J Intell Inf Syst (2017) 49:37–57
DOI 10.1007/s10844-016-0433-4

Human computation for constraint-based recommenders

Thomas Ulz[1] · Michael Schwarz[1] · Alexander Felfernig[1] · Sarah Haas[1] · Amal Shehadeh[1] · Stefan Reiterer[1] · Martin Stettinger[1]

Received: 8 December 2015 / Revised: 11 September 2016 / Accepted: 15 September 2016 /
Published online: 28 September 2016

Abstract PEOPLEVIEWS is a Human Computation based environment for the construction of constraint-based recommenders. Constraint-based recommender systems support the handling of complex items where constraints (e.g., between user requirements and item properties) can be taken into account. When applying such systems, users are articulating their requirements and the recommender identifies solutions on the basis of the constraints in a recommendation knowledge base. In this paper, we provide an overview of the PEOPLEVIEWS environment and show how recommendation knowledge can be collected from users of the environment on the basis of micro-tasks. We also show how PEOPLEVIEWS exploits this knowledge for automatically generating recommendation knowledge bases. In this context, we compare the prediction quality of the recommendation approaches integrated in PEOPLEVIEWS using a DSLR camera dataset.

✉ Alexander Felfernig
alexander.felfernig@ist.tugraz.at

Thomas Ulz
thomas.ulz@student.tugraz.at

Michael Schwarz
michael.schwarz@student.tugraz.at

Sarah Haas
sarah.haas@student.tugraz.at

Amal Shehadeh
amal.shehadeh@ist.tugraz.at

Stefan Reiterer
stefan.reiterer@ist.tugraz.at

Martin Stettinger
martin.stettinger@ist.tugraz.at

[1] Applied Software Engineering Group, Institute for Software Technology, TU Graz, Graz, Austria

Keywords Constraint-based systems · Recommender systems · Constraint-based recommendation · Human computation · Knowledge acquisition

1 Introduction

In contrast to collaborative filtering (Goldberg et al. 1992; Konstan et al. 1997), content-based filtering (Pazzani and Billsus 1997; Roy and Mooney 2004), and case-based recommendation (Burke and Hammond 1997; McCarthy et al. 2005; Musto et al. 2014), constraint-based recommender systems (as a specific type of knowledge-based recommender system) (Burke 2000; Felfernig and Burke 2008; Jannach et al. 2010) rely on a predefined set of constraints that perform the selection of candidate items (Burke and Ramezani 2010). Constraints make it possible to explicitly define specific domain properties and restrictions which is not possible in recommendation approaches such as collaborative filtering, content-based filtering, and case-based recommendation (Jannach et al. 2010; Ricci et al. 2011). Especially for complex items, content-based and collaborative filtering approaches can not be directly applied, since these approaches do not allow the inclusion of rules and/or constraints. However, such an inclusion of constraints comes along with a major challenge which is the *knowledge acquisition bottleneck*: domain experts and knowledge engineers have to extensively communicate to correctly encode the recommendation knowledge. The PEOPLEVIEWS approach[1] to generate constraints out of user feedback aims to reduce overheads related to the communication between domain experts and engineers.

Constraint-based recommender systems are applied in more complex item domains where a user specifies a set of criteria (Felfernig and Burke 2008; Felfernig et al. 2009) and the system proposes a corresponding set of recommendations. Items in these domains are purchased less frequently which makes traditional recommendation approaches such as collaborative filtering less applicable (Felfernig and Burke 2008). Constraint-based recommenders are typically conversational (Salem et al. 2014), i.e., the user provides answers to a set of questions and the system tries to retrieve relevant items. Constraints define restrictions regarding the possible combinations of customer requirements (e.g., *a pocket camera does not support exchangeable lenses*) and the relationships between customer requirements and item properties (e.g., *the price of the recommended item must not exceed the maximum price specified by the customer*). Constraints are used to pre-filter candidate items which are then ranked on the basis of a utility scheme provided, for example, by multi-attribute utility theory (Felfernig et al. 2006; Winterfeldt and Edwards 1986).

Constraint-based recommender systems are applied in various domains. Peischl et al. (2010) present an application of constraint-based recommendation technologies for supporting the selection of software testing methods. Felfernig et al. (2006) report applications of these technologies in the domain of financial services and electronic equipment. Leitner et al. (2012) show the application of constraint-based recommendation for recommending smart home solutions to users. WEEVIS[2] (Reiterer 2015) is a constraint-based environment embedded in MediaWiki[3] which allows the definition of recommender applications with a wiki-style syntax and the integration of these recommenders into standard Wiki

[1] The work presented in this article has been developed within the scope of the PEOPLEVIEWS project which is funded by the Austrian Research Promotion Agency (843492).

[2] www.weevis.org.

[3] www.mediawiki.org.

pages. Jannach and Bundgaard-Joergensen (2007) introduce an application that focuses on the recommendation of business plans for company founders. Finally, Torrens et al. (2003) 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 (von Ahn 2005) 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 (Shah and Zhou 2015; von Ahn 2005) 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 (e.g., sales representatives) 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).

Applying Human Computation (von Ahn 2005) in the context of recommender systems development is not new. Hacker and Matchin (2009) introduce the MATCHIN environment where the elicitation of preferences from users is based on the idea of asking questions on which decisions other users would take when being confronted with a set of items. Walsh and Golbeck (2009) 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. Colson (2013) introduces a personalized styling service for apparel items. In this context, a hybrid approach (combination of human computation and machine learning) is applied to determine recommendations. Finally, Larson et al. (2013) introduce the idea of not only recommending items to users but also users to items in situations 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. (2013) and Said et al. (2014), PEOPLEVIEWS does not only support the recommendation of items but also the recommendation of users who should provide item evaluations – this is also denoted as *task routing*. Solutions for task routing in micro-task environments have been presented, for example, in Jung (2014) who introduces a matrix factorization (Koren et al. 2009) based approach to learn reasonable assignments of micro-tasks to users. A corresponding key insight is that task similarity is a core factor in identifying appropriate task routings.

In PEOPLEVIEWS, task similarity is taken into account by determining the content-dependent similarity between already completed tasks of a user (the corresponding information is stored in the user profile) and a new task. The assignment of tasks to users is also addressed in the context of active learning (Elahi et al. 2016) which represents the task of selecting items that should be rated by a user with the goal to improve the overall recommendation accuracy. Further related work on task routing can be found in

Cosley et al. (2007) and Iren and Bilgen (2014). For a detailed discussion of the micro task scheduling approaches included in PEOPLEVIEWS we refer to Felfernig et al. (2015). Nasery et al. (2014) compare true user preferences on features with those predicted by the recommender system. The outcome of their analysis is that there exists a significant mismatch between these two types of preferences. These results justify investments related to the crowd-sourced collection of user item evaluations. A major difference between PEOPLEVIEWS and related work is the focus on the derivation of constraints which enables conversational recommendation and repair processes (Felfernig et al. 2006). Although also operating on user queries, the major difference between PEOPLEVIEWS compared to tagging-based recommendation approaches (Zanardi and Cara 2008) is the focus on similarity metrics when determining recommendations; again, no constraint-based mechanisms such as inconsistency checking and consistency restoration can be applied in tagging-based approaches.

The contributions of this paper are the following. First, we introduce a new approach to the development of knowledge bases for constraint-based recommendation scenarios. This approach helps to assure scalability since it makes it possible to also engage domain experts without a computer science background into knowledge engineering tasks. Second, we introduce new user interfaces that support the acquisition of recommendation knowledge on the basis of micro-tasks. Third, we introduce a new type of constraint representation that allows to automatically derive constraints from user item evaluations. Fourth, we report the results of an empirical study that compares the prediction quality of the included recommendation approach with some baseline versions (random, most popular, and user similarity based recommendation). Finally, we discuss a couple of relevant issues for future work.

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 values 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 attributes specified by a user, is depicted in Table 2.

In addition to user attributes, *product attributes* describe technical properties of an item. This distinction between the user view and the technical item view is frequently applied in knowledge-based recommendation and configuration scenarios – see, for example, Felfernig et al. (2006) and Mittal and Frayman (1989). 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

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

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}

be specified via a search interface), each product attribute is associated with a corresponding *attribute level similarity metric* (McSherry 2003). 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) metric 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 (see Formula (5)).

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 evaluate 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 providing feedback on assigned micro-tasks or by proactively evaluating items.

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

Table 2 Example user requirements $req_i \in REQ$

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

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

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
maxshutterspeed	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
maxresolution	single	Max. resolution?	{8 Mpix, 12 Mpix, 16Mpix, 18Mpix, 21Mpix, 23Mpix}	MIB	yes
price	single	Max. price?	integer	LIB	no

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 attribute values 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 (user-specific evaluation: 1.0). User-specific evaluations 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 (user-specific evaluation: 1.0) for *macro* photography. Furthermore, *Giselle* thinks that the camera should only be recommended to experts (user-specific evaluation: 1.0), the usability of the camera is regarded as *high* (user-specific evaluation: 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-specific 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 recommendation-relevant filter constraint.

The *user-specific evaluations* (see Table 5) for an item Φ, user attribute u, and user attribute value v are aggregated into one *recommendation-relevant evaluation* (see Formulae (1) and (2)). This *eval* value is calculated from the average of all existing user-specific evaluations of attribute value v of attribute u of item Φ *times* the corresponding *support*. The *support* parameter is used to relativize a user-specific evaluation in terms of how often one attribute value for a specific item Φ has been selected by users compared to other values

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

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 5 User-specific evaluations (numbers in brackets) of cameras in PeopleViews. Each row corresponds to a *user-specific filter constraint*

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)

of the same attribute, i.e., support in our context is an indication of *how frequently a user attribute value evaluation for an item appears in the global evaluation set*. For example, the support for the attribute value *sport* for item Φ_3 is $\frac{1}{5}$. Finally, $count(\Phi, u, v)$ denotes the number of evaluations of attribute value v for item Φ and $count(\Phi, u)$ is the number of evaluations of attribute u for item Φ.

$$eval(\Phi, u, v) = \frac{sum_v(\Phi, u, v)}{count(\Phi, u, v)} \times support(\Phi, u, v) \tag{1}$$

$$support(\Phi, u, v) = \frac{count(\Phi, u, v)}{count(\Phi, u)} \tag{2}$$

Table 6 Example of *recommendation-relevant filter constraints* derived from user-specific filter constraints (see Table 5). User-specific evaluations (Table 5) are integrated into recommendation-relevant evaluations (see *eval* in Formula (1)) – if eval = 0, the corresponding attribute value is not part of the filter constraint)

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 (0.29)	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)

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 (with $eval > 0$) of user attribute $u \in U$, furthermore, $noval$ denotes the fact that no value has been selected for the corresponding user attribute.

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

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 evaluations of user attribute values (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.

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

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_3.

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 evaluations for items Φ, product attributes p, and product attribute values v (see Formula (5)). In PeopleViews, such evaluations 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)*. A related overview of similarity metrics can be found, for example in McSherry (2003).

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

Improving the Basic Recommendation Approach using Beta Distributions. The afore discussed recommendation approach calculates the average evaluation values but does not take into account the aspect of *representativity*. For example, if camera A has been evaluated by only one user and received an evaluation of 90 % for the usage in *landscape photography* and camera B already received 100 evaluations but has an average evaluation of 85 % then an interpretation could be that the first camera is better in the dimension landscape photography. In order to take into account such situations, we employ *beta distributions* that can be used to describe the behavior of random variables. For the sample data of each attribute value and the corresponding item, we have to derive a corresponding beta distribution. The *mode* of each such distribution is the most likely value of the distribution and in our case corresponds to the average of the evaluations used to derive the distribution. Applying the beta distribution function to the determined mode value results in the corresponding *peak value*, i.e., the maximum value of the distribution.

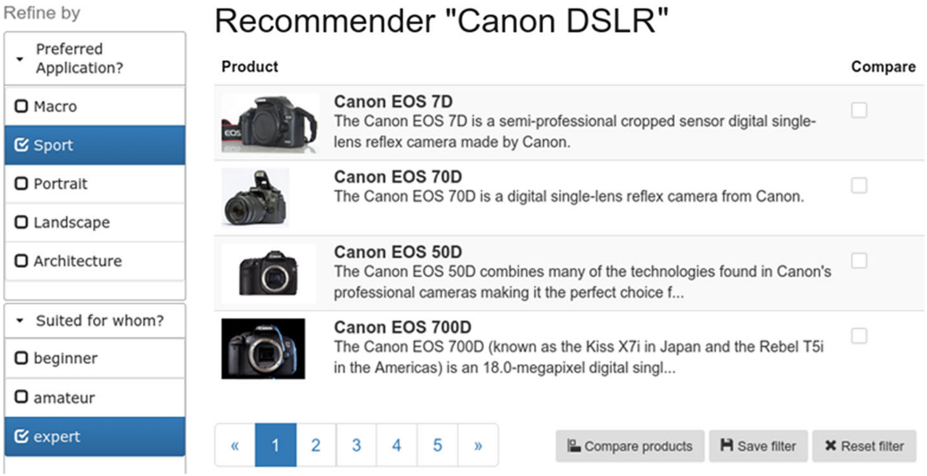
The higher the peak value of a beta distribution related to a set of item x attribute value evaluations, the higher the corresponding *confidence*. In this context, we interpret confidence as *an indication of how often an attribute value is referred to by a user-specific evaluation*. Formula (6) shows the calculation of recommendation-relevant evaluations ($eval_\beta$) when peak values of the underlying beta distributions are taken into account. This approach to determine item attribute evaluations will be compared with the aforementioned standard approach in Section 4. The underlying expectation is that taking into account the aspect of confidence leads to a better prediction quality of the recommendation approach.

$$eval_\beta(\Phi, u, v) = eval(\Phi, u, v) \times confidence(\Phi, u, v) \tag{6}$$

$$confidence(\Phi, u, v) = \frac{peak(\Phi, u, v)}{max(peaks(\Phi, u, v))} \tag{7}$$

3 PeopleViews user interface

In the PeopleViews *recommendation mode*, users can specify their requirements (see Fig. 1) and PeopleViews recommenders determine recommendations on the basis of recommendation-relevant filter constraints and the utility function (see Formula (4)).



Recommender "Canon DSLR"

Preferred Application?
- ☐ Macro
- ☑ Sport
- ☐ Portrait
- ☐ Landscape
- ☐ Architecture

Suited for whom?
- ☐ beginner
- ☐ amateur
- ☑ expert

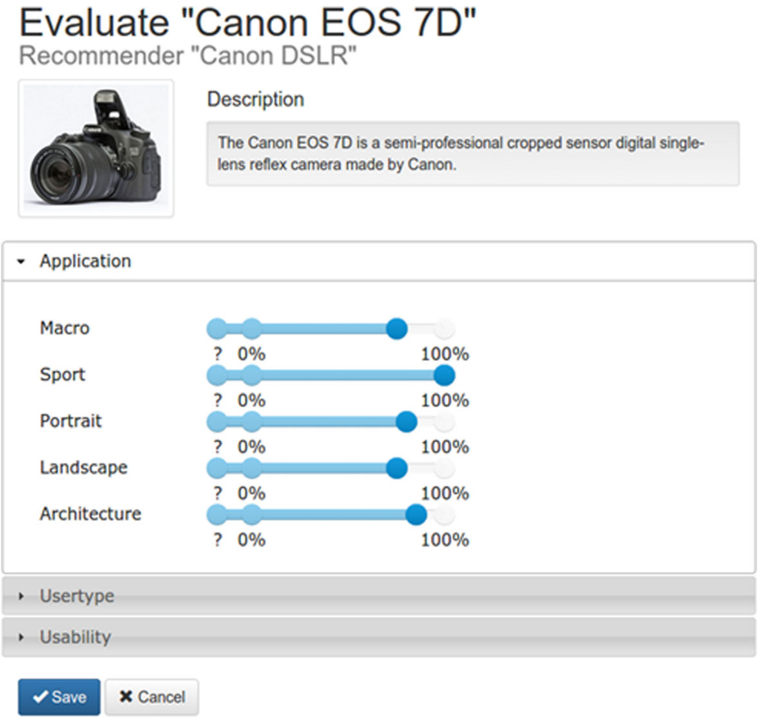
Canon EOS 7D The Canon EOS 7D is a semi-professional cropped sensor digital single-lens reflex camera made by Canon. ☐

Canon EOS 70D The Canon EOS 70D is a digital single-lens reflex camera from Canon. ☐

Canon EOS 50D The Canon EOS 50D combines many of the technologies found in Canon's professional cameras making it the perfect choice f... ☐

Canon EOS 700D The Canon EOS 700D (known as the Kiss X7i in Japan and the Rebel T5i in the Americas) is an 18.0-megapixel digital singl... ☐

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



Evaluate "Canon EOS 7D"

Recommender "Canon DSLR"

Description

The Canon EOS 7D is a semi-professional cropped sensor digital single-lens reflex camera made by Canon.

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

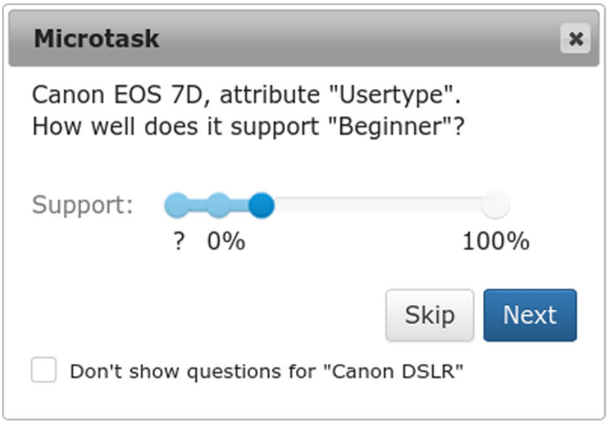
Items in the recommendation list can be selected for product comparison, search criteria entered by the user can be saved, i.e., are taken into account when the user activates the recommender the next time.

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 Fig. 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 acquisition of knowledge relevant for deriving

Table 7 Micro-task types supported in PEOPLEVIEWS

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



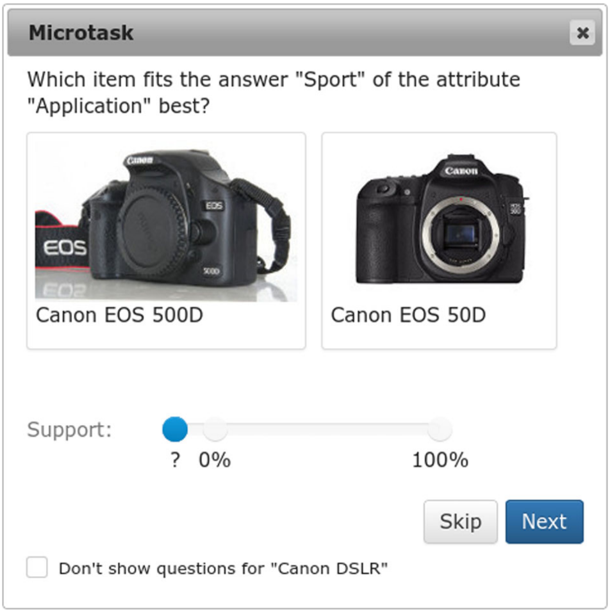
Canon EOS 7D, attribute "Usertype".
How well does it support "Beginner"?

□ Don't show questions for "Canon DSLR"

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

user-specific filter constraints (including the corresponding user-specific evaluations). 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*.

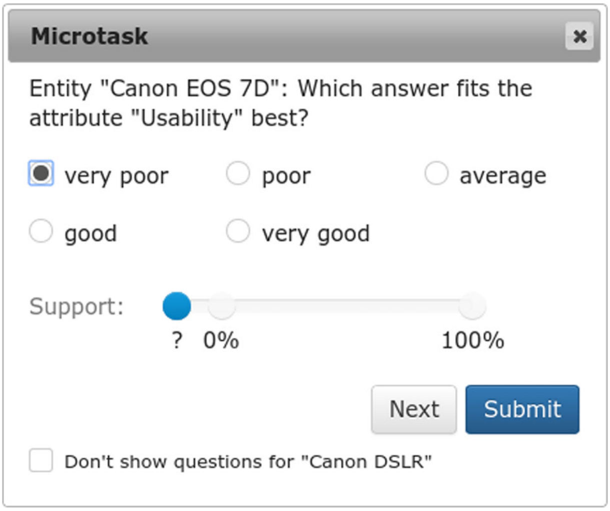
Similar to Fig. 3, the user interface of Fig. 4 allows the evaluation of item-specific properties. In addition, the interface supports the acquisition of performance relationships between items regarding specific attribute values, for example, information regarding the camera



Which item fits the answer "Sport" of the attribute "Application" best?

□ Don't show questions for "Canon DSLR"

Fig. 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*)



Microtask

Entity "Canon EOS 7D": Which answer fits the attribute "Usability" best?

● very poor ○ poor ○ average

○ good ○ very good

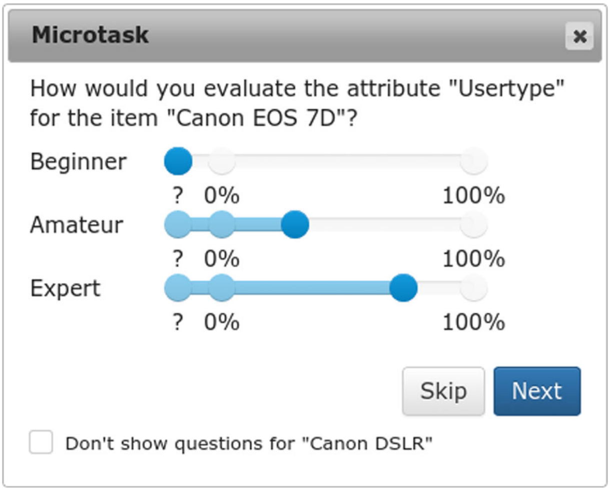
Support: ? 0% 100%

□ Don't show questions for "Canon DSLR"

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

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, Fig. 7 depicts the user interface of a *CAPTCHA*-style micro-task which helps to figure out whether the user is a “real” user or a bot.

Gamification-based acquisition of recommendation knowledge 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



Microtask

How would you evaluate the attribute "Usertype" for the item "Canon EOS 7D"?

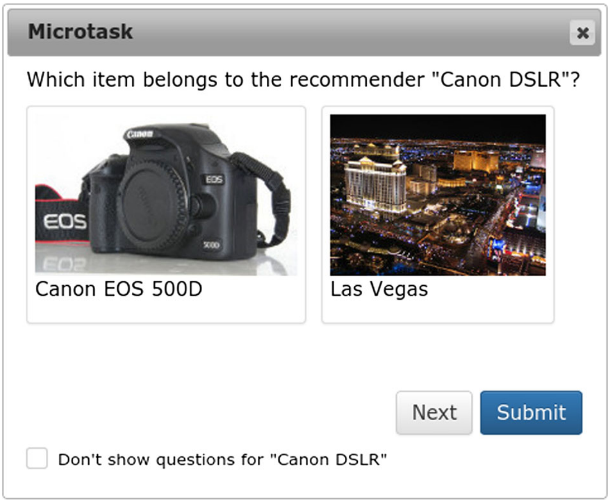
Beginner ? 0% 100%

Amateur ? 0% 100%

Expert ? 0% 100%

□ Don't show questions for "Canon DSLR"

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



Microtask

Which item belongs to the recommender "Canon DSLR"?

Canon EOS 500D

Las Vegas

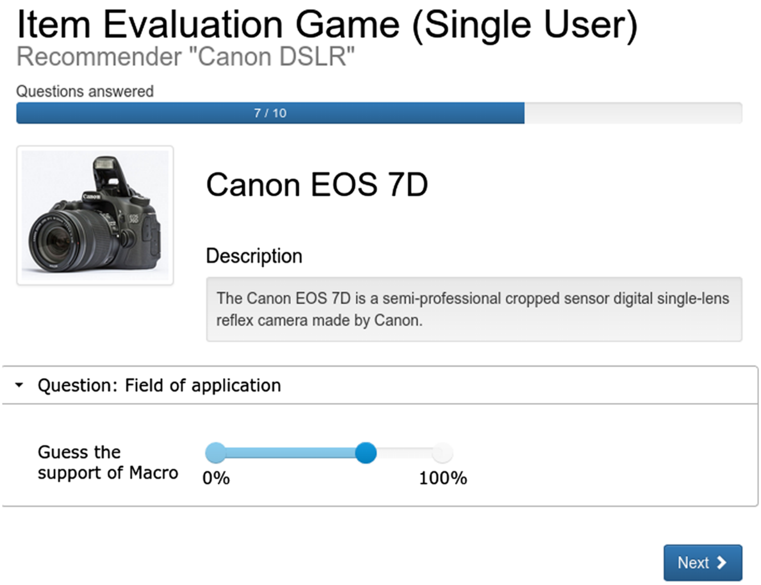
□ Don't show questions for "Canon DSLR"

Fig. 7 CAPTCHA-style micro-task

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 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 Fig. 8. Answers of users (including their item attribute value evaluations) provided within the scope of a gaming session are used in the same way as user answers to micro-tasks.

Inconsistency management When interacting with constraint-based recommenders, users are sometimes confronted with a situation where the recommender is not able to identify a recommendation for a given set of customer requirements. The reason for this is that requirements can induce an inconsistency with the underlying set of constraints. For example, if a user articulates the preferences (requirements) *pocket camera* and at the same time *exchangeable lenses*, the recommendation knowledge base will not be able to identify a corresponding solution. Many user interfaces simply inform the user about the situation and indicate that the requirements have to be adapted. PEOPLEVIEWS follows a more sophisticated approach to deal with such a situation: each recommender has a built-in analysis functionality that automatically determines a minimal set of requirements that have to be adapted by the user such that at least one solution can be identified by the recommendation knowledge base. The algorithmic approach to determine such sets of requirements is denoted as FASTDIAG which is a divide-and-conquer based algorithm for the determination of minimal hitting sets – details of the algorithm are discussed in Felfernig et al. (2012).

PeopleViews mobile The PEOPLEVIEWS environment provides an HTML-5 user interface that includes both, the *modeling mode* (recommenders can be defined on the basis of



Item Evaluation Game (Single User)

Recommender "Canon DSLR"

Questions answered: 7 / 10

Canon EOS 7D

Description

The Canon EOS 7D is a semi-professional cropped sensor digital single-lens reflex camera made by Canon.

Question: Field of application

Guess the support of Macro

Fig. 8 *Item Evaluation Game* in single-user mode

micro-tasks) and the *recommendation mode*. We have also developed an iOS-based mobile version (Promitzer et al. 2016) that supports a subset of the functionalities provided by the HTML-5 version. In the mobile version, users can provide answers to micro-tasks, engage in games, and apply the different recommenders already developed by the PEOPLEVIEWS community. An example of the user interface of the PEOPLEVIEWS mobile version is depicted in Fig. 9.

4 Evaluation

In order to evaluate the different recommendation approaches currently integrated in the system, we used a PEOPLEVIEWS dataset created by users with expertise in the domain of digital cameras. Each of the N=35 users had experiences in the Canon DSLR camera domain and performed the following two basic tasks: (1) specification of a set of user requirements related to user attributes selected by the user and selection of a corresponding digital camera that best fulfills these requirements and (2) evaluation of a user-selected set of digital cameras (cameras selected in (1) were not selectable in this context) with regard to the given set of user attributes. The result of task 1 is a set of 45 test cases (100 % coverage of the Canon DSLR cameras is given) that are the basis for evaluating the predictive quality of the integrated recommendation algorithms. The result of task 2 is a set of 285 item evaluations that are the basis for the derivation of a set of recommendation-relevant filter constraints. The study was announced via the online channels of Graz University of Technology. The set of user attributes that could be instantiated by a user in terms of requirements is a superset of those used in our working example (see Table 1): *preferred application*, *photography experiences*, *usability*, *cost effectiveness*, and *allowed loss of value*.

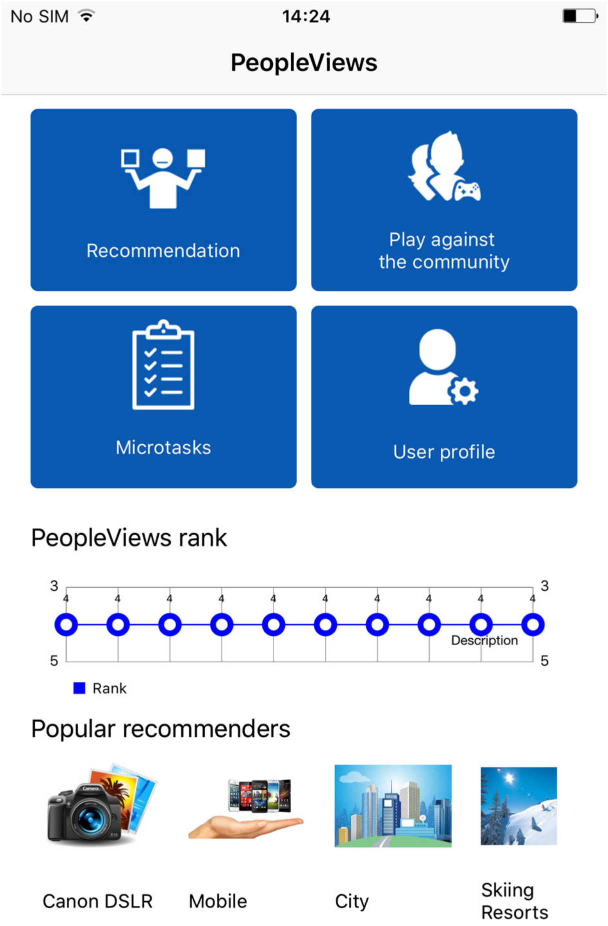


Fig. 9 Start-screen of the PEOPLEVIEWS mobile version

The resulting *test set* (result of task 1) was used for evaluating the predictive quality of the presented recommendation approaches. Using this test set, we compared the baseline methods *random* (the recommended items were selected randomly), *most frequent* (the most frequently selected items were recommended), and *user similarity based* (recommendation of items selected by users with similar preferences) with the ones presented in this paper (*basic* and *extended* where the latter is based on beta distributions as discussed in Section 2). Obviously, all except one of the baseline approaches do not take into account the preferences (requirements) defined by the current user.

In order to evaluate our recommendation approaches, we applied *f-measure* and *recall* (Herlocker et al. 2004) as a metric (see Formulae (8) – (10)) since for digital camera recommendation it is important to find as many relevant items as possible but omit the irrelevant ones (Gunawardana and Shani 2009). In Formulae (8)–(10), $\#hits(N)$ denotes the number of times an item that was selected by a user in the test set was also included in the top-N recommended items. Furthermore, $|D|$ is the size of the test set. The number of hits divided by the number of possible hits ($|D|$) represents the *recall*. *Precision* is then defined as the ratio between the number of *hits* and the number of recommended items which is $N \times |D|$.

$$f\text{-}measure(N) = 2 \times \frac{precision(N) \times recall(N)}{precision(N) + recall(N)} \tag{8}$$

$$precision(N) = recall(N) \times \frac{1}{N} \tag{9}$$

$$recall(N) = \frac{\#hits(N)}{|D|} \tag{10}$$

Figures 10 and 11 depict a comparison of the predictive quality of the following approaches: *random*, *most frequent*, *user similarity based*, *basic*, and *extended* (beta distribution based). Due to the low number of items (#13 cameras were used in our study), all algorithms achieve nearly 100 % recall for $N = 10$. In addition, we also analyze to which extent recommendation performance can be further improved by supporting the learning of the importance weights of user preferences. A comparison of the recommendation approaches with learned weights (see Formula (4)) is depicted in Fig. 12. In this context, weights have been learned on the basis of a genetic algorithm with the goal to improve recall (details on the implementation of this learning approach are provided in Ulz (2016)).

In order to achieve high data quality, it is important to assure that domain experts perform micro-tasks. This can be done by letting users themselves select the items they would like to evaluate or to identify relevant users by analyzing user interaction logs (see also Section 5). When taking into account learned weights (see Fig. 12), a nearly 100 % recall rate can be achieved for the top-3 items which is acceptable for a digital camera recommender (at least one among three recommended items has a very high probability of being accepted by the user). Since only N=35 users participated in our study, further improvements with regard to prediction quality can be expected. The number of potential users who can contribute to micro-tasks is in general limited by the item domain, for example, a recommender for iOS mobile phones has more potential contributors compared to mobile phones with a lower market share.

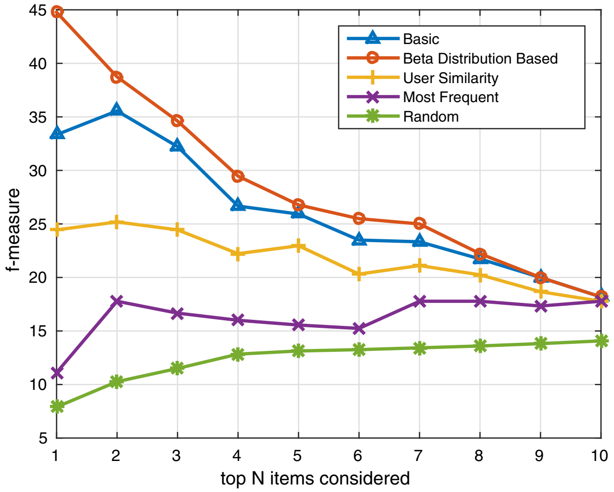


Fig. 10 Comparison of baseline versions (random, most frequent, and user similarity) with PEOPLEVIEWS recommenders (basic and extended) with regard to *f-measure* (see Formula (8))

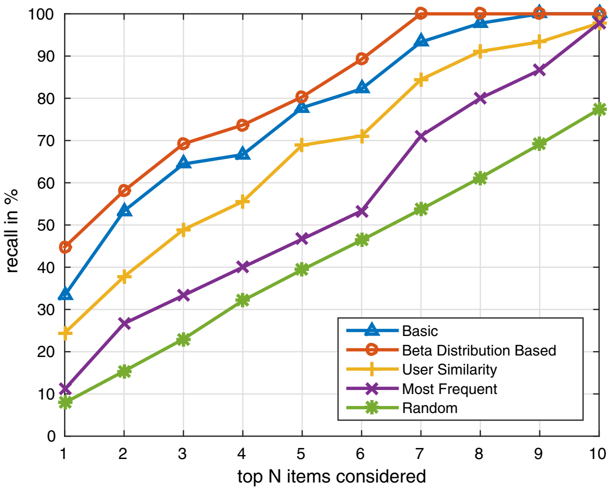


Fig. 11 Comparison of baseline versions (random, most frequent, and user similarity) with PEOPLEVIEWS recommenders (basic and extended) with regard to *recall* (see Formula (10))

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 further approaches to case-based reasoning (CBR) that include, for example, a probability-based approach to determine similarities (Deshpande and Karypis 2004). 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

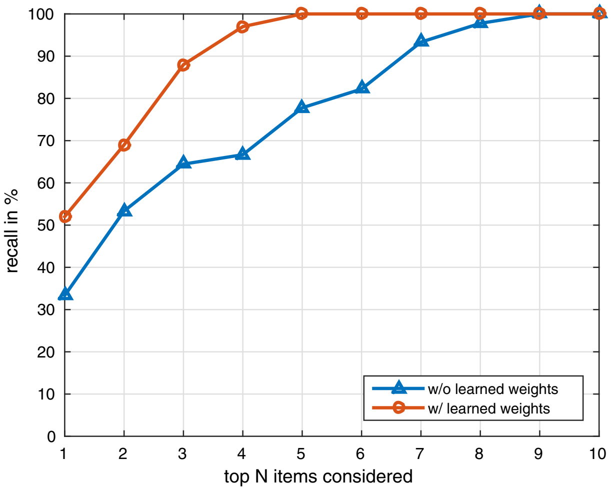


Fig. 12 Comparison of PEOPLEVIEWS recommenders (basic with and without learned weights) with regard to *recall* (see Formula (10))

domains, where there is not a unique, objective truth about items, for example, in the context of collaborative filtering recommendation scenarios (see, e.g., (Larson et al. 2013; Organisciak et al. 2013)).

Usability studies The current version of the PEOPLEVIEWS user interface is the result of a couple of iterations that focused on improving the understandability of the user interface. The different presented types of micro-tasks are the result of this process. However, further studies are needed to continuously improve the usability of the interface, for example, we plan to improve the overall quality of explanations. In this context we also want to identify ways to include the results of micro-tasks into explanation generation. For example, if no solution can be identified for a set of user requirements, the determination of corresponding repair actions can be guided on the basis of user feedback from micro-tasks. In order to achieve this goal, additional micro-task templates have to be developed that extend the existing set.

Micro-task scheduling Recommendation-relevant filter constraints are derived from data collected by micro-tasks. We have already integrated a basic version of a scheduling approach that assigns micro-tasks to users. This is in the line of *user-item reciprocity* (Larson et al. 2013; Said et al. 2014): (1) users should be interested in items recommended to them and also have the potential to enrich them and (2) mechanisms are needed for enticing users to provide data. The first aspect can be considered as a more traditional recommendation related task whereas the second one includes motivational factors to engage users. In PEOPLEVIEWS, the second aspect is currently covered by a *scoring approach* were points can be received, for example, for completing micro-tasks, adding new items to a recommender, or evaluating items. For industrial scenarios (e.g., in the financial services domain) we assume the inclusion of different types of motivation mechanisms ranging from monetary approaches to recognitions. These can be based on scoring information from the PEOPLEVIEWS environment. This way, PEOPLEVIEWS remains open which is important since mechanisms can differ significantly.

The micro-task scheduling approach currently integrated in PEOPLEVIEWS is based on the dimensions of *importance*, *ability*, and *interest*. We interpret *importance* of a micro-task as a measure of time, i.e., the more time passed between the creation and distribution of the micro-task and the current time, the more important the task is.

Furthermore, the *ability* of a user to complete a micro-task is measured in terms of the similarity between the description of the micro-task and the information about micro-tasks that were already successfully completed by the user in the past (information contained in the user profile). Finally, a user's *interest* in completing a micro-task is measured in terms of the amount of micro-tasks that have already been completed in the early past. We denote the union of ability and interest as *qualification* of a user to complete a certain micro-task. In PEOPLEVIEWS, the creation of a micro-task starts with an initial agenda, for example, if a new item has been inserted by a user, a set of micro-tasks is created to evaluate the attributes of the new item. Depending on the quality of the evaluation data, further micro-tasks need to be created or not (e.g., if there is a uniform-like distribution in the resulting data, further micro-tasks are triggered). We want to emphasize that micro-task generation and user assignment will be further investigated in our future work where we will also take into account existing work in the context of active learning (Elahi et al. 2016).

Quality assurance *CAPTCHA-style micro-tasks* (see, e.g., Fig. 7) are used to identify micro-task contributions that should not be forwarded to the recommendation process. The higher the share of faulty answers of a certain user, the higher the probability that the user is, for example, not a real user but a bot trying to insert item evaluations into the PEOPLEVIEWS environment. Furthermore, *timing models* indicate to which extent the time efforts of the current user correspond to those of other users. The higher the deviation of the time efforts of the current user, the lower the impact of the corresponding contribution. A basic means to assure the quality of micro-contributions are constraints representing *ground truth*. In the context of collecting micro-task feedback for a digital camera recommender, some questions can be formulated that help to validate the qualification of a user with regard to a certain topic. If, for example, the question regarding the sensor type included in a Canon EOS 5D Mark III has been answered incorrectly, further contributions of this user are considered of lower value compared to users who answered such questions correctly. One of our goals for future work is to compare the performance of different user communities with regard to the completion of micro-tasks. For the domain of digital cameras we want to compare the recommendation quality of knowledge bases constructed by *micro-worker communities* such as *microworkers.com* with *expert communities* in the domain of digital cameras. The comparison of such types of communities and their properties is our focus for future work.

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 can be used for collecting recommendation knowledge. We introduce the currently integrated recommendation approaches and analyze their prediction quality. Furthermore, we sketch the currently developed approaches to micro-task scheduling and quality assurance of micro-task data. Major issues for future work are the inclusion of additional recommendation algorithms that help to improve prediction quality, detailed empirical studies related to the quality of micro-task scheduling, i.e., selecting and assigning micro-tasks to users, and empirical studies related to the quality of datasets acquired from different user communities.

Acknowledgments Open access funding provided by Graz University of Technology, Graz, Austria.

References

Burke, R. (2000). Knowledge-based Recommender Systems. *Encyclopedia of Library and Information Systems*, *69*(32), 180–200.

Burke, R., & Hammond, K. (1997). The FindMe Approach to Assisted Browsing. IEEE Expert, 32–40.

Burke, R., & Ramezani, M. (2010). Matching Recommendation Technologies and Domains. In *Recommender Systems Handbook* (pp. 367–386): Springer.

Colson, E. (2013). Using human and machine processing in recommendation systems. In *human computation and crowdsourcing: Works in progress and demonstration abstracts, Technical Report CR-13-01* (pp. 16–17).

Cosley, D., Frankowski, D., Terveen, L., & Suggestbot, J.Riedl. (2007). Using intelligent task routing to help people find work in wikipedia. In *IUI'07* (pp. 32–41).

Deshpande, M., & Karypis, G. (2004). Item-Based Top-N Recommendation algorithms. *ACM Transactions on Information Systems*, *22*(1), 143–177.

Elahi, M., Ricci, F., & Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, *20*, 29–50.

Felfernig, A., & Burke, R. (2008). Constraint-based Recommender Systems: Technologies and Research Issues. In *IEEE ICEC'08, 17–26, Innsbruck, Austria*.

Felfernig, A., Friedrich, G., Jannach, D., & Zanker, M. (2006). An Environment for the Development of Knowledge-based Recommender Applications. *International Journal of Electronic Commerce (IJEC)*, *11*(2), 11–34.

Felfernig, A., Haas, S., Schwarz, M., Ulz, T., Stettinger, M., & Reiterer, S. (2015). Micro-Task Scheduling for Constraint-based Recommendation. In *Technical Report, Graz University of Technology, pages 1–6, Graz, Austria*.

Felfernig, A., Schubert, M., Friedrich, G., Mandl, M., Mairitsch, M., & Teppan, E. (2009). Plausible Repairs for Inconsistent Requirements. In *21st International Joint Conference on Artificial Intelligence (IJCAI'09), pages 791–796, Pasadena, CA*.

Felfernig, A., Schubert, M., & Zehentner, C. (2012). An efficient diagnosis algorithm for inconsistent constraint sets. *AIEDAM*, *25*(2), 175–184.

Goldberg, D., Nichols, D., Oki, B., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 51–60.

Gunawardana, A., & Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, *10*, 2935–2962.

Hacker, S., & Matchin, L.v.on.Ahn. (2009). Eliciting user preferences with an online game. In *CHI'09* (pp. 1207–1216).

Iren, D., & Bilgen, S. (2014). Cost of quality in crowdsourcing. *Human Computation*, *1*(2), 283–314.

Herlocker, L.T.erveen.J., Konstan, J., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, *22*(1), 5–53.

Jannach, D., & Bundgaard-Joergensen, U. (2007). SAT: A Web-Based interactive advisor for Investor-Ready business plans. In *Intl. Conference on e-Business (ICE-B 2007)* (pp. 99–106).

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems – An Introduction*. Cambridge: Cambridge University Press.

Jung, J. (2014). Quality Assurance in Crowdsourcing via Matrix Factorization based Task Routing. In *WWW'14, pages 3–7, Seoul, Korea*.

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., & Riedl, J. (1997). GroupLens: Applying collaborative filtering to usenet news. *Comm. of the ACM*, *40*(3), 77–87.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, *42*(8), 30–37.

Larson, M., Said, A., Shi, Y., Cremonesi, P., Tikk, D., & Karatzoglou, A. (2013). Activating the Crowd: Exploiting User-Item Reciprocity for Recommendation. In *CrowdRec'13, pages 1–2, Hong Kong, China*.

Leitner, G., Fercher, A., Felfernig, A., & Hitz, M. (2012). Reducing the Entry Threshold of AAL Systems Preliminary Results from Casa Vecchia. In *13th Intl. Conference on Computers Helping People with Special Needs* (pp. 709–715).

McCarthy, K., Reilly, J., Smyth, B., & McGinty, L. (2005). Generating diverse compound critiques. *AI Review*, *24*(3–4), 339–357.

McSherry, D. (2003). Similarity and compromise. In *5th International Conference on Case-Based Reasoning* (pp. 291–305).

Mittal, S., & Frayman, F. (1989). Towards a Generic Model of Configuration Tasks. In *11th International Joint Conference on Artificial Intelligence (IJCAI'89), pages 1395–1401, Detroit, MI, USA*.

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

Nasery, M., Elahi, M., & Cremonesi, P. (2014). Polimovie: a Feature-based Dataset for Recommender Systems. In *CrowdRec'14* (pp. 1–6).

Organisciak, P., Teevan, J., Dumais, S., Miller, R., & Kalai, A. (2013). Personalized Human Computation. In *HCOMP'13, pages 56–57, Palm Springs, CA, USA*.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, *27*, 313–331.

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

Promitzer, A., Felfernig, A., Schwarz, M., Ulz, T., Shehadeh, A., & Haas, S. (2016). A Short Overview of the PeopleViews Mobile User Interface. In *TU Graz Technical Report* (pp. 1–4).

Roy, L., & Mooney, R. (2004). Content-based Book Recommending Using Learning for Text Categorization. *User Modeling and User-Adapted Interaction*, *14*(1), 37–85.

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

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (2011). Recommender systems handbook springer.

Said, A., Larson, M., Tikk, D., Cremonesi, P., Karatzoglou, A., Hopfgartner, F., Turrin, R., & Geurt, J. (2014). User-Item Reciprocity in Recommender Systems: Incentivizing the Crowd. In *UMAP Project Synergy Workshop (UMAP ProS'14), pages 23–26, Aalborg, Denmark*.

Salem, Y., Hong, J., & Liu, W. (2014). History-guided Conversational Recommendation. In *23rd International Conference on World Wide Web (WWW'14), pages 999–1004. ACM*.

Shah, N., & Zhou, D. (2015). On the impossibility of convex inference in human computation. In *AAAI'15* (pp. 1291–1297).

Torrens, M., Hertzog, P., Samson, L., & Faltings, B. (2003). Reality: A Scalable Intelligent Travel Planner. In *ACM Symp. on Applied Computing, pages 623–630, Melbouurne, FL, USA*.

Ulz, T. (2016). Human Computation based Recommendation Technologies. In *Master Thesis, Graz University of Technology, pages 1–103, Graz, Austria*.

von Ahn, L. (2005). Human Computation. In *Technical Report CM-CS-05-193*.

Walsh, G., & Golbeck, J. (2009). Curator: A Game with a Purpose for Collection Recommendation. In *CHI 2009, pages 2079–2082, Boston, Massachusetts, USA*.

Winterfeldt, D., & Edwards, W. (1986). Decision analysis and behavioral research, Cambridge University Press, Cambridge.

Zanardi, V., & Cara, L. (2008). Social Ranking: Uncovering Relevant Content Using Tag-based Recommender Systems. In *Recommender Systems*, (Vol. 2008 pp. 51–58).