Enhancing web-based configuration with recommendations and cluster-based help

Rickard Cöster, Andreas Gustavsson, Tomas Olsson, and Åsa Rudström

Swedish Institute of Computer Science, SICS, Box 1263
SE-164 29 Kista, Sweden
{rick, angus, tol, asa}@sics.se
http://www.sics.se/

Abstract. In a collaborative project with Tacton AB, we have investigated new ways of assisting the user in the process of on-line product configuration. A web-based prototype, RIND, was built for ephemeral users in the domain of PC configuration.
Two mechanisms were added to a commercial configurator produced by Tacton: i) automated recommendations that display social trails associated with the configuration; and ii) a help system based on term clustering. Recommendations based on previous customer selections are made on separate attributes as well as full configurations, i.e. complete PCs. The early rater problem is solved using a probabilistic bootstrapping approach. The help system supports novice users browsing for help information, as well as experienced users able to pose exact queries.

1 Introduction

Configurators assist users in selecting attributes and features such as customer requirements and product attributes of a complex product. In a collaborative project with Tacton AB, a commercial company selling and marketing a constraint-based configurator, we have investigated new and complementary ways of assisting the user in the task of configuring complex products. The result of the project is a prototype, RIND, in which we have added i) automated, collaborative recommendations for displaying social trails associated with the configuration, and ii) a help interface. The prototype is built on top of the Tacton configurator in the domain of PC configuration.

By tagging the configuration process with social information we aim to give the user a better idea of which attributes or complete configurations that are common or not. The recommendations guide the user by displaying two types of social trails. First, the user can get recommendations on selected product attributes. Attributes that are recommended are those that other people have selected in similar situations. Second, the user can ask: Given my current selections, what is the most popular final product configuration?

The help system is designed to adapt to the current user selections in the configuration. For experienced users, detailed help can easily be found with keyword search. For users who are new to the product domain, the structure of the help system functions as a guide to the domain.

Using a rule-based knowledge system, the configurator calculates and displays which attributes are compatible with the user's previous selections. Whenever a choice is incompatible, the configurator will inform the user what needs to be changed in order to keep her most recent selection. The configurator can also select attributes that optimize some product variable, e.g. price. In this way, a user may select the most important product attributes and let the configurator select all the other.

The configurator is good at handling product attributes. Adding recommendations and cluster-based help takes us yet another step closer to the underlying customer needs.

2 The RIND Prototype

Several different user categories can be identified in the domain of on-line PC configuration: ephemeral/frequent user; novice/expert; buying for oneself or for others; customer or sales person (sales support, help desk) etc. In addition, the system could be designed to be stand-alone or a web client; to be accessed over the Internet/extranet/intranet; using a login procedure or not. The design of the system is fundamentally dependent on the choices made for these factors. The RIND prototype is a web client designed for users that only use the system

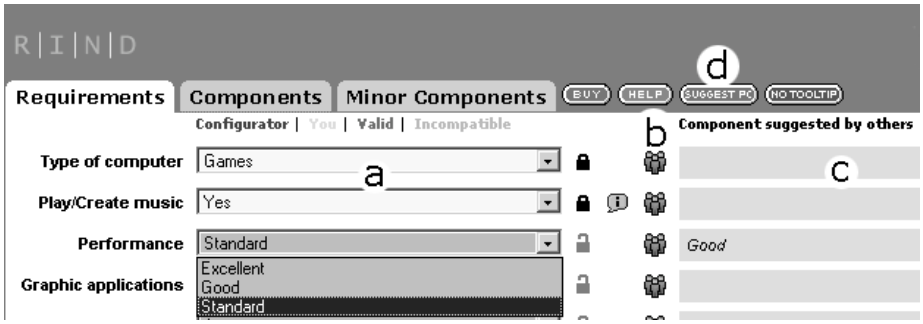

Fig. 1. Interface detail

once or twice. Part of the interface is depicted in Figure 1. Ephemeral users should not be required to log in to use the system until the time of purchase. In light of this, it is hard to anticipate the background of the user - novice or expert? Our hypothesis is that ephemeral users may be unfamiliar to the use of a configurator, but not necessarily unfamiliar with the domain. We also assume that users purchasing products over the Internet can be assumed to be reasonably familiar with computer applications as such.

We believe that it is in this context that recommendations and help interfaces increase the benefit to the user.

Our approach differs from other approaches combining knowledge based reasoning and collaborative filtering in two respects. Firstly, there is no explicit model of user preferences, ratings, or the like, such as discussed in e.g. [1]. Only complete configurations are stored and used as a basis for recommendation. Secondly, our system is not a hybrid system combining the two techniques, as suggested by [10] or [6]. Instead, the results from the recommender engine are presented alongside the results from the constraint-based configurator.

The PC configuration domain is expressed in 35 attributes, listed in Table 1. Each attribute has 2 to 18 values. 12 of the 35 attributes are not settable by the user (e.g. price excl. VAT), and are therefore not used in the recommendation process. 8 attributes represent user requirements (e.g. whether the user will play/create music), and the remaining 15 are computer components. For better overview, the 23 settable attributes are grouped into Requirements, Components, and Minor Components. This grouping is reflected in the interface, where each group is displayed on a separate tab. The demands on the interface are high.

Requirements	*Components*	*Minor components*	*Non-settable attributes*
Type of computer	Service	Mouse	Performance value
Play/Create music	Monitor	CD/DVD	Max days to delivery
Performance	Hard Drive	Keyboard	Price
Graphic applications	OS	Speaker	Price incl.VAT
Minimum component quality	Video Card	Network Card	Total RAM
Minimum RAM size	Extra Video Card	Mother Board	memory cards (5 attrs.)
Minimum hard drive size	Cpu	Computer Case	Free Dimm slots
Delivery		Sound Card	Floppy Drive

Table 1. PC domain attributes

For each attribute, a large number of different possibilities need to be displayed. Each attribute can take on a number of values, displayed as a color coded pull-down menu (a in Fig. 1). The configurator's choice is presented first in the list (blue), followed by valid (green) choices and choices that are incompatible with the user's previous selections (orange). Once a value is selected (yellow), the lock icon to the right of the value list closes.

Next, recommendations can be made on single attributes as well as on full configurations. To maintain a reasonable complexity in the interface, recommendations are separated from the selection of values. Recommendations on single attributes are given on demand (clicking on the icon b in Fig. 1), and are displayed to the right (c in Fig. 1). The most popular full configurations are recommended in a separate window (clicking on icon d). This sub-window has a similar but simplified interface, with the possibility to select all attributes of one of the displayed configurations with a single click.

RIND is an HTML/JavaScript client, communicating with three logical servers: the configurator, the recommender engine, and the help system. Data is commu-

nicated in XML format, and the client makes extensive use of the XML DOM provided by MS Internet Explorer 5.5 and higher.

3 Recommendations

For recommendations, we use a recommender engine developed at SICS. This engine supports both collaborative and content-based filtering, and has previously been used in the EFOL project [9] recommending food recipes, and in GeoNotes, a position-based system for annotation of physical places [2].

At a conceptual level, the architecture of the recommender engine is very simple. Each user's interests are represented in a profile. In the case of collaborative filtering, the profile is usually a feature vector of the user's explicit or implicit votes. In other cases, e.g. content-based filtering, a classifier or regression machine may be trained to learn the user's interest model.

In RIND, each completed and purchased product configuration is represented and stored as a feature vector. The component choices are represented as the feature values. Each choice is interpreted as an implicit vote for the corresponding component value. The current user's configuration is used as a query to the recommender, but is not stored until the final purchase.

For this specific project we have developed three basic algorithms for filtering, which are all based on neighborhood formation and prediction from the neighborhood. The distance between two profiles is the number of features with equal values. We write v_a to denote the profile of user a, and $v_{a,j}$ for the value of feature j in that profile. Predictions are denoted p and use the same subscript notation as profiles. Furthermore, we define $X(a,k)$ as the k nearest neighbors to user a.

3.1 Weighted Majority Voter

The simplest algorithm is the Weighted Majority Voter, which predicts the value of a component on the basis of a weighted majority vote of the k nearest neighbors. The weight is set equal to the distance between two user profiles, i.e. the number of equal component values in the two configurations. The prediction p for user a on item j is then

$$p_{a,j} = \underset{s \in S_j}{\operatorname{argmax}} \sum_i w(a,i)[v_{i,j} = s]$$

where $i \in X(a,k)$, $S_j = \{v_{i,j}\}$ and $w(a,i) = \sum_j [v_{a,j} = v_{i,j}]$.

3.2 Most Popular Choice

The second algorithm, Most Popular Choice predicts the most popular entire configuration of the k nearest neighbors. The algorithm is an extended Naïve Bayes classifier using m-estimate for estimating the probabilities [5]. Because the

prediction regards entire configurations, the Naïve Bayes classifier is extended to handle multiple components. The sought components (components not already chosen by the user) are assumed to be independent.

The m-estimate gives weight to the probability according to the user's current choices even when there is no configuration with all of the user's choices. The extension to many components gives weight to the probability according to the popularity (= number of occurrences) of the sought component values. These two properties lead to the following (good) characteristic; if the user has not chosen any component-value pairs or if no stored configuration has the component-value pairs of the user's choice, then the configuration with the most popular component-value pairs will be recommended. Otherwise, the probability of a configuration is weighted according to whether it has any of the component-value pairs of the user. The prediction is

$$p_a = \operatorname*{argmax}_i \Pr(\{v_{i,u} | u \in \bar{J}\}) \times \prod_{j \in J} \Pr(v_{a,j} | \{v_{i,u} | u \in \bar{J}\})$$

where $J = \{j \mid v_{a,j} \neq \{\}\}$, the set of features for which user a have selected values, and $\bar{J}$ is the set of features for which a has not yet selected any values. We let $\Pr(\{v_{i,u} | u \in \bar{J}\}) = \prod_{u \in \bar{J}} \Pr(v_{i,u})$ by assuming independency.

For the probabilities, we count the frequencies over the neighbors as follows. We let $\Pr(v_{i,u}) = f_u/k$ where f_u is the number of neighbors with component u set to the value $v_{i,u}$. We define $\Pr(v_{a,j} | \{v_{i,u} | u \in \bar{J}\})$ as $(f_{i-a+j} + mp)/(f_{i-a} + m) = (f_{i-a+j} + 1)/(f_{i-a} + k)$ when as m-estimate we use $m = k$ and $p = 1/k$ (the latter is the smallest probability any component value can have except for the zero probability). f_{i-a} is the number of neighbors with the same component-value pairs as v_i except for all components set in v_a. Finally, f_{i-a+j} is the number of neighbors with the same component-value pairs as v_i except for all components in v_a but component j that has value $v_{a,j}$ (that is $v_{i,j} = v_{a,j}$). This means that $f_{i-a+j} > 0$ only if $v_{i,j} = v_{a,j}$.

3.3 Naïve Bayes Voter

The third and last algorithm is the Naïve Bayes Voter, which is similar to the Weighted Majority Voter but uses Naïve Bayes with m-estimate as a basis for the predictions. Instead of using the sum of the distances between the user profiles as a vote, it uses the probability of a component value given the k nearest neighbors. Thus the most probable value for any component is recommended. As m-estimate we use again $m = k$ and $p = 1/k$. The m-estimate makes it possible to recommend values for all components regardless of whether any other user profile contains all the component-value pairs of the active user.

$$p_{a,j} = \operatorname*{argmax}_{s \in S_j} \Pr(v_{i,j} = s) \prod_{s' \in v_a} \Pr(v_{a,t} = s' | v_{i,j} = s)$$

where again $S_j = \{v_{i,j}\}$. For the frequencies, we define $\Pr(v_{i,j} = s) = f_{j=s}/k$ where $f_{j=s}$ is the number of neighbors with value s for feature j. Furthermore,

we let $\Pr(v_{a,t} = s' | v_{i,j} = s) = (f_{j=s \wedge t=s'} + 1)/(f_{j=s} + k)$ where $f_{j=s \wedge t=s'}$ is the number of neighbors with both value s for feature j and value s' for feature t.

4 Bootstrapping

When starting up any recommender system every developer will be faced with the early-rater problem. Without any users it is hard to make good recommendations. Even though there are many users, making good recommendations can be hard when each item has only a few user opinions and the overlap between the opinions is sparse.

The ordinary approach to tackle this problem is to combine entirely opinion based collaborative filtering with content-based methods, expecting the two approaches to level out each other's weaknesses [8]. In this project, the content is already used as the base for recommendations and hence another approach is required. We propose an approach that makes use of prior knowledge acquired from asking experts, analyzing the recommended items, and using prejudices and good guesses. The acquired knowledge is often uncertain and thus represented with a probabilistic model. Simulated user profiles can then be sampled from the probabilistic model and bootstrap any used learning algorithm including the k nearest neighbor.

4.1 The probabilistic model

The current RIND implementation cannot track individual users and thus only dependencies between attributes and not between users are modeled. The prior knowledge is encoded as a Bayesian belief net, which can be constructed with any schoolbook procedure for knowledge acquisition [4]. The prior knowledge modeled in the belief net is based on:

- Normally, users prefer cheap products.
- People who want to play games usually need graphics and music.
- An interest in graphics indicates a need for a good monitor and graphics card.
- An interest in music indicates a need for a good sound card and speakers.
- An interest in both graphics and music indicates a need for a larger RAM memory, a larger hard disk and good performance.
- For variables without good value distribution guesses, uniform probabilities are used. For example, each type of computer (standard/build your own/games/business) gets a uniform probability of 0.25.

User profiles used for bootstrapping the system are then sampled from the belief net by probabilistically selecting feature values in random order. Only values validated by the configurator are considered. If there is no selectable valid value for a feature, the user profile is discarded and instead a new profile is formed. Every time a value is selected for a feature the posterior probability of the belief

net is recomputed given the currently selected feature values. The procedure is then repeated until there are no more selections left for each user profile.

The proposed approach of course makes it harder to discern the meaning of a recommendation, at least when the recommendations are based on very few real users. A recommendation is no more only about whether other people liked or disliked an item but also what the system designer recommends. This makes the problem of visualizing a recommendation more complicated. We could partly get around this problem by distinguishing between recommendations based on real user profiles and sampled profiles in the user interface.

5 RIND Help System

The key requirement for the help system is to guide ephemeral users to those help documents that meet their information request, independent of previous knowledge of the PC domain. Requirements such as maintaining consistent help for newly released product information and integration with the configuration interface are also considered as central.

To support ephemeral users with varying domain knowledge, the help system needs a navigation tool that is suitable for novices as well as for experts. Our approach presents a hierarchic view of the domain with terms that are more general or more specific in relevance to each other based on the user query. A novice can browse the domain and refine the query by selecting a more specific or more general term in the hierarchic view. By entering a precise query an expert can search for help documents that fit the information request, but also refine the query using the hierarchic view.

5.1 The Help Web Interface

The help interface (fig. 2) is separated into three parts. The user query is displayed at the top as an editable input box. In addition to input from the user, the input box is also used to display the current hierarchic location in the help domain. In the bottom part, the hierarchic view can be found to the left, while the help text itself is displayed to the right. Some selectable terms are embedded in the text and have the same functionality as a selectable spread topic.

In the hierarchical view, more general terms are displayed under the Spread Topics header. If the user selects the spread topic "PC" in figure 2 the help interface is updated and "PC" becomes the new user query. In this case the user will see how the previous query term "motherboard" is related to other topics also related to PC.

Terms displayed under the Narrow Topics header are terms that are subtopics to the current user query. When the user selects the narrow topic "perfomance" the term is added to the user query and the interface is updated accordingly.

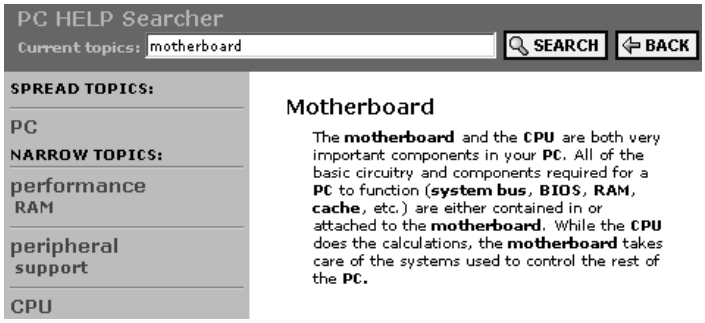

Fig. 2. Help interface

5.2 Clustering Method

Van Rijsbergen [11] defines two types of automated methods for associating documents with each other. In the polythetic clustering method the membership between a document and the cluster is based on how many of the document terms that are in cluster defined terms. The problem with polythetic methods is that not all the terms defining the clusters are guaranteed to be present in the documents.

In the monothetic clustering method, all terms defined by the cluster are guaranteed to exist in all documents. A common form of monothetic cluster hierarchy is taxonomic web directories such as Yahoo!. These directories manually place web pages into a hierarchical structure. One problem with this approach is that most texts are not taxonomically related, but discuss different topics simultaneously. A second problem is that the web pages have to be manually placed into the structure. Finally, users have to learn and remember the structure of the hierarchy.

Sanderson and Croft [7] attempt to overcome these problems by automatically constructing a concept hierarchy derived from a set of documents. They extracted terms for the hierarchy from the documents such that a parent term would refer to a more general concept than its child.

The RIND help system uses a similar approach but with predefined terms describing each help text, and a monothetic clustering method with additional polythetic clustered terms describing the monothetic term cluster. The predefined terms are based on a controlled dictionary without any synonyms, since the attempt is to solve the polysemy part of the vocabulary problem [3].

6 Concluding Remarks

We have identified a domain in which recommendations and cluster-based help systems seem well suited to assist ephemeral users in the task of selecting product

configurations. To this end, we have built a prototype on top of a configurator to demonstrate our approach.

We found several ways of performing recommendations by using the database of purchased products, and we have also developed a principled way of bootstrapping recommender systems with prior knowledge and good guesses.

A key requirement for the prototype was that the help system should be able to support both novice and expert users. We have therefore developed a cluster-based help system in which experts can find exact information and novice users can see a hierarchical structure of how components or parts of the configuration are related.

7 Acknowledgements

This work was performed in a collaborative project between Tacton AB (`www.tacton.se`) and the Swedish Institute of Computer Science, SICS (`www.sics.se`), supported by funding from VINNOVA, the Swedish Agency for Innovation Systems.

References

1. L. Ardissono, A. Felfernig, D. Jannach, G. Friedrich, R. Schäfer, and M. Zanker. Customer-adaptive and distributed online product configuration in the CAWICOMS project. In *Proceedings of the Configuration Workshop at IJCAI 01*, Seattle, USA, 2001.
2. F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund. Geonotes: Social and navigational aspects of location-based information systems. In Abowd, Brumitt, and Shafer, editors, *Ubicomp 2001: Ubiquitous Computing, International Conference*, pages 2–17, Atlanta, Georgia, 2001. Berlin: Springer.
3. G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.
4. A. Gonzalez and D. Dankel. The engineering of knowledge-based systems: Theory and practice, 1993.
5. T. Mitchell. *Machine Learning*. New York: McGraw Hill, 1997.
6. Martin Molina. An intelligent sales assistant for configurable products. In Ning Zhong, Yi Yu Yao, Jiming Liu, and Setsuo Ohsuga, editors, *Proceedings of the 1st Asia-Pacific conference on Web Intelligence: Research and Development*, volume 2198 of *Lecture Notes in Computer Science*. Springer, 2001.
7. M. Sanderson and W. B. Croft. Deriving concept hierarchies from text. In *Research and Development in Information Retrieval*, pages 206–213, 1999.
8. Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jonathan L. Herlocker, Bradley N. Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Computer Supported Cooperative Work*, pages 345–354, 1998.
9. M. Svensson, K. Höök, J. Laaksolahti, and A. Waern. Social navigation of food recipes. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 341 – 348, Seattle, Washington, 2001.

10. Thomas Tran and Robin Cohen. Hybrid recommender systems for electronic commerce. In *Papers from the Seventeenth National Conference on Artificial Intelligence (AAAI-2000) Workshop on Knowledge-Based Electronic Markets*, pages 78–84. AAAI Press, 2000.
11. C. J. Van Rijsbergen. *Information Retrieval*. Dept. of Computer Science, University of Glasgow, 1979.