PeopleViews: Human Computation for Constraint-Based Recommendation

Alexander Felfernig
Graz University of Technology
A-8010 Graz, Austria
afelfern@ist.tugraz.at

Thomas Ulz
Graz University of Technology
A-8010 Graz, Austria
thomas.ulz@student.tugraz.at

Sarah Haas
Graz University of Technology
A-8010 Graz, Austria
sarah.haas@student.tugraz.at

Michael Schwarz
Graz University of Technology
A-8010 Graz, Austria
michael.schwarz@student.tugraz.at

Stefan Reiterer
Graz University of Technology
A-8010 Graz, Austria
stefan.reiterer@ist.tugraz.at

Martin Stettinger
Graz University of Technology
A-8010 Graz, Austria
mstettinger@ist.tugraz.at

ABSTRACT

In this paper we introduce PEOPLEVIEWS which is a constraint-based recommendation environment based on the concepts of Human Computation. We show how recommendation knowledge can be collected from users of the recommendation environment on the basis of micro-tasks and how PEOPLEVIEWS exploits this knowledge for recommendations. In this context, we compare different recommendation approaches using a set of DSLR cameras.

Keywords

Constraint-based Recommendation, Human Computation

1. INTRODUCTION

In contrast to collaborative and most content-based approaches, constraint-based recommender systems (as a specific type of knowledge-based recommender system) [2] rely on a predefined set of explicit constraints that perform the selection of candidate items for a user. These systems are applied in more complex item domains where a user specifies a set of criteria (search criteria or critiques) and the system proposes a corresponding set of recommendations. Items in these domains are purchased less frequently which makes more traditional recommendation approaches such as collaborative filtering less applicable [2]. Constraint-based recommendation sessions are typically conversational, i.e., the user answers questions and the system tries to retrieve relevant items.

Constraint-based recommender systems are applied in various item domains. Peischl et al. [9] present an application of constraint-based recommendation technologies for supporting the selection of software testing methods. Felfernig et al. [3] report applications of these technologies in the domain of financial services and electronic equipment. Leitner et al. [6] show the application of these technologies for recommending smart home solutions to users. WEEVIS[1] [10] is a constraint-based environment embedded in MediaWiki[2] which allows the definition of recommender applications with a wiki-style syntax and the integration of these recommenders into standard Wiki pages. Finally, Torrens et al. [11] introduce an environment for the planning and recommendation of travel itineraries which is based on soft constraint technologies used for determining preferred solutions.

Development processes related to constraint-based recommender systems often come along with the so-called *knowledge acquisition bottleneck* where knowledge engineers are overwhelmed by the increasing size and complexity of knowledge bases. In this paper we show how Human Computation concepts [12] can be employed to extract recommendation knowledge directly from users and domain experts and thus to lift the burden of effortful engineering from the knowledge engineer's shoulders. The major idea of Human Computation [12] is to let humans perform simple and shortterm tasks they are better in compared to computers – in the context of knowledge engineering scenarios, the overall idea is to let domain experts perform simple and shortterm knowledge engineering tasks they are much better in compared to knowledge engineers. The idea of PEOPLEVIEWS is to engage domain experts more deeply into knowledge engineering processes by making the corresponding engineering tasks more accessible to them. This is achieved via so-called *micro-tasks* that are basic tasks which can be completed within a short period of time. User feedback on micro-tasks is exploited for deriving recommendation rules (filter constraints) that help to select items.

The potential advantages of applying PEOPLEVIEWS technologies are less efforts related to recommendation knowledge base development and maintenance, fewer erroneous constraints (compared to knowledge engineers, domain experts know more about the item domain), and a significantly higher degree of scalability, i.e., more recommenders can be maintained in parallel (which could not be achieved by a small number of knowledge engineers).

The idea of exploiting the concepts of Human Computation [12] in the context of recommender systems development is not new. Hacker and von Ahn [4] introduce the MATCHIN environment where the elicitation of preferences from users is based on the idea of asking questions of which decisions other users would take when being confronted with a set of items. Walsh and Golbeck [13] introduce a game environment (CURATOR) which can be used to configure item collections. The overall goal is, for example, to enforce users to develop a common understanding of a reasonable item collection. Finally, Larson et al. [5] introduce the idea of not only recommending items to users but also users to items in situations

[1] www.weevis.org

[2] www.mediawiki.org

The work presented in this paper has been conducted within the scope of the PEOPLEVIEWS project funded by the Austrian Research Promotion Agency (843492).
CrowdRec 2015 19 Sep. 2015, Vienna, Austria

attribute	explanation	choice type	question to user	domain
application	application domains	single	Preferred Application?	{sport, architecture, macro, landscape, portrait}
usertype	photography experiences	multiple	Suited for whom?	{beginner, amateur, expert}
usability	usability of digital camera	single	Minimum accepted usability?	{average, high, very high}

Table 1: Example *user attributes* ($u \in U$) of a PEOPLEVIEWS digital camera recommender.

id	user attribute	value
req_1	application	sport
req_2	usertype	expert
req_3	usability	very high

Table 2: Example user requirements $req_i \in REQ$.

where additional evaluations are needed in order to deal with sparse data and thus to improve recommendation quality. In the line of the term *user-item reciprocity* introduced by Larson et al. [5], PEOPLEVIEWS does not only support the recommendation of items but also the recommendation of users who should provide item evaluations needed to improve recommendation quality.

The remainder of this paper is organized as follows. In Section 2 we show how recommender knowledge bases can be defined in the PEOPLEVIEWS environment. Thereafter we provide an overview of the current version of the PEOPLEVIEWS user interface (Section 3). In Section 4 we compare recommendation approaches currently implemented in PEOPLEVIEWS with regard to their predictive quality. In Section 5 we provide an overview of ongoing and future work. With Section 6 we conclude the paper.

2. RECOMMENDATION APPROACH

When applying a PEOPLEVIEWS recommender, users are specifying their preferences (requirements) in terms of user attribute value preferences that are in the following translated into corresponding item recommendations. An example set of *user attributes* ($u \in U$) is depicted in Table 1: *application* (single-valued) represents the preferred application of a digital camera, *usertype* specifies the type of user the camera is suited for, and *usability* specifies the minimum degree of usability a user is willing to accept. Each user attribute has an associated choice type (single or multiple choice), a question that is posed to the user, and an attribute domain definition (enum, integer, boolean, and text). An example set of user requirements ($req_i \in REQ$), i.e., instantiations of user attributes specified by a user, is depicted in Table 2.

In addition to user attributes, *product attributes* describe technical properties of an item. Table 3 depicts a set of example product attributes of a digital camera which include *sensorsize*, *maxshutterspeed*, *maxISO*, *maxresolution*, and *price* (a corresponding item set is depicted in Table 4). In addition to the choice type and the question posed to a recommender user (technical properties of items can also be specified via a search interface), each product attribute is associated with a corresponding *attribute level similarity metric*. The metrics can be exploited to determine items that are similar compared to the criteria (product attribute properties) specified by the user. In this context, the *equal is better* (EIB) denotes the fact that two attribute values are only similar if they have the same value; *nearer is better* (NIB) denotes the fact that the lower the distance between two attribute values, the better. Furthermore, *more is better* (MIB) denotes the fact that the higher the value of an item the better, and *less is better* (LIB) denotes the fact that the lower the value the better. These metrics will be explained in more detail in the context of the PEOPLEVIEWS recommendation strategies.

User attributes as well as product attributes can only be defined by the creator of a recommender application – related requests for additional attributes or changes in the domain of an attribute can be posed in the PEOPLEVIEWS forum. Users of the PEOPLEVIEWS community are free to integrate additional items – in this context, each item (product) attribute value has to be specified. PEOPLEVIEWS users are also in charge of specifying the relationship between user attributes and items. This is achieved through the completion of *micro-tasks*, for example, a user has to estimate to which extent a digital camera supports users who are beginners in digital photography (i.e., *usertype = beginner*). The "wisdom of the crowd" is then exploited for deriving a corresponding recommendation rule (filter constraint), for example, if the majority of users provide a low evaluation for *usertype = beginner* for the digital camera *Canon EOS 5D Mark III*, the corresponding recommender will not regard the camera as best candidate for beginners. Users of a PEOPLEVIEWS recommender can add recommendation knowledge via assigned micro-tasks or proactively.

PEOPLEVIEWS supports two interaction modes. First, PEOPLEVIEWS recommenders support users in finding a product (item) that fits their wishes and needs (*recommendation mode*) – in this context, user and product attributes are used to specify user (customer) requirements $req_i \in REQ$. Second, recommenders can be developed in the *modeling mode*. In this mode, user attributes are central elements used in a micro-task: given a certain item, users estimate which values of a user attribute fit the item to which extent. There are different types of micro-tasks used to figure out which user attribute settings are compatible with which items – details on the different types of PEOPLEVIEWS micro-tasks will be given in Section 3. An example of the evaluation of items with regard to the given set of user requirements is depicted in Table 5 – in PEOPLEVIEWS, only logged-in users are allowed to enter evaluation data (one evaluation per item and user attribute). For example, *Giselle* rates the *macro* capabilities of item Φ_3 as very high (individual user support: 1.0). User-individual support values indicate on a scale 0..1 the degree to which an item supports a given attribute value from a specific user's point of view.

Each entry in Table 5 represents one *user-specific filter constraint* which specifies selection criteria for one item from the viewpoint of an individual user. For example, user *Giselle* evaluates the *Canon EOS 5D Mark III* (item Φ_3) as an excellent candidate (individual user support: 1.0) for *macro* photography. Furthermore, Giselle thinks that the camera should only be recommended to experts (individual user support: 1.0), the usability of the camera is regarded as *high* (individual user support: 1.0). Since user-specific filter constraints only reflect the views of individual users, these constraints have to be aggregated. *Recommendation-relevant filter constraints* are aggregated user individual filter constraints – examples of this aggregation step are depicted in Table 6. For Φ_1, all related user-specific filter constraints are integrated into one corresponding recommendation-relevant filter constraint. The user-individual support values $s(\Phi, a, v)$ (see Table 5) for an item Φ, user attribute a, and user attribute value v are aggregated into one *support* value (see Formulae 1 and 2).

attribute	choice type	question to recommender user	domain	similarity metric	show to user?
sensorsize	single	Preferred sensor size?	{fullframe, APS-C, MFT, 1", 2/3"}	EIB	yes
max-shutterspeed	single	Required max. shutter speed?	{1/4.000, 1/6.000, 1/8.000, 1/16.000}	MIB	no
maxISO	single	Required max. ISO sensitivity?	{6.400, 12.800, 25.600}	MIB	no
max-resolution	single	Max. resolution?	{8 Mpix, 12 Mpix, 16Mpix, 18Mpix, 21Mpix, 23Mpix}	MIB	yes
price	single	Max. price?	integer	LIB	no

Table 3: Example *product attributes* ($p \in P$) of a PEOPLEVIEWS digital camera recommender.

item id	item	sensorsize	maxshutterspeed	maxISO	maxresolution	price
Φ_1	Canon EOS 7D	APS-C	1/8000	12.800	18Mpix	800
Φ_2	Canon EOS 550D	APS-C	1/4000	6.400	18Mpix	500
Φ_3	Canon EOS 5D Mark III	fullframe	1/8.000	25.600	23Mpix	2.000

Table 4: Example set of digital cameras defined using the product attributes of Table 3.

$$support(\Phi, u, v) = \frac{\Sigma s(\Phi, u, v)}{|s(\Phi, u, v)|} \times popularity(\Phi, u, v) \quad (1)$$

$$popularity(\Phi, u, v) = \frac{|s(\Phi, u, v)|}{|s(\Phi, u)|} \quad (2)$$

The recommendation-relevant filter constraints of Table 6 have to be interpreted, for example, as follows: item Φ_1 (Canon EOS 7D) is included (recommended) if the specified requirements regarding the user attributes *application*, *usertype*, and *usability* are consistent with the condition in the corresponding recommendation-relevant filter constraint: $application \in \{sport, portrait\} \wedge usertype \in \{beginner, amateur, expert\} \wedge usability \in \{high, very\ high\} \rightarrow include(\Phi_1)$. This aggregation step has to be performed for each individual item – in our working example, the result of this aggregation step are three different recommendation-relevant filter constraints (each item has exactly one) which are then executed on the item table.

The logical representation of a recommendation-relevant filter constraint f for an item Φ is depicted in Formula 3. In this formula, $values(\Phi, u)$ is the set of supported domain values ($support > 0$) of user attribute $u \in U$, furthermore, $noval$ denotes the fact that no value has been selected for the corresponding user attribute.

$$f(\Phi) : \bigwedge_{u \in U} u \in values(\Phi, u) \cup \{noval\} \rightarrow \text{include}(\Phi) \quad (3)$$

Depending on the requirements articulated by the user ($req_i \in REQ$), PEOPLEVIEWS identifies and ranks items as follows. First, all recommendation-relevant filter constraints f are used to preselect items (identification of a candidate set). On the basis of the user requirements defined in Table 2, the items $\{\Phi_1, \Phi_3\}$ can be identified as candidate items. The following utility function (Formula 4) can now be used to rank the items in the candidate set. This way, the utility of each item is determined on the basis of the support values of individual requirements (see Formula 1). The function $w(a)$ denotes an attribute weight which has been specified by a user or has been learned on the basis of already existing user interaction logs with the recommender.

$$utility(\Phi, REQ) = \Sigma_{a=v \in REQ}\ support(\Phi, a, v) \times w(a) \quad (4)$$

The item ranking in our working example determined on the basis of Formula 4 is the following. The utility of item Φ_1 is 0.78 + 1.0 + 0.39 = 2.17 whereas the utility of item Φ_3 is 0.16 + 0.97 + 0.56 = 1.69. Consequently, item Φ_1 is ranked before item Φ_2.

Up to now we did not take into account user requirements related to item properties, i.e., we discussed an example scenario that only takes into account the user requirements depicted in Table 2. If user requirements ($req_i \in REQ$) should also include requirements regarding item properties, we have to define the calculation of support values for items Φ, product attributes p, and product attribute values v (see Formula 5). In PEOPLEVIEWS, such support values are determined, for example, on the basis of the attribute level similarity metrics *Equal Is Better (EIB)*, *Nearer Is Better (NIB)*, *More Is Better (MIB)*, and *Less Is Better (LIB)*.

$$support(\Phi, p, v) = \begin{cases} 1 \text{ if } v = val(\Phi, p), 0 \text{ otherwise} & \text{EIB} \\ 1 - \frac{|v - val(\Phi,p)|}{max(\Phi,p) - min(\Phi,p)} & \text{NIB} \\ \frac{val(\Phi,p) - min(\Phi,p)}{max(\Phi,p) - min(\Phi,p)} & \text{MIB} \\ \frac{max(\Phi,p) - val(p)}{max(\Phi,p) - min(\Phi,p)} & \text{LIB} \end{cases} \quad (5)$$

3. PEOPLEVIEWS USER INTERFACE

In the PEOPLEVIEWS *recommendation mode*, users can specify their requirements (see Figure 1) and PEOPLEVIEWS recommenders determine recommendations on the basis of recommendation-relevant filter constraints and the utility function (see Formula 4). Items in the recommendation list can be selected for product comparison, search criteria entered by the user can be saved, i.e., are taken into account when the user activates the recommender the next time.

Recommenders are created in the PEOPLEVIEWS *modeling mode*. Figure 2 depicts an interface for the acquisition of a user-specific filter constraint for the item Canon EOS 7D (Φ_1). The user can evaluate the item with regard to all existing user attributes, however, if the user does not have the knowledge, the corresponding attributes can be omitted. The user input can be directly translated into a corresponding user-specific filter constraint (see, e.g., Table 5). The interface depicted in Figure 2 can be used when a new item is added or a user wants to evaluate an item.

In the PEOPLEVIEWS modeling mode, there are five different micro-task interfaces (of different complexity) that support the ac-

id	user	item id	application	usertype	usability
1	Jenny	Φ_1	sport (1.00)	amateur (0.60), expert (1.00)	high (0.95)
2	Giselle	Φ_1	sport (0.95)	amateur (0.50), expert (1.00)	high (1.0)
3	Sarah	Φ_1	sport (0.95)	amateur (0.75), expert (1.00)	very high (1.0)
4	David	Φ_1	portrait (0.80)	beginner (0.00), amateur (0.10), expert (1.00)	high (0.95)
5	Mike	Φ_1	sport (1.00)	beginner (0.40), amateur (0.80), expert (1.00)	very high (0.95)
6	Jenny	Φ_2	landscape (0.75)	beginner (1.00), amateur (0.80), expert (0.00)	high (0.95)
7	Giselle	Φ_2	sport (0.60)	beginner (0.90), amateur (0.00)	high (0.90)
8	Sarah	Φ_2	landscape (0.80)	beginner (1.00)	average (1.00)
9	David	Φ_2	portrait (0.70)	beginner (0.90), amateur (0.90)	high (1.00)
10	Mike	Φ_2	portrait (0.75)	beginner (1.00)	very high (1.00)
11	Jenny	Φ_3	sport (0.80)	beginner (0.10), amateur (0.60), expert (1.00)	very high (1.00)
12	Giselle	Φ_3	macro (1.00)	expert (1.00)	high (1.00)
13	Sarah	Φ_3	macro (0.90)	expert (1.00)	very high (0.90)
14	David	Φ_3	macro (0.95)	beginner (0.30), amateur (0.40), expert (0.90)	very high (0.90)
15	Mike	Φ_3	portrait (0.95)	amateur (0.00), expert (0.95)	high (0.95)

Table 5: Evaluations of cameras in PEOPLEVIEWS – the numbers in brackets denote user-individual support values (s).

constraint	item name (id)	application	usertype	usability
f_1	Canon EOS 7D (Φ_1)	sport (0.78), portrait (0.16)	beginner (0.08), amateur (0.55), expert (1.0)	high (0.58), very high (0.39)
f_2	Canon EOS 550D (Φ_2)	landscape (0.31), sport (0.12), portrait (0.27)	beginner (0.96), amateur (0.34)	average (0.2), high (0.57), very high (0.2)
f_3	Canon EOS 5D Mark III (Φ_3)	sport (0.16), macro (0.57), portrait (0.19)	beginner (0.08), amateur (0.2), expert (0.97)	high (0.39), very high (0.56)

Table 6: Example of *recommendation-relevant filter constraints* derived from user-specific filter constraints (see Table 5). User-individual support values (Table 5) are integrated into support values. If a support value = 0, it is not part of the filter constraint.

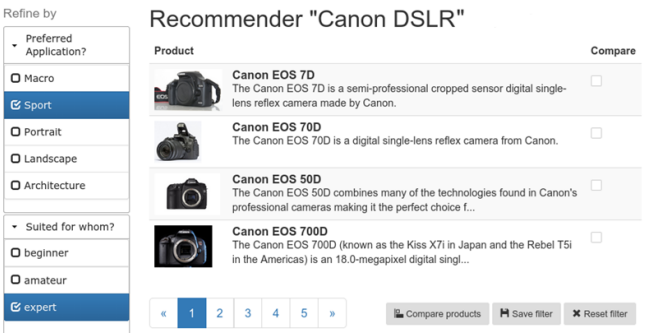

Figure 1: PEOPLEVIEWS user interface: user requirements ($req_i \in REQ$) are specified on the left hand side, the corresonding recommendations are shown on the right hand side.

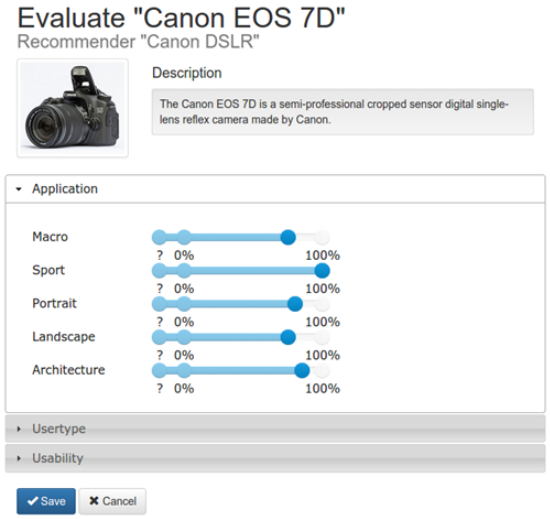

Figure 2: Evaluation of item *Canon EOS 7D* with regard to all user attributes $u \in U$.

quisition of knowledge relevant for deriving user-specific filter constraints (including the corresponding support values). An overview of these micro-tasks is given in Table 7. Figure 3 depicts a simple micro-task interface which supports the acquisition of only one aspect of a user attribute. In this example, the user expresses his/her personal opinion that the item *Canon EOS 7D* has a very low *support* regarding the attribute value *usertype=beginner*.

Similar to Figure 3, the user interface of Figure 4 allows the evaluation of item-specific properties. In addition, the interface supports the acquisition of performance relationships between items regarding specific attribute values, for example, information regarding the camera performance in the *sports* application area. Figure 5 depicts an example of an interface that supports the evaluation of an item with regard to one single choice user attribute (user attribute *usability*). Figure 6 supports the same functionality for multiple-choice answers (user attribute *usertype*). Finally, Figure 7 depicts the user interface of a *Captcha*-style micro-task which helps to figure out whether the current user is a "real" user or a bot.

One source of recommendation knowledge in PEOPLEVIEWS are user evaluations of item properties performed within the scope of completing micro-tasks. Another source of recommendation knowledge are games where PEOPLEVIEWS users can check their personal item domain knowledge. Users can either play in groups of two users or in a single-user mode. Two users receive points if they manage to agree on an answer to a question (without seeing the answer of the other user) – the less iterations are needed for

id	description	figure
1	Evaluation one user attribute with regard to one specific value	3
2	Selection and evaluation of best-performing item with regard to a specific user attribute value	4
3	Evaluation of one single-choice user attribute with regard to all possible values	5
4	Evaluation of one multiple-choice user attribute with regard to all possible values	6
5	Captcha-style check	7

Table 7: Micro-task types supported in PEOPLEVIEWS.

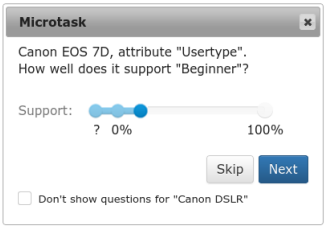

Figure 3: Evaluation of *Canon EOS 7D* with regard to a specific user attribute value (*usertype=beginner*).

achieving the goal, the better. Questions are always related to topics out of the product (recommender) domain (e.g., Canon DSLR cameras), the game is assigned to – users have to pre-select the topic before a game can be started. In the single-user mode, the goal is to answer questions the same way as already done by the majority of other PEOPLEVIEWS users confronted with the same question (posed, e.g., in micro-tasks or prior games). An example of a game in single-user mode is given in Figure 8. Answers of users (including their specification of the support level) provided within the scope of a gaming session are used in the same way as user answers to micro-tasks.

4. EVALUATION

In order to evaluate the predictive performance of the recommendation approach presented in this paper, we performed an initial evaluation on the basis of a dataset collected with a WEEVIS [10] DSLR recommender that used the same user attributes, product attributes, and items as included in the corresponding PEOPLEVIEWS recommender (the constraints in WEEVIS where defined by an expert in the domain of digital cameras). The digital cameras in the WEEVIS recommender were ranked according to the degree the requirements were fulfilled by the corresponding recommendation (item). The dataset includes N=356 different sessions where users first defined their requirements and then selected a corresponding digital camera they were interested in. In PEOPLEVIEWS, 10 users (computer science students) evaluated the items w.r.t. seven different attributes. We compared the recommendation approaches currently integrated in PEOPLEVIEWS with the baselines of *random recommendation* and *popularity-based* recommendation.

The *first* PEOPLEVIEWS approach is utility-based recommendation with the assumption of equally distributed attribute weights (see $w(a)$ in Formula 4). The *second* approach includes a learning component that derives an item-independent attribute importance. The component is based on the implementation of a genetic algorithm where individuals are represented by vectors of item weights. We measured the prediction quality of the recommendation approaches in terms of *precision*. The results of this initial evaluation show that the utility-based version with learned weights clearly outperforms the baseline versions as well as the utility-based approach with equally distributed weights (see Figure 9).

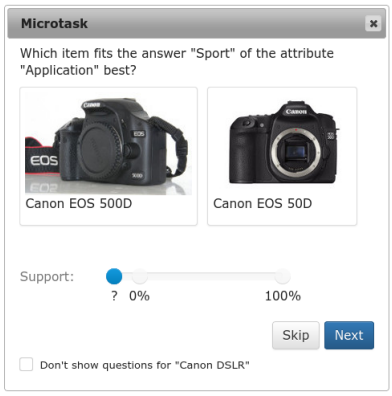

Figure 4: Evaluation of an item set with regard to a specific user attribute value. Preference relationships regarding item performance are derived as well (e.g., a camera is better with regard to *application=sport*).

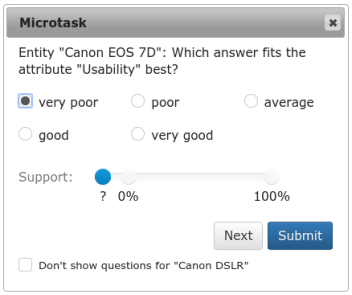

Figure 5: Evaluation of *Canon EOS 7D* with regard to a single-choice attribute (*usability*).

5. ONGOING AND FUTURE WORK

Recommendation Approaches. The comparison with further recommendation approaches is within the scope of our future work. Using user interaction logs we will compare utility-based approaches with different approaches to case-based reasoning (CBR) and probability-based recommendation [7]. In this context, we will also compare ensemble-based (hybrid) approaches with regard to their capability of improving prediction quality. Finally, we want to expand the way in which PEOPLEVIEWS concepts can be applied in taste domains, where there is not a unique, objective truth about items, for example, in the context of collaborative filtering recommendation scenarios (see, e.g., [5, 8]).

Micro-task scheduling. Recommendation-relevant filter constraints are derived by micro-tasks. Currently, we are working on the integration of scheduling concepts that automatically assign micro-tasks to users – this is in the line of *user-item reciprocity* [5]. The assignment is based on content-based recommendation approaches where the available expertise derived from the user profile is matched with the required expertise derived from item descriptions including associated tags and forum discussions. The assignment of users to micro-tasks is represented as an optimization problem where the goal is to find the "optimal" user for a micro-task while not overloading individual users.

Quality assurance. While micro-task scheduling deals with the assignment of micro-tasks to users, quality assurance in PEOPLEVIEWS focuses on the analysis of existing micro-task contributions and the derivation of corresponding new micro-tasks that are needed to improve dataset quality and also to prevent efforts that

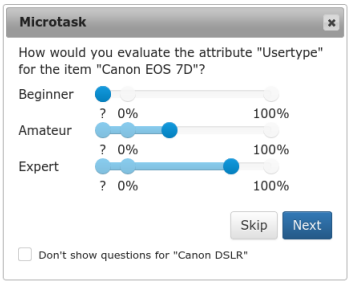

Figure 6: Evaluation of *Canon EOS 7D* with regard to a multiple-choice attribute (*usertype*).

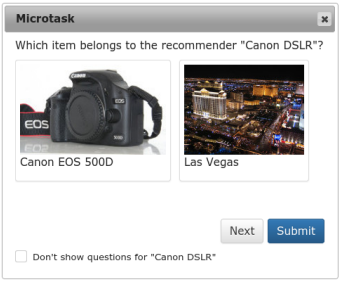

Figure 7: CAPTCHA-style micro-task.

have the goal to manipulate the outcome of a recommender session, for example, in terms of item inclusion and ranking. Currently, we develop and compare different approaches to anomaly detection [1] to identify dataset regions where additional user feedback is needed for improving the data quality.

Dealing with Inconsistencies. Upcoming versions of PEOPLEVIEWS will include the FASTDIAG algorithm (see, e.g., [10]) which supports the efficient determination of preferred minimal diagnoses in the case of inconsistent requirements. A diagnosis in this context is represented by a minimal set of requirements that need to be changed such that the system can identify a solution.

6. CONCLUSIONS

In this paper we provide an overview of the PEOPLEVIEWS environment which supports the Human Computation based development of constraint-based recommenders. PEOPLEVIEWS includes five types of micro-tasks that are used for collecting recommendation knowledge. We analyzed the prediction quality of initially integrated utility-based recommendation approaches. Major functionalities to be included in upcoming versions of the system are intelligent micro-task scheduling, quality assurance, and automated repair mechanisms for inconsistent requirements.

7. REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, 2009.

[2] A. Felfernig and R. Burke. Constraint-based recommender systems: Technologies and research issues. In *IEEE ICEC'08*, pages 17–26, Innsbruck, Austria, 2008.

[3] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. An environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce (IJEC)*, 11(2):11–34, 2006.

[4] S. Hacker and L. von Ahn. Matchin: Eliciting User Preferences with an Online Game. In *CHI'09*, pages 1207–1216, 2009.

[5] M. Larson, A. Said, Y. Shi, P. Cremonesi, D. Tikk, and A. Karatzoglou. Activating the crowd: Exploiting user-item reciprocity for recommendation. In *CrowdRec'13*, pages 1–2, Hong Kong, China, 2013.

[6] G. Leitner, A. Fercher, A. Felfernig, and M. Hitz. Reducing the Entry Threshold of AAL Systems: Preliminary Results from Casa Vecchia. In *13th Intl. Conference on Computers Helping People with Special Needs*, pages 709–715, 2012.

[7] C. Musto, G. Semeraro, P. Lops, M. DeGemmis, and G. Lekkas. Financial Product Recommendation through Case-based Reasoning and Diversification Techniques. In *ACM RecSys*, pages 1–2, Foster City, CA, USA, 2014.

[8] P. Organisciak, J. Teevan, S. Dumais, R. Miller, and A. Kalai. Personalized human computation. In *HCOMP'13*, pages 56–57, Palm Springs, CA, USA, 2013.

[9] B. Peischl, M. Zanker, M. Nica, and W. Schmid. Constraint-based Recommendation for Software Project Effort Estimation. *Journal of Emerging Technologies in Web Intelligence*, 2(4):282–290, 2010.

[10] S. Reiterer. An Integrated Knowledge Engineering Environment for Constraint-based Recommender Systems. In *FinRec'15*, pages 11–18, Graz, Austria, 2015.

[11] M. Torrens, P. Hertzog, L. Samson, and B. Faltings. reality: a scalable intelligent travel planner. In *ACM Symp. on Applied Computing*, pages 623–630, Melbouurne, FL, USA, 2003.

[12] L. von Ahn. Human Computation. In *Technical Report CM-CS-05-193*, 2005.

[13] G. Walsh and Jennifer Golbeck. Curator: A Game with a Purpose for Collection Recommendation. In *CHI 2009*, pages 2079–2082, Boston, Massachusetts, USA, 2009.

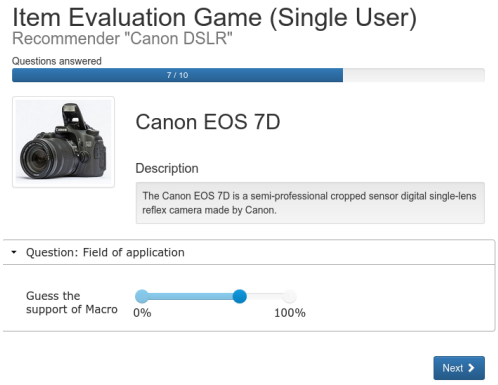

Figure 8: *Item Evaluation Game* in single-user mode.

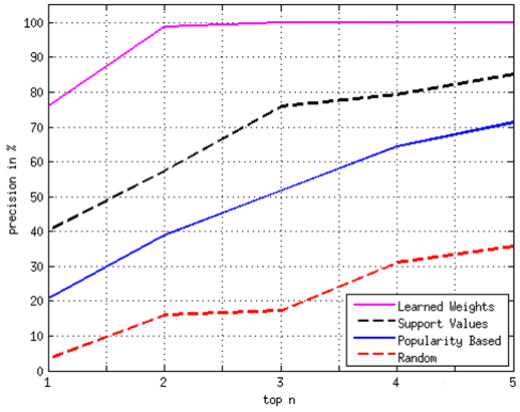

Figure 9: Comparison of baseline versions (random and popularity-based) with PEOPLEVIEWS recommenders (utility-based) with regard to precision.