervised and probability-based algorithms for feature detection. Multiple Feature mining algorithms are used to even out shortcomings of the respective approaches. For example, the supervised associate rule mining algorithm proposed by Popescu and Etzioni [22] is used to detect features mentioned explicitly or implicitly. For each product domain, training data is, however, necessary. The unsupervised approach based on the ideas of Hu and Liu [12] does not need training data but is limited to features mentioned explicitly. It relies on the assumption that most of the features appear as nouns or in noun phrases. The probability-based word frequency algorithm proposed by Scaffidi et al. [26] uses a statistical approach to calculate probabilities for each mentioned noun in order to identify uncommon phrases as feature candidates.

After that, the feature lists obtained from these algorithms are compared to the features currently specified in the PDO, using the Levenshtein Distance algorithm for matching purposes. Candidates that could not be matched to features in the PDO are stored in order to be reviewed manually.

These steps are processed asynchronously. A scheduler crawls and analyzes new reviews on a regular basis.

Product domain ontology (PDO)

A PDO is the formal, explicit specification of a shared conceptualization of products and is specified in OWL DL[1]. In our case, the PDO includes both classes (concepts) and individuals as well as annotations (for example on priorities for the respective recommendation processes). The PDO should always contain the most important features for a certain product type. Thus, the PDO is adjusted regularly to reflect the results of the mining and analysis step.

Generally, ontologies may need to change if (i) there is an error in a prior version, (ii) a new way of modeling the domain is preferred, or (iii) a new terminology has been created[2]. In this research, customer feedback leads to reason (ii). The main research question is: How can feedback be utilized to facilitate PDO evolution with little or no human effort?

To answer the question, our recommender has been specified according to the adaptation strategy proposed in [29], and a feedback cycle has been established in which the feature relevance is fed back. This feedback metric is transformed into the Success Trend *ST*. The PDO is modeled according to GoodRelations[3] and evolved within that upper ontology. An excerpt of the PDO for mobile phones and related GoodRelations entities is depicted in Figure 3.

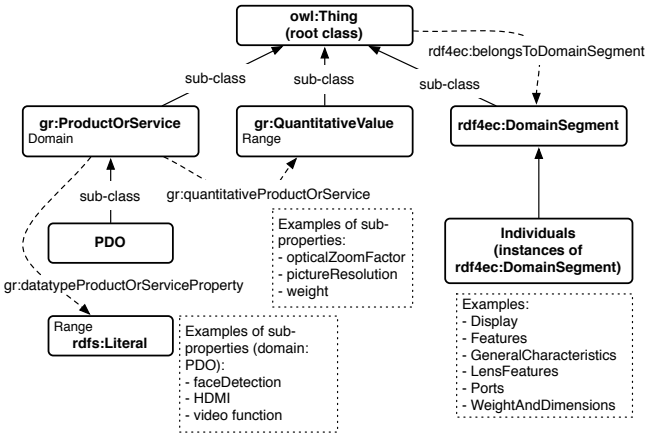

Figure 3. PDO 'MobilePhone' and related GoodRelations entities (excerpt). Solid lines indicate class relations, while dashed lines represent property relations.

The root class is `owl:Thing`, which has several subclasses. Among them are GoodRelations classes[4] such as `gr:ProductOrService` or `gr:QuantitativeValue`. Instances of the class `rdf4ec:DomainSegment`[5] represent groups of properties used in the respective product domain such as *WeightAndDimensions*. The PDO for the running example has 11 DomainSegments and is modeled as a sub-class of the class `gr:ProductOrService`.

The features of a product are modeled as properties (for instance, the weight of a mobile phone). In OWL, a property maps elements of a certain *domain* onto elements of a certain *range*. *Object properties* map objects or classes onto other objects or classes, whereas the range for *datatype properties* are literals such as Boolean or integer values, for instance.

In our PDO, the class `gr:ProductOrService` is the domain for all properties. In case the feature is modeled as an object property (`gr:quantitativeProductOrServiceProperty`), the range is the class `gr:QuantitativeValue`. Examples of object properties are depth, height, and weight which are sub-properties of `gr:quantitativeProductOrServiceProperty`. In case the feature is modeled as a datatype property (`gr:datatypeProductOrServiceProperty`), the range is a literal (`rdfs:Literal`). Examples of datatype properties are android, keyboard, and touchscreen which are sub-properties of the property `gr:datatypeProductOrServiceProperty` and have as domain the class MobilePhone, which is the sub-class of the class `gr:ProductOrService`. Each property is correlated with one individual via the annotation property `rdf4ec:belongsToDomainSegment`.

[1] OWL DL provides sufficient expressivity and is still decidable

[2] www.w3.org/TR/webont-req/#goal-evolution

[3] www.purl.org/goodrelations

[4] Those entities start with the namespace `gr:`

[5] `rdf4ec` is a custom namespace.

Additionally, the property `rdf4ec:priority` represents annotations on the relevance of product features for the related recommendation processes. The domain of this annotation property is the class `owl:Thing` and the range is specified through the datatype `integer` with values [0...+100]. It indicates the importance of a property and is used for processing the explicit customer feedback. A higher value of `rdf4ec:priority` means that the question about this feature is more important and, hence, be asked earlier in the dialogue.

The explicit customer feedback gathers information about products based on PDO extractions and is thus PDO- and property-based. The corresponding metric is "Feature relevance", whose RDF includes the property (i.e., feature name) and its relevance based on the count of appearances on the Web over a period of time. It is defined as $Feat(t)$ with the range $[0 \dots [$. $Feat(t)$ is converted into ST, which has the range $[0 \dots + 100]$, by calculating the new relevance of each property with the relative frequencies of the properties contained in the customer feedback. First, the maximum and minimum values of the properties as well as the sum of the values of all properties are determined and the interval for 100 classes is calculated. Second, the relative frequencies of the properties are calculated. Third, after having determined the classes and their bounds, the corresponding properties (i.e., the relative frequency of the property is within the bounds of the respective class) as well as the respective relevance are assigned to the classes. The new relevance is represented as given above (i.e., as annotation property `rdf4ec:priority`).

The feature relevance of the initial PDO was manually set to the same value for each property (i.e., all 74 properties had a priority of 30 as a heuristic guess, since only properties with high property values were used to generate the dialogues). Also, the triggering result of the A/B-variant test was manually designed. Subsequently, the customer feedback from the explicit customer feedback channel was retrieved, and $Feat(t)$ values for twenty properties were delivered (e.g., the property "android" describing the operating system appeared 343 times in the reviews). "keyboard" had the maximum $Feat(t)$, with a value of 500; "integratedRadio", however, was named only once. In total, all identified properties were discussed 3,393 times. The processing of the explicit feedback followed the three steps described above and finally led to the new priorities. Table 1 lists the development of `rdf4ec:priority` of the product features.

Table 1. Relevance changes of selected properties.

Property	$ST(t_0)$ (rdf4ec:priority)	$Feat(t_1)$	$ST(t_1)$ (rdf4ec:priority)
keyboard	30	500	100
touchscreen	30	380	77
android	30	343	69
flash	30	248	50
displaySizeInch	30	160	33
resolutionText	30	99	21
symbian	30	33	7
operatingSystem	30	23	6
integratedRadio	30	1	1

For properties that do not appear in the customer feedback, a specific function, the *Decay Function*, has been developed, calculating the priority for each of those product features. This function proposes a slight decrease of the relevance of a product feature (i.e., priority) in the first time period and a quicker decay later. This is due to the fact that the PDO already contains the priorities of product features derived from customer feedback earlier. Missing explicit customer feedback on a product feature should neither lead to its abolition nor to a strong decrease of its priority in a short time period. The Decay Function further proposes that the priority of a product feature will drop to zero (i.e., decay) after a certain period of time (e.g., in case a product feature has not been discussed for one year), which can be manually specified. It is a function: $Prio(t_0) \in \mathbb{N} \rightarrow Prio(t) \in \mathbb{N}$ where *Prio* is the priority of a product feature, t_0 is the point of time of the precedent customer feedback processing, and *t* is the point of time of the current customer feedback processing.

$$Prio(t) = Prio(t_0) + 1 - Rep * e^{\frac{t * \ln \frac{Prio(t_0)+1}{Rep}}{Decay}} \in \mathbb{N}$$

where *Prio(t)* is the priority of a product feature at time *t* (with 0.5 being rounded up), which is the point of time of the current customer feedback processing, *t* is the number of days from an initial date, *Prio(t_0)* is the priority of a product feature at time t_0, which is the point of time of the precedent customer feedback processing[6], *Rep* is the representativeness of not discussing a product feature, which reduces the priority of a product feature the more reliable the absence of a discussion is (the default value is +1), *Decay* is the number of days to a priority of zero (the default value is 365 days).

For the running example, the customer feedback processing was on October 28, 2011. The initial date is defined to be September 1, 2011 (i.e., the deployment of the prototype). Hence, *t* is 57 (i.e., 57 full days after September 1). With the default values for *Rep* and *Decay*, the new priorities are calculated to be 29.29, thus the initial priorities are decreased from 30 to 29.[7] Examples of properties that got this priority are bluetooth, GPS, integratedCamera, mp3Player, videoFunction, and WLAN.

After the feedback processing, a new PDO version is created. With the process described, the PDO adapts to the feedback automatically without any explicit, direct intervention from a human.

Recommendation processes as discourse-based models

From such an annotated PDO, we automatically generate a recommendation process on a conceptual level first. It is represented as a discourse-based model (for the background and a definition of such models see, for instance, [6, 23]). The Discourse Model, as its major part, essentially consists of questions and answers about those items annotated as important in the PDO, and their sequence.

Figure 4 shows an excerpt of such a Discourse Model. The related questions and answers are modeled as so-called *Adjacency Pairs* (shown as diamonds), with opening and closing Communicative Acts (shown as rounded rectangles). These

[6] A property that has never received customer feedback gets the value it had on the initial date

[7] 29.29 is rounded off to the next integer, which is 29

Adjacency Pairs are related to so-called *Discourse Relations*. In a model of such a recommendation process, only three types of Discourse Relations are used. The first one, `Sequence`, is shown as hexagon (since it is more specifically a Procedural Construct), the second, `Joint`, and the third one, `Background`, are shown as rectangles (since they are Rhetorical Relations). Because such a model can have many `Joint` relations and many Question-Answer Pairs, only one such element is shown each in order to visualize the pattern, which is repeated.

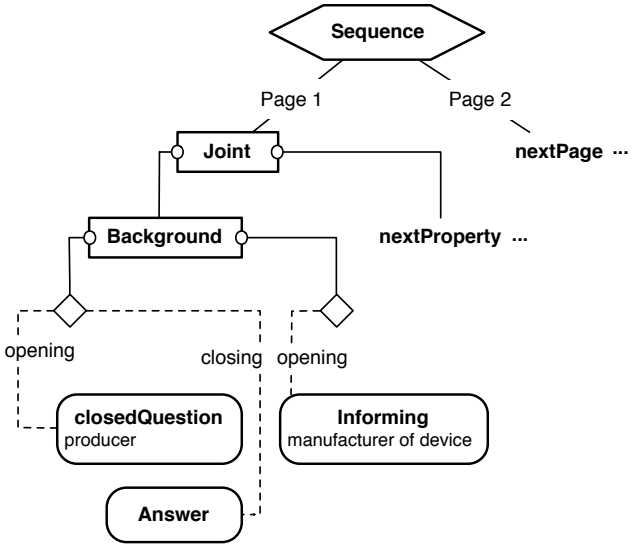

Figure 4. Excerpt of Discourse Model of recommendation process.

The automatic generation of such a model comprises two steps. First, our model-transformation approach transforms the instances of the PDO classes and their concrete datatypes and object property values into a model of the *content* of the communication (the Domain-of-Discourse model). Second, a set of model-transformation rules matches parts of the PDO (including its individuals) and transforms them automatically into corresponding parts of a Discourse Model. This step also defines the content of the Communicative Acts, so that the Discourse Model refers to the Domain-of-Discourse Model.

In the first step, all properties of the PDO are analyzed. Each property is stored as a datatype in the Domain-of-Discourse Model. For most of the datatypes, the values used by the individuals are added to the datatype. In case of numeric properties, only the minimum and maximum values are stored in the Domain-of-Discourse Model. All properties that are set on a minimum of 70% of the individuals are taken into account for further calculations in the second step, since it was observed in real-world practice over time that users trust the results more, if an answer leads to a sufficient number of products. The other properties are not used, since they do not seem to be relevant for a sufficient number of individuals.

In the second step, a Discourse Model is created. For each used property a question-answer pair is created according to a predefined template. Such a template consists of a *Question* and a corresponding *Answer* Communicative Act. These two Communicative Acts are connected with an *Adjacency Pair*. The description property of the transformed property is added to the defined Discourse-Model part with a *Background* relation connecting the Question-Answer Adjacency Pair with an *Informing* Communicative Act, containing the description. It is also possible that some properties of a domain segment are combined into a single question. This combination is applied, if there is more than one property of type Boolean. In our running example, the properties of the domain segment *Operating Systems* are combined. Several properties like *android*, *symbian*, *windowsMobile* have been combined to form a single question.

The generated parts of the Discourse Model are then sorted according to previously specified heuristics. An example of such a heuristic is, that the question for the producer property always has to be the first one. Another heuristic is, that questions for properties with a high priority are asked before properties with a lower priority. This sorted list is then divided into more logical units (eventually becoming pages in the resulting user interface). Each of these logical units has a configurable value of questions. In the Discourse Model, each of these logic units is represented with a *Joint* Relation, connecting the respective parts. The generated Joint Relations are connected then with a *Sequence* Relation specifying their order.

Operationalized recommendation process and its UI

Finally, the discourse-based models can be used to generate so-called *concrete user interfaces (CUI)* as depicted in Figure 5 as well as the actual recommendation processes that are eventually used on the Website. The CUI comprises both the structure of the process and the course of events, which defines how end-users navigate through the process.

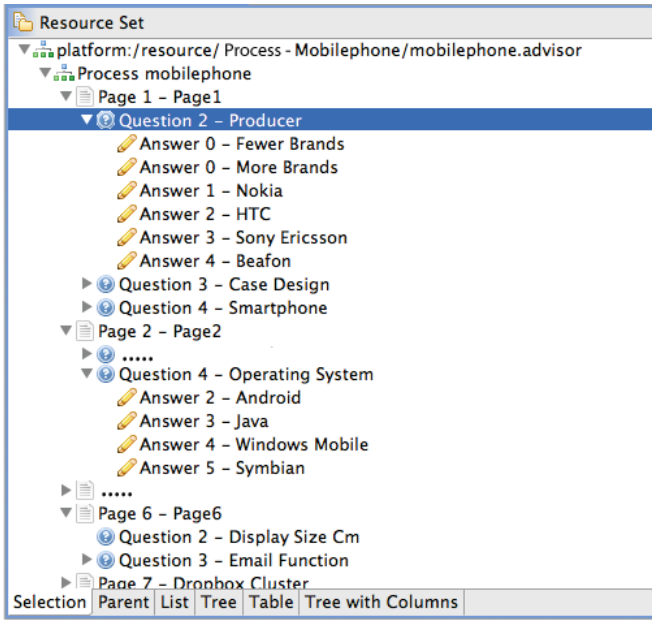

Figure 5. Advisor Model example at the CUI level.

Several heuristics are used for refining the discourse-based models and for transforming elements into concrete represen-

tations. The following list illustrates selected specifications of the question element:

- *String questions* have several predefined answer sets that are defined in the Domain-of-Discourse Model and refer to a specific property. The "Producer" property in Figure 5 represents such a question, as each answer (a brand in this case) is only related to one property.
- *Numeric questions* referring to specific properties enable to define a value range such as "Display Size Cm". Boundary values and step-width are defined in the Domain-of-Discourse Model.

Furthermore, a specific representation of each question type is defined. A numeric question, for example, can be represented by a single (minimum or maximum value) or double slider (range of values). In Figure 5 for instance, "Display Size Cm" represents a numeric question. Slider values, such as start and end value, are based on the actual values of the products and the specific representation is predefined in the Domain-Of-Discourse Model. Similar representations apply to the other question types. In addition to that, several other heuristics support the model transformation, covering for instance:

- text-patterns for questions,
- definition of background information,
- restricted answers per question,
- restricted answer selection per question, etc.

The resulting model comprises the logical layers used during a recommendation process. This means that a root element "process" serves as a starting point, containing an ordered set of pages. These pages are used as containers for question-answer pairs, where each question can have multiple related answers. Answers themselves can be restricted and special answer elements can be used to show/hide additional answers ("more"/"less" switch). Moreover, questions are based on product properties and have varying characteristics, which are rendered differently in the final user interface. Figure 6 illustrates an example with two different question types and their respective representations.

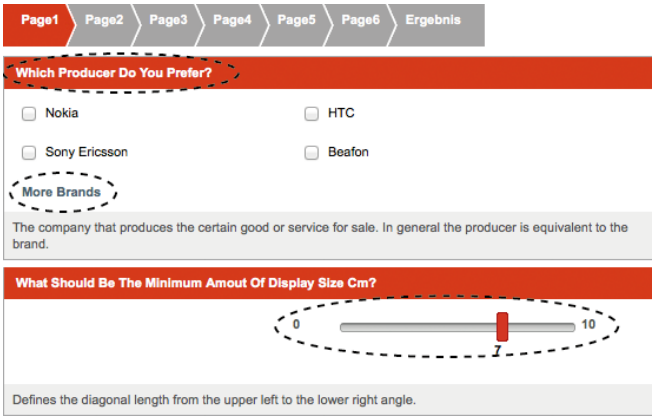

Figure 6. Example of a generated user interface.

Enabling the user to easily navigate through the process and to identify the important parts is mandatory. Thus, heuristics have been put in place to support this task. We demonstrate this behavior using the "Producer" property. Text-patterns are used to automatically generate meaningful questions that can easily be processed by end-users. Different approaches are used for different question types. In case of a string question, the text-pattern uses the property name to create a comprehensive question. Furthermore, the number of answers is limited and only a fragment is shown, so that customers are not overwhelmed by options. A special switch can be used to show/hide additional answers ("more"/"less" switch).[8]

A CUI model can be seen as a 1:1 mapping for the final user interface but still is independent from concrete implementations. One of the benefits of this method is that it facilitates manual adjustment of the process, if necessary, without changes at the implementation level. Adjustments can involve, for example, adding/deleting elements (pages, questions and answers) as well as changing question types. In addition to that, it facilitates the use of different implementations or layouts for the final user interface.

Figure 5 shows how a generated recommendation process model on the CUI-Level looks like. Technically, it is transformed into an RDF representation and stored in a database, serving as input for an already existing GUI-building component of the participating company. From this information on structure and content of the recommendation process, this component generates HTML-based GUIs as the one depicted in Figure 6. In essence, it queries the database with the RDF representation to automatically populate flexible HTML-based templates with the actual content such as questions, answers, and texts, and configures the layout according to the recommendation process model at hand. This does not require any modifications of source code.

The combination of actual instance data (e.g., products of a specific Web-shop) is done at runtime by an external engine, enabling a more flexible UI behavior than the runtime engine that some of the authors presented in previous publications [7, 23]. The engine used for recommendation generation allows the user to redefine his or her criteria and also to go back in the process.

EVALUATION

In order to measure the effectiveness of our approach, we compared the semi-automatically generated recommendation processes to manually created ones in a real-world retail setting.

General design

The study was designed and executed together with an Austrian company that offers recommendation process dialogues as a service for retailers. A large German online retailer (client of the Austrian company) was involved as well. The experiment ran for 14 consecutive days and involved 2,168 uninformed online customers, who accessed the recommendation processes to find suitable mobile phones.

[8]For the sake of simplicity, this is a simplified description of the CUI-Model. It is possible to parameterize the model transformation, and various heuristics are in place for different requirements.

Usually, the recommendation processes are designed manually by the company's project managers, who have the necessary domain knowledge and are skilled in designing successful recommendation processes for different online shops. The designed recommendation processes meet high quality standards and thus, can serve as a baseline for the evaluation. In addition to the effectiveness of the semi-automatically created recommenders, we also wanted to know if our approach can help reduce the manual effort required for the setup of recommender processes. Although we did not execute a specific study dedicated to this question, the evaluation process yielded some information regarding this aspect as well.

Study 1: Mobile phones

We used A/B-variant tests to measure the effectiveness of the semi-automatically generated recommendation processes for mobile phones. The manually created recommendation process (A) and the semi-automatically created recommendation process (B) were each displayed to approximately 50% of all customers who visited the retailer's Website within the test period. From July 7 to October 28, 2011, we crawled and analyzed a total of 4,044 reviews from amazon.com for the domain of mobile phones. Based on these reviews, 3,393 features were identified and compared to those in the existing PDO for mobile phones. For example, the feature "touchscreen" was mentioned 380 times, whereas "weight" was only mentioned 165 times.

Both the semi-automatically generated and the manually created recommendation processes order product features according to their relevance for users. This procedure is based on the underlying assumption that the effectiveness of a dialogue-based recommender systems increased, if the selectable product features were arranged by relevance. While human domain experts applied their domain knowledge to identify the relevance of product features, the semi-automatically generated recommendation processes exploited information from user reviews and the corresponding ranking of the product features within the PDO, which is explained above. At the time of the test, the manually created variants were about 7 months old.

For the evaluation, we applied the frequently used "click-out rate"[9] in order to measure the effectiveness of the process. In our case, it denotes the fraction of successful recommendations compared to the total amount of generated recommendations. A recommendation is regarded as successful, if the user requests details about the particular product or puts it in his or her electronic shopping cart.

Results of Study 1

Comparing the mobile phone recommendation process (Table 2), we found that the semi-automatically generated variant of the mobile phone recommendation process placed features such as *display* or *operating systems* on higher positions, while these were ranked differently or neglected by humans.

Moreover, product features that were related to Multimedia (Camera, Video, MP3) and Connectivity (WLAN, GPS, Bluetooth) were listed on lower positions within the semi-automatically generated variant. This suggests that these features have lost relevance over time, unrecognized by the domain experts. These results reflect the relevance of properties as given in Table 1. Here, the product features *touchscreen*, *displaySizeInch* and *resolutionText (Display)* and *android (Operating system)* received higher priorities than *videoFunction* and *mp3Player (Multimedia)* or *WLAN, GPS and Bluetooth (Connectivity)*.

[9]Sometimes referred to as "click-through rate."

Table 2. Comparison of ranking of properties in mobile phone recommendation processes. Left: Manually created process. Right: Semi-automatically generated process.

• Brand	• Brand
• Price	• Price
• Multimedia	• **Display**
– Camera	– Touch Display
– Video	– Color Display
– MP3	– Display size
• Connectivity	• **Operating System**
– WLAN	– Android
– GPS	– Symbian
– Bluetooth	• Camera
• Operating System	• Connectivity
– Android	– WLAN
– Symbian	– GPS
	– Bluetooth

Table 3 shows the results of the A/B-variant test in which the semi-automatically generated mobile phone recommendation variant (B) was tested against a manually created variant (A).

Table 3. Results of A/B test with instances of the recommendation process generated (A) manually and (B) semi-automatically. UC: unique clients, CO: click-outs, COR: click-out rate.

Variant	UC	CO	COR
(A) Manual	1,068	421	0.394
(B) Generated	1,100	496	0.451

The method for comparing two proportions from univariate inferential statistics was used to calculate the results. The null hypothesis, which states that the two samples show no significant differences ($H_0 : variantA = variantB$), was tested. Additionally, the confidence interval was calculated in order to determine a lower and upper bound of the absolute difference between the two proportions.

A total of 1,068 unique clients used the manually created variant, which had 421 click-outs. The semi-automatic variant had 1,100 unique clients and 496 click-outs. The comparison of the two mobile phone variants shows, that the null hypothesis (H_0) should be rejected and a significant difference between the two variants exists ($z = -2.67, \alpha = 0.05$). With a confidence of 95%, the semi-automatic variant performed between 2% and 10% better in terms of absolute proportions than the manual variant ($\alpha = 0.05$).

Study 2: Other product categories

As a follow-up study, we carried out another one that included a total of seven product categories: Blu-ray player, camcorder, printer, receiver, video projector, DVD player and TFT screen. This study was set up analogously to the mobile phone study.

Results of Study 2

Table 4 sums up the results of the second study. For each domain the proportion was compared and tested, whether a significant difference between the two cases exists and H_0 should be rejected for $\alpha = 0.05$. Additionally, the range of the difference between the semi-automatic version and the manually created version is given with a confidence of 95%.

Table 4. Results and comparison of A/B-variant tests of recommendation process instances in 7 product categories — manually vs. semi-automatically generated process instance. $C_{A/B}$: unique clients for cases A (manual) and B (semi-automatic), $COR_{A/B}$: click-out-rates for cases A and B, $D(b, a)$: Difference from b to a with 95% confidence, z: computed z-value for the two proportions. Significant values are printed in bold. The null hypothesis is rejected for $\alpha = 0.05$.

Domain	C_A	C_B	COR_A	COR_B	$D(b, a)$	z
Blu-ray pl.	806	825	0.458	0.509	0..10%	**-2.071**
Camcorder	205	233	0.444	0.485	-5..13%	-0.860
DVD player	217	239	0.415	0.213	-28..-12%	**4.646**
Printer	382	364	0.683	0.547	-21..-7%	**3.834**
Receiver	677	686	0.316	0.319	-5..5%	-0.120
TFT screen	525	494	0.455	0.348	-17..-5%	**3.481**
Video proj.	156	173	0.429	0.462	-7..14%	-0.600

The semi-automatic advisor for Blu-ray player showed a significant difference over the manually created variant ($z = -2.071, \alpha = 0.05$) and performed between 0% and 10% better in terms of absolute proportions ($\alpha = 0.05$).

The comparison in the domains DVD player ($z = 4.646, \alpha = 0.05$), printer ($z = 4.646, \alpha = 3.834$) and TFT screens ($z = 4.646, \alpha = 3.481$) showed significant differences between the two proportions. In all three cases the semi-automatic version performed worse than the corresponding manually created one. The semi-automatic versions performed 12% to 28% (DVD player), 7% to 21% (printer) and 5% to 17% (TFT screen) worse in terms of absolute proportions than the manually created versions ($\alpha = 0.05$).

The domains of camcorder ($z = -0.860, \alpha = 0.05$), receiver ($z = -0.120, \alpha = 0.05$) and video projector ($z = -0.600, \alpha = 0.05$) showed no significant differences between the two variants and, therefore, H_0 cannot be rejected. The performance of the semi-automatic versions varied from -5% to 13% for camcorder, -5% to 5% for receiver and -7% to 14% for video projector ($\alpha = 0.05$).

Efficiency of the proposed approach

We interviewed several project managers of the Austrian company mentioned earlier who are usually involved in the (manual) setup of process recommendation systems. Depending on the complexity of the product domain, setting up or optimizing new advisors usually takes 1–2 days of work. This includes analyzing product reviews and pertinent Websites to identify the relevance that certain product features have to customers.

This task is followed by a reassessment of current recommendation processes and updating their contents according to the findings (restructuring questions and answers, adjusting the underlying product data structure, refining wordings and display).

With the support of the presented lifecycle to generate recommendation systems automatically, the domain experts were relieved of the manual research tasks and manual restructuring of the recommendation systems, which reduced the overall time required to several (6–8) hours. Thus, the automated process proposed in this paper led to a considerable reduction of manual effort.

DISCUSSION

The results of the studies suggest that the proposed lifecycle is capable of generating well-performing recommendation processes for real-life use.

However, analyzing the individual structures of the generated processes and their performance, led to the assumption that the automated approach works well for basic recommendation process structures. The three recommendation processes with significant negative performance (Printer, DVD Player, TFT screen) had more complex structures, in which the questions where interrelated (e.g., selecting a question triggered the display of an additional question), while this was not the case for the generated recommendation processes with significant positive performance (mobile phone, Blu-ray player). This preliminary assumption will be evaluated in future tests.

Measuring and comparing the efforts for the manual set up of recommendation processes to using the semi-automatic lifecycle showed that applying the latter led to a reduction of the manual work required by roughly half as much time.

Since our approach to generating recommender processes is semi-automatic, the question may arise what has to be done manually. First, the PDO was given and not automatically created. And of course, the whole machinery such as repositories, queries, transformation rules, etc. was manually devised. Once it had been set-up, however, it created the recommender processes fully automatically. Still, the selection of a new (generated) recommender process for deployment is done manually. This may be automated as well, however, once our closed optimization cycle will be available.

While our approach works and even results in recommendation processes applicable in practice, a few related questions should be discussed. In particular, is it necessary or, at least useful to generate a high-level model of such a process first? Although it seems possible to generate an operationalized process in one shot, we argue that a two-step approach is useful by analogy to compiler construction. Typically, intermediate languages are used on the way from a high-level programming language to machine code. These provide useful levels of abstraction for the compiler developers, and so does our high-level model of recommendation processes.

Still, the question remains, whether other kinds of languages or models could serve the same purpose, and possibly even better. This question cannot be answered with certainty as it stands, since our approach seems to be the very first along these lines.

As our discourse-based models are primarily used for specifying models on a high abstraction level for automated generation of user interfaces, how about the most often used ap-

proach for this purpose? Instead of discourse-based models, task-based *ConcurTaskTrees* [21] may be used for bridging the semantic gap between ontologies and user interfaces. ConcurTaskTrees facilitate modeling *tasks* and their causal and temporal relations. Such models are also being transformed into a user interface semi-automatically. However, we are not aware of any approach for generating ConcurTaskTrees out of ontologies.

UsiXML [8] is an XML-based specification language for user interface design. It allows specifying a user interface at different levels of abstraction, from high-level task models (like ConcurTaskTrees) to the concrete code of a user interface. Also for UsiXML, we are not aware of any approach for generating UsiXML models out of ontologies.

Since a recommendation process of the kind generated here primarily consists of pairs of questions and related answers, Communicative Acts as used in the discourse-based approach are an excellent fit for modeling them. In contrast, tasks would have to model questions and answers in the sense of corresponding interactions with a specific kind of user interface. So, while this would certainly be feasible, it appears to be less appealing than our approach.

From discourse-based Communication Models, it is possible to generate general-purpose graphical user interfaces according to [7] and even optimized ones for devices with small screens according to [25]. Contrasting them with the user interfaces generated for recommendation processes as explained above, it is clear that the latter are preferable in terms of usability.

As a matter of fact, they have been successfully used for real-world application of these recommendation processes. They are special-purpose, however, and their overall appearance was predefined, while only the content has been generated automatically for the given structure and with many given heuristics.

CONCLUSION

In this paper, we present a novel approach for semi-automatically generating dialogue-driven recommendation processes based on product ontologies. By analyzing product reviews, we were able to identify trends for product types automatically and to adapt these product ontologies accordingly. In rapidly evolving domains like mobile phones, the automated generation led to improved recommendation dialogues (in terms of click-outs).

The evaluation showed that our approach can reduce the effort for creating and adjusting the recommendation processes significantly. In addition, data from real-world deployment of this new and automated approach provide empirical evidence of its usefulness.

We are currently working on implementation and test of including automated feedback for the evolution of the PDO also from comparing recommendation processes during their deployment. It will close the loop for an optimization cycle, and this will hopefully improve the results further.

ACKNOWLEDGMENTS

This research has been carried out in the SOFAR project (No. 825061), partially funded by the Austrian FIT-IT Program of the FFG.

REFERENCES

1. S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Recommender system based on consumer product reviews. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 719–723. IEEE Computer Society, 2006.

2. X. Amatriain, J. M. Pujol, N. Tintarev, and N. Oliver. Rate it again: Increasing recommendation accuracy by user re-rating. In *RecSys '09: Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 173–180. ACM, 2009.

3. R. Burke, K. Hammond, and B. C. Young. The FindMe approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

4. J. Carrillo de Albornoz, L. Plaza, and P. G. A. Díaz. A joint model of feature mining and sentiment analysis for product review rating. In P. Clough, C. Foley, C. Gurrin, G. J. F. Jones, W. Kraaij, H. Lee, and V. Mudoch, editors, *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2011.

5. M. Doyle and P. Cunningham. A dynamic approach to reducing dialog in on-line decision guides. In *EWCBR '00: Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning*, pages 49–60. Springer, 2000.

6. J. Falb, H. Kaindl, H. Horacek, C. Bogdan, R. Popp, and E. Arnautovic. A discourse model for interaction design based on theories of human communication. In *Extended Abstracts on Human Factors in Computing Systems (CHI '06)*, pages 754–759. ACM, 2006.

7. J. Falb, S. Kavaldjian, R. Popp, D. Raneburger, E. Arnautovic, and H. Kaindl. Fully automatic user interface generation from discourse models. In *IUI '09: Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 475–476. ACM, 2009.

8. D. Faure and J. Vanderdonckt. User interface extensible markup language. In *EICS '10: Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 361–362. ACM, 2010.

9. J. R. Hauser, G. L. Urban, G. Liberali, and M. Braun. Website morphing. *Marketing Science*, 28(2):202–223, 2009.

10. M. Hepp. Products and services ontologies: A methodology for deriving owl ontologies from industrial categorization standards. *International Journal on Semantic Web & Information Systems*, 2(1):72–99, 2006.

11. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
12. M. Hu and B. Liu. Mining opinion features in customer reviews. In *AAAI'04: Proceedings of the 19th National Conference on Artificial Intelligence*, pages 755–760. AAAI Press, 2004.
13. T. Hussein and J. Ziegler. Adapting web sites by spreading activation in ontologies. In *ReColl '08: Proceedings of the International Workshop on Recommendation and Collaboration*. ACM, 2008.
14. D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.
15. G. Jawaheer, M. Szomszor, and P. Kostkova. Characterisation of explicit feedback in an online music recommendation service. In *RecSys '10: Proceedings of the 4th ACM Conference on Recommender Systems*, pages 317–320. ACM, 2010.
16. A. Levi, O. Mokryn, C. Diot, and N. Taft. Finding a needle in a haystack of reviews: Cold start context-based hotel recommender system. In *RecSys '12: Proceedings of the 6th ACM Conference on Recommender Systems*, pages 115–122. ACM, 2012.
17. T. Mahmood and F. Ricci. Learning and adaptivity in interactive recommender systems. In *ICEC '07: Proceedings of the 9th international Conference on Electronic Commerce*, pages 75–84. ACM, 2007.
18. L. McGinty and J. Reilly. On the evolution of critiquing recommenders. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 419–453. Springer, 2010.
19. S. E. Middleton, N. R. Shadbolt, and D. C. de Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88, 2004.
20. B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.
21. F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTrees: A diagrammatic notation for specifying task models. In *Interact '97: Proceedings of the IFIP TC13 6th International Conference on Human-Computer Interaction*, pages 362–369, 1997.
22. A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346. ACL, 2005.
23. R. Popp and D. Raneburger. A high-level agent interaction protocol based on a communication ontology. In C. Huemer, T. Setzer, W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, and C. Szyperski, editors, *E-Commerce and Web Technologies*, volume 85 of *Lecture Notes in Business Information Processing*, pages 233–245. Springer, 2011.
24. S. Raghavan, S. Gunasekar, and J. Ghosh. Review quality aware collaborative filtering. In *RecSys '12: Proceedings of the 6th ACM Conference on Recommender Systems*, pages 123–130. ACM, 2012.
25. D. Raneburger, R. Popp, S. Kavaldjian, H. Kaindl, and J. Falb. Optimized GUI generation for small screens. In H. Hussmann, G. Meixner, and D. Zuehlke, editors, *Model-Driven Development of Advanced User Interfaces*, volume 340 of *Studies in Computational Intelligence*, pages 107–122. Springer, 2011.
26. C. Scaffidi, K. Bierhoff, E. Chang, M. Felker, H. Ng, and C. Jin. Red opal: product-feature scoring from reviews. In *EC '07: Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 182–191. ACM, 2007.
27. S. Schmitt. simvar: A similarity-influenced question selection criterion for e-sales dialogs. *Artificial Intelligence Review*, 18(3-4):195–221, 2002.
28. A. Sieg, B. Mobasher, and R. Burke. Improving the effectiveness of collaborative recommendation with ontology-based user profiles. In *HetRec '10: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, pages 39–46. ACM, 2010.
29. E. P. Wach. Automated ontology evolution for an e-commerce recommender. In W. Abramowicz, L. Maciaszek, K. Wecel, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Business Information Systems Workshops*, volume 97 of *Lecture Notes in Business Information Processing*, pages 166–177. Springer, 2011.
30. B. Xiao and I. Benbasat. E-commerce product recommendation agents: Use, characteristics, and impact. *MIS Quarterly*, 31(1):137–209, 2007.
31. W. Zhang, G. Ding, C. L. Li Chen, and C. Zhang. Generating virtual ratings from chinese reviews to augment online recommendations. *ACM Transactions on Intelligent Systems and Technology*, 4(1), 2013.
32. C.-N. Ziegler, S. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th International World Wide Web Conference*, pages 22–32. ACM, 2005.