

Human Computation Based Acquisition Of Financial Service Advisory Practices

Alexander Felfernig¹ and Michael Jeran¹ and Martin Stettinger¹ and
Thomas Absenger¹ and Thomas Gruber¹ and Sarah Haas¹ and Emanuel Kirchengast¹ and
Michael Schwarz¹ and Lukas Skofitsch¹ and Thomas Ulz¹

Abstract. Knowledge-based recommenders support an easier comprehension of complex item assortments (e.g., financial services and electronic equipment). In this paper we show (1) how such recommenders can be developed in a Human Computation based knowledge acquisition environment (PEOPLEVIEWS) and (2) how the resulting recommendation knowledge can be exploited in a competition-based e-Learning environment (STUDYBATTLE).

1 Introduction

Knowledge-based recommenders [2] support users on the basis of semantic knowledge about the item (product) domain.² One variant of knowledge-based recommenders are *constraint-based recommenders* [8] which exploit explicit constraints (rules) that encode the recommendation knowledge. Another variant are *critiquing-based recommenders* [4]: new items are presented to the user as long as the user is unsatisfied and articulates critiques (e.g., an item should be *cheaper*). In critiquing-based recommendation, new items are determined by similarity functions. For a detailed overview of recommendation approaches we refer to [3, 20].

In this paper we focus on constraint-based recommenders, i.e., recommenders that are based on explicit recommendation rules (constraints). The development of such recommenders is often a time-consuming and error-prone process which can be primarily explained by the *knowledge acquisition bottleneck*: in the formalization of product domain and recommendation knowledge, misunderstandings can occur and as a result knowledge engineers encode this knowledge in an unintended fashion. The more recommenders have to be developed and maintained the higher the risk that the organization runs into a scalability problem where additional resources are needed to be able to perform knowledge engineering and maintenance.

An alternative to the hiring of additional staff for development and maintenance of recommendation knowledge bases is to change the underlying knowledge engineering paradigm. The idea of PEOPLEVIEWS is to engage domain experts more deeply into knowledge engineering tasks. We do not want to "convert" them into technical experts but to define basic tasks (*micro tasks*) that are easy to understand and complete even for domain experts without the corresponding technical expertise. Micro tasks completed by users pro-

vide knowledge chunks that can be aggregated into a PEOPLEVIEWS recommender knowledge base.

The resulting PEOPLEVIEWS recommenders support customers (and especially in the financial services domain also sales representatives) in finding products that fit their wishes and needs. Using such a recommender, items are retrieved within the scope of a dialog (these systems are often also denoted as conversational) where users articulate their requirements and the system tries to identify corresponding solutions. Major advantages of such systems are reduced error rates in the phase of order acquisition, more time that can be invested in contacting new customers due to fewer errors, more satisfied customers, and also pre-informed customers due to the fact that recommender applications can be made publicly available.

Knowledge-based recommender systems have been applied in various item domains – due to the diversity of applications, we can only give some examples of applications of these systems. In the financial services domain, for example, the following applications of knowledge-based recommendation technologies are reported in the literature. Felfernig et al. [11, 12] show an application in the context of investment decisions where recommenders are provided to sales representatives who exploit the recommenders in sales dialogs. Time savings are reported as one of the major improvements directly related to the application of recommendation technologies. Another application of knowledge-based technologies in financial services is presented by Fano and Kurth [7] who introduce a simulation environment that can directly visualize the effects of financial decisions on the financial situation of a family.

Felfernig et al. [9] present a digital camera recommender deployed on a large Austrian product comparison platform. Peischl et al. [22] show the application of constraint-based recommendation technologies in the domain of software effort estimation. WEEVIS[25]³ is a MediaWiki⁴ based environment for the development and maintenance of constraint-based recommender applications – a couple of freely available recommenders have already been deployed. Knowledge-based technologies for the recommendation of business plans are introduced by Jannach and Bundgaard-Joergensen [19]. The recommendation of equipment configuration in the context of smarthomes is introduced by Leitner et al. [21]. Technologies that recommend changes in software development practices are introduced by Pribik and Felfernig [23]. Finally, Burke and Ramezani [5] show how to select recommendation algorithms by introducing rules for *recommending recommenders*.

¹ Applied Software Engineering, Institute for Software Technology, Graz University of Technology, Austria, email: {felfernig, mjeran, stettinger}@ist.tugraz.at, {thomas.absenger, th.gruber, sarah.haas, emanuel.kirchengast, michael.schwarz, lukas.skofitsch, thomas.ulz}@student.tugraz.at.

² The terms *item* and *product* are used synonymously throughout the paper.

³ www.weevis.org.

⁴ www.mediawiki.org.

In PEOPLEVIEWS, principles of Human Computation [26] are included into the development of knowledge-based recommenders. The idea of Human Computation is to let persons perform tasks in which they are better than computers, for example, the identification of product properties from a website. In the context of knowledge base development and maintenance the idea is to let domain experts perform tasks they are much better in compared to knowledge engineers who typically have less knowledge about the product domain and thus relieve the work of knowledge engineers. MATCHIN [18] is based on the idea of preference elicitation by asking users what a person would typically prefer when having to choose between alternatives. Compared to this work, PEOPLEVIEWS allows to derive constraint-based recommenders which are the basis for intelligent user interfaces that support, for example, deep explanations [17] and the diagnosis and repair of inconsistent requirements [13, 14].

The major contributions of this paper are the following. First, we show how financial service recommender knowledge bases can be developed by a community of domain experts. Second, we sketch how such knowledge bases can also be exploited for teaching advisory practices on the basis of games (STUDYBATTLE environment). Third, we provide a discussion of major issues for future research.

The remainder of this paper is organized as follows. In Section 2 we introduce basic concepts of Human Computation based knowledge construction. To give an impression of the PEOPLEVIEWS and the STUDYBATTLE user interface, we present example screenshots in Section 3. Preliminary results of empirical evaluations are shortly discussed in Section 4. In Section 5 we provide an overview of issues for future work. We conclude the paper with Section 6.

2 Developing PEOPLEVIEWS Recommenders

The PEOPLEVIEWS environment supports two basic modes of interaction. First, recommender applications can be created in the *modeling mode* and second, the applications can be executed in the *recommendation mode*. In this section we discuss different tasks to be performed in order to create a PEOPLEVIEWS recommender. Table 1 provides an overview of the users of our working example. These users will jointly develop a PEOPLEVIEWS recommender.

user	email	pwd
Andrea	andrea@...	****
Mary	mary@...	*****
Luc	luc@...	*****
Torsten	torsten@...	****

Table 1. Example users of PEOPLEVIEWS environment.

Table 2 contains an overview of items (financial services) that are used in our working example. The *Investment Funds (A and B)* have a higher risk of loss and require that customers have a high willingness to take risks, otherwise these services will not be recommended. *Building Loan, Bond, and Savings Book* are lower-risk items. In the current version of PEOPLEVIEWS, items can be characterized by additional item attributes, however, these attributes are not used by recommendation rules constructed from micro contributions.

In PEOPLEVIEWS, user requirements $req_i \in REQ$ are specified as assignments of *user attributes*. For our financial services recommender we define a set of user attributes which are enumerated in Table 3. In the current version of the system, user attributes are defined by the creators of a recommender application, i.e., attribute definitions can not be extended by other users who contribute to the further

id	item name
Φ_1	Investment Fund A
Φ_2	Investment Fund B
Φ_3	Building Loan
Φ_4	Bond
Φ_5	Savings Book

Table 2. Example set of items used in working example.

development of the application on the basis of micro tasks.

user attribute	question to user	attribute domain
goal (gl)	What are your personal goals?	{Studies, Pension, Speculation, Car, House, World trip, noval}
runtime (rt)	When is the money needed?	{in 1 year, in 2 years, in 3-5 years, in 5-10 years, in 10-20 years, in more than 20 years, noval}
risk (ri)	Preparedness to take risks?	{low, medium, high, noval}

Table 3. User attributes $u \in U$ of example financial services recommender.

In the PEOPLEVIEWS *recommendation mode*, user attributes can be used to specify user (customer) requirements $req_i \in REQ$. In the *modeling mode*, user attributes represent a central element of a micro task: given a certain item, users are asked to estimate which values of user attributes are compatible with the item, i.e., are a criteria for selecting and recommending the item. The evaluation of items with regard to user attributes is the central micro task implemented in the current PEOPLEVIEWS prototype. A detailed evaluation of the example items (Table 2) regarding the user attributes *goal*, *runtime*, and *risk* is provided in Table 4.

Each row of Table 4 specifies a so-called *user-specific filter constraint* [10], i.e., a filter constraint (specified by a user) regarding a specific item. For example, user *Luc* specified *Pension* and *Speculation* as possible goals that lead to an inclusion of the item *Investment Fund B* into a recommendation. Furthermore, *Luc* believes that a user should have a high preparedness to take risks (attribute *risk*) and should need the payment in 3-5 years, 5-10 years or 10-20 years from now on. Semantically, an item X is selected by a user-specific filter constraint if all the preconditions are fulfilled.

In order to derive *recommendation-relevant filter constraints* (recommendation rules) [10]), user-specific filter constraints have to be aggregated. An example of this aggregation step is depicted in Table 5. For each item all related user-specific filter constraints are integrated into one constraint. Each row in this table has to be interpreted as a filter constraint for a specific item, for example, the constraint in the first row of Table 5 is the following. The item Φ_1 (*Investment Fund A*) is *included* (recommended) if the user requirements regarding goal (*gl*), runtime (*rt*), and risk (*ri*) are consistent with the condition of the recommendation-relevant filter constraint $gl \in \{Studies, Pension, Speculation, noval\} \wedge rt \in \{in\ 5-10\ year, in\ 10-20\ years, noval\} \wedge ri \in \{medium, high, noval\} \rightarrow include(\Phi_1)$.

Table 5 includes the complete set of recommendation-relevant filter constraints (recommendation rules). Exactly these conditions are applied by PEOPLEVIEWS to determine recommendations for a user. In PEOPLEVIEWS, each item has exactly one related recommendation-relevant filter constraint; each such filter constraint is represented by one row in Table 5. The general logical representation of a recommendation-relevant filter constraint f for an item Φ is shown in Formula 1. In this context, $values(\Phi, u)$ is the set of

user	item name (id)	goal	runtime	risk
Andrea	Investment Fund A (Φ_1)	Studies, Pension, Speculation	in 5-10 years, in 10-20 years	high
Luc	Investment Fund A (Φ_1)	Pension, Speculation	in 5-10 years, in 10-20 years	high
Mary	Investment Fund A (Φ_1)	Pension, Speculation	in 5-10 years, in 10-20 years	medium, high
Torsten	Investment Fund B (Φ_2)	Pension, Speculation	in 3-5 years, in 5-10 years, in 10-20 years	high
Luc	Investment Fund B (Φ_2)	Pension, Speculation	in 3-5 years, in 5-10 years, in 10-20 years	high
Mary	Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years, in 10-20 years	low, medium, high
Andrea	Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years	low, medium
Luc	Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years	low, medium
Mary	Bond (Φ_4)	Studies, Car, House	in 2 years, in 3-5 years, in 5-10 years	low, medium
Andrea	Savings Book (Φ_5)	Studies, Car, House, World trip	in 1 year, in 2 years, in 3-5 years, in 5-10 years	low
Torsten	Savings Book (Φ_5)	Studies, House, World trip	in 1 year, in 2 years, in 3-5 years, in 5-10 years	low

Table 4. Example of user-specific filter constraints (= micro contributions).

supported domain values of user attribute $u \in U$ (see Table 4). The constant *noval* denotes the fact that no value has been selected for the corresponding user attribute.

$$f(\Phi) : \bigwedge_{u \in U} u \in \text{values}(\Phi, u) \cup \{\text{noval}\} \rightarrow \text{include}(\Phi) \quad (1)$$

For each pair $(\Phi, \text{val} \in \text{values}(\Phi, u))$, PEOPLEVIEWS determines a corresponding support value (see Formula 2). In this context, $\text{occurrence}(\Phi, \text{val})$ denotes the number of times, value *val* occurs in a user-specific filter constraint for item Φ and $\text{occurrence}(\Phi)$ denotes the number of times an item Φ is referred in a user-specific filter constraint. For example, $\text{support}(\Phi_1, \text{Studies}) = \frac{1}{3}$.

$$\text{support}(\Phi, \text{val}) = \frac{\text{occurrence}(\Phi, \text{val})}{\text{occurrence}(\Phi)} \quad (2)$$

The complete set of support values is depicted in Table 6. In PEOPLEVIEWS, an item Φ can have an associated *rating* ($\text{rating}(\Phi)$) which represents an item evaluation with regard to quality and related services. Such a rating can be determined, for example, by calculating the average of the individual user item ratings.⁵ For simplicity, we do not take into account user ratings in the utility function discussed below (see Formula 3).

Depending on the requirements articulated by the current user (see, e.g., Table 7), PEOPLEVIEWS determines and ranks a set of relevant items as follows. First, recommendation-relevant filter constraints are applied to pre-select items that fulfill the user requirements $REQ = \{\text{req}_1, \text{req}_2, \dots, \text{req}_k\}$. In our example, the set $\{\text{Investment Fund A}, \text{Building Loan}\}$ would be selected by the recommendation-relevant filter constraints (see Table 5).

item name (id)	attribute:value	support value
Investment Fund A (Φ_1)	goal: Studies	0.33
	goal: Pension, Speculation	1.0
	runtime: in 5-10 years, in 10-20 years	1.0
	risk: medium	0.33
	risk: high	1.0
Investment Fund B (Φ_2)	goal: Pension, Speculation	1.0
	runtime: in 3-5 years, in 5-10 years, in 10-20 years	1.0
	risk:high	1.0
Building Loan (Φ_3)	goal: Studies, Pension, Car, House	1.0
	runtime:in 5-10 years	1.0
	runtime:in 10-20 years	0.33
	risk:low, medium	1.0
	risk:high	0.33
Bond (Φ_4)	goal: Studies, Car, House	1.0
	runtime:in 2 years, in 3-5 years, in 5-10 years	1.0
	risk:low, medium	1.0
Savings Book (Φ_5)	goal: Studies, House, World trip	1.0
	goal:Car	0.5
	runtime:in 1 year, in 2 years, in 3-5 years, in 5-10 years	1.0
	risk:low	1.0

Table 6. Support values (see Formula 2) derived from user-specific filter constraints (see Table 4).

⁵ Similar to ratings provided by platforms such as *amazon.com*.

item name (id)	goal	runtime	risk
Investment Fund A (Φ_1)	Studies, Pension, Speculation	in 5-10 years, in 10-20 years	medium, high
Investment Fund B (Φ_2)	Pension, Speculation	in 3-5 years, in 5-10 years, in 10-20 years	high
Building Loan (Φ_3)	Studies, Pension, Car, House	in 5-10 years, in 10-20 years	low, medium, high
Bond (Φ_4)	Studies, Car, House	in 2 years, in 2-5 years, in 5-10 years	low, medium
Savings Book (Φ_5)	Studies, Car, House, World trip	in 1 year, in 2 years, in 3-5 years, in 5-10 years	low

Table 5. Example of *recommendation-relevant filter constraints* which are the result of integrating user-specific filter constraints (see Table 4).

id	requirement
req_1	$goal = \text{Studies}$
req_2	$goal = \text{Pension}$
req_3	$runtime = \text{in 5-10 years}$
req_4	$risk = \text{medium}$

Table 7. Example set of user requirements ($req_i \in REQ$).

The determined recommendation set must be ranked before being presented to the user. In PEOPLEVIEWS, item ranking is based on the following utility function (see Formula 3). The utility of each item is derived from the support values of individual requirements (see Formula 2).

$$utility(\Phi, REQ) = \sum_{req \in REQ} support(\Phi, req) \quad (3)$$

The item ranking of our working example as a result of applying Formula 3 is depicted in Table 8. For example, $utility(\Phi_3, REQ = \{goal = \text{Studies}, goal = \text{Pension}, runtime = \text{in 5-10 years}, risk = \text{medium}\}) = support(\Phi_3, goal = \text{Studies}) + support(\Phi_3, goal = \text{Pension}) + support(\Phi_3, runtime = \text{in 5-10 years}) + support(\Phi_3, risk = \text{medium}) = 1.0 + 1.0 + 1.0 + 1.0 = 4.0$.

item name (id)	utility	rank
Building Loan (Φ_3)	4.0	1
Investment Fund A (Φ_1)	2.66	2

Table 8. Utility-based ranking of items in the recommendation set.

3 User Interface

3.1 PEOPLEVIEWS

In this section we discuss the PEOPLEVIEWS user interface⁶ and also show how PEOPLEVIEWS recommendation knowledge can be exploited by the STUDYBATTLE learning environment. The PEOPLEVIEWS homescreen is depicted in Figure 1. For applying PEOPLEVIEWS recommenders, there is no explicit need for being logged in. Recommenders can be selected and activated directly from the homescreen (see the tag cloud in Figure 1).

⁶ The user interface is currently only available in German.


If users are logged in, they are allowed to contribute to the development of PEOPLEVIEWS recommender applications. Only the creators of a recommender application are allowed to define user attributes. Other users can complete micro tasks in terms of evaluating items with regard to a defined set of user attributes. The list of user attributes used in our working example is depicted in Figure 2 (corresponds to the entries of Table 3).



Figure 1. PEOPLEVIEWS homescreen – the current version of the user interface is provided in German. The homescreen explains the basic functionalities of the system (development, maintenance, and execution of recommender applications).

Logged-in users are also allowed to enter new items to the recommender product catalog. The PEOPLEVIEWS representation of product catalogs is exemplified in Figure 3 (corresponds to the list of items shown in Table 2).


The interface for evaluating an item with regard to a set of user attributes is depicted in Figure 4. The screenshot depicts the evaluation of *Building Loan* with regard to the user attribute *goal*. After having completed the definition of a PEOPLEVIEWS recommender,



Namen	Fragen	Aktionen	
goal	What are your personal goals?		
runtime	When is the money needed?		
risk	Preparedness to take risks?		

Hinzufügen


Figure 2. PEOPLEVIEWS: example user attributes.

10 Einträge anzeigen  Filtern

Produkt	Recommender	Aktionen
Investment Fund A	Financial Services	
Investment Fund B	Financial Services	
Building Loan	Financial Services	
Bond	Financial Services	
Savings Book	Financial Services	

Figure 3. PEOPLEVIEWS: example of an item list.

the recommender can directly be executed. The user interface of our financial services recommender is depicted in Figure 5.



What are your personal goals?

(mehrere Antworten möglich)

- Studies
- Pension
- Speculation
- Car
- House
- World trip
- weiß nicht / keine Angabe

Figure 4. PEOPLEVIEWS: example of an item evaluation user interface (evaluation of item *Building Loan* with regard to the user attribute *goal*).

3.2 STUDYBATTLE

Recommendation-relevant filter constraints can be further exploited for generating different learning applications that are part of the STUDYBATTLE environment. STUDYBATTLE is a game-based learning environment which can be utilized as an environment for learning

product knowledge and sales practices. Examples of STUDYBATTLE games are the following.

Assign Properties. Figure 6 depicts an example user interface of a STUDYBATTLE application that implements a quiz related to knowledge about the relationship between user attributes and items. In the example, users have the task to assign items on the left hand side to user attribute values on the right hand side where each product has to be assigned to at least one attribute value and vice-versa.

Find Items. A different version of the game depicted in Figure 6 is to ask for products that fulfill certain criteria (represented by a combination of user attribute settings).

Find Incompatibilities. This game focuses on combinations of user attribute values that do not lead to a solution, i.e., users have to specify combinations of user attribute values from which they think that no corresponding solution could be found.

Maximize Requirements. The task is to identify minimal sets of requirements (from a given set of requirements *REQ*) that have to be deleted from *REQ* such that the remaining requirements lead to at least one solution. This game type reflects the principles of model-based diagnosis [6, 24], i.e., support users in learning and improving repair behavior in situations where no solution can be identified.

Maximize Items. A similar task is focused on the repair of item sets; in this context the task of users is to identify a maximal set of items from a given set of items such that there exists at least one combination of user attribute values that lead to these items (not necessarily exclusively). An additional criteria could be that at least *n* items from the original item list must remain in the result set.

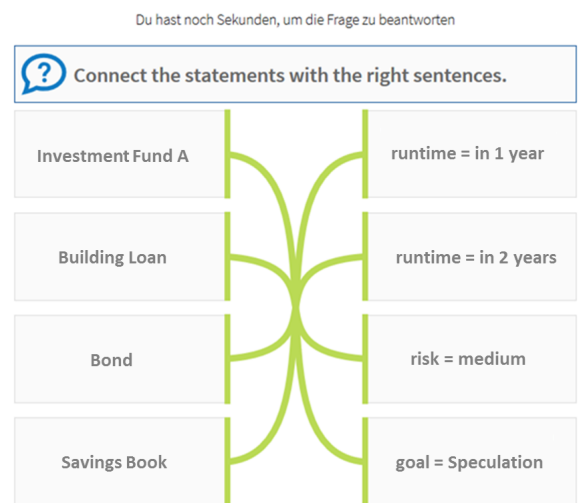


Figure 6. STUDYBATTLE "Assign Properties" learning application. The task of the user is to relate items with corresponding attribute values.

4 Preliminary Evaluation Results

Human Computation based Knowledge Acquisition. Applying Human Computation concepts [26] in the context of recommender application development and maintenance has the potential to lift the burden of enormous engineering and maintenance efforts from the

Figure 5. PEOPLEVIEWS: example of a recommender application (Financial Services).

shoulder of knowledge engineers. Micro tasks as sketched in this paper can be structured in a way that they are understandable for domain experts without a computer science background. Knowledge gained from completed micro tasks can be easily integrated into a corresponding recommender knowledge base. Due to the increasing size and complexity of knowledge bases, the development of such technologies is crucial since they help to tackle scalability issues which otherwise could cause a complete failure with regard to a company-wide recommender deployment. As such, PEOPLEVIEWS technologies can be considered as a first step towards more scalable development methods that will also help to further increase the popularity of knowledge-based (recommendation) technologies.

Usability. An initial user study has been conducted with an early version of PEOPLEVIEWS at the Graz University of Technology [10]. N=161 (15% female and 85% male) students interacted with the system with the goal to develop different recommender applications. After having completed the development, the study participants had to complete a questionnaire which was based on the system usability scale (SUS) [1]. Evaluation results regarding the SUS aspects are summarized in Figure 7. Besides usability questions, further feedback has been provided by the study participants, for example, the majority of the participants (69% of all study participants) would like to further contribute to PEOPLEVIEWS recommenders. 56% out of those participants who wanted to contribute agreed to contribute within a time frame of less than 30 minutes per week.

5 Future Work

The major goal of this paper was to provide an overview of the PEOPLEVIEWS recommendation environment. There are many issues for future work that we want to tackle and integrate corresponding solutions in upcoming PEOPLEVIEWS versions.

Weighting of Item Evaluations. In the current PEOPLEVIEWS version it is possible to assign user attribute values to items, i.e., to specify which criteria are relevant for the selection of a certain item. In future versions of PEOPLEVIEWS it will be possible to integrate weights into item evaluations. This maybe does not play a major role in financial service related recommender applications but can be important in other domains where nuances and personal tastes play a more important role. For example, in the context of recommending digital cameras, it can be important to specify degrees regarding certain camera properties, for example, the degree to which a camera is able to support sports photography.

Further Micro Tasks. In the current system version, the only micro task to be completed is to define the relationship (compatibility properties) between items and corresponding user attribute values. In future versions of PEOPLEVIEWS we will extend this list of micro tasks (see Table 9).

User Selection for Micro Tasks. An important enhancement will be the inclusion of methods that automatically select users for a given set of micro tasks and also take into account fairness in the distribution of micro tasks. As detected in our initial studies, users are willing to contribute to the further development of PEOPLEVIEWS recommenders. An important issue in this context is to find the users with the right expertise for certain tasks and also to not overload users. Our approach in this context will be to maintain user profiles which are derived from observing the activities of a user within PEOPLEVIEWS. For example, if a user selects a certain item when interacting with the financial services recommender, the keywords extracted from the corresponding item description are stored in the user profile. If (in the future) micro tasks related to similar items (items with a similar description) have to be completed, users with expertise regarding such items will be the preferred contact persons.

Games. Games will be another mechanism for data collection in

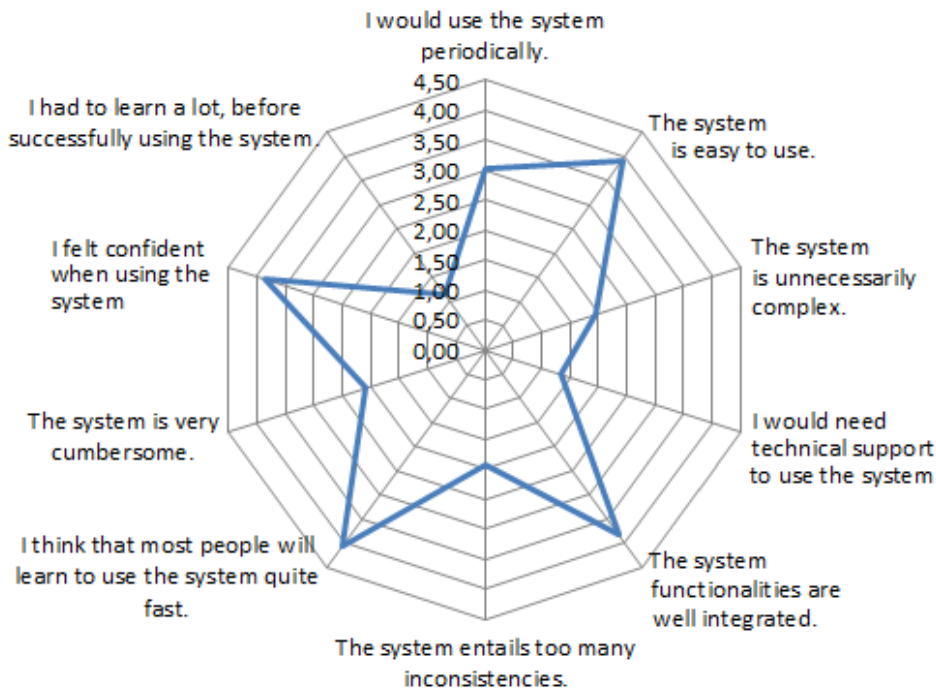


Figure 7. Results of a SUS-based usability study [1] of the PEOPLEVIEWS environment.

the PEOPLEVIEWS modeling mode. A single user game will be included that is quiz-based. The overall goal is to guess user attribute settings correctly that best describe a certain item. In a second game two users will jointly try to figure out user attribute values that best describe shown items. The more matching item evaluations exist the better the team performs.

Dependencies between User Attributes and Item Attributes. An extension of the current PEOPLEVIEWS version will be the possibility to identify direct relationships between user attribute values and technical product properties. This is not the case in the current PEOPLEVIEWS version since dependencies are only defined between user attribute values and items.

Recommendation Algorithms. The current version of PEOPLEVIEWS relies on the discussed recommendation-relevant filter constraints – item ranking is based on a utility-based evaluation (see Formula 3). In future versions of PEOPLEVIEWS we will extend the quality of recommendation algorithms by, for example, adapting the determination of support values. If, for example, additional information about the performance of a certain user is available (e.g., performance with regard to correctly completed micro tasks in the past), this information can be used to increase/decrease the weight of a user when determining support values. Finally, when users are specifying their requirements, future versions of PEOPLEVIEWS will allow the specification of preferences (weights) which indicate user preferences regarding certain requirements. This will also include approaches to the learning of weights (users should not have to specify all weights explicitly).

Inconsistency Management. Given a set of customer requirements it could be the case that no solution can be presented to the user. In upcoming versions of PEOPLEVIEWS we will focus on integrating state-of-the-art diagnosis algorithms that help to automatically determine repair actions in such inconsistent situations [15]. These repairs

name	description
item quality check	check whether a certain item belongs to a specific recommender (is an existing recommender-related item)
attribute quality check	check whether a certain attribute belongs to a specific recommender (user attribute or item attribute exists in the item domain)
attribute value quality check	check whether a certain value belongs to the domain of an attribute (user attribute or item attribute)
graphic check	check whether a certain figure belongs to a certain item
evaluate item	assign user attribute values to items
attribute value utility check	derive a ranking that shows which items best support a user attribute value

Table 9. Example list of micro tasks to be integrated in PEOPLEVIEWS.

will take into account user weights (preferences) and thus minimize the number of interaction cycles needed to find a reasonable solution. In addition to this more intelligent management of inconsistent requirements, we will integrate mechanisms that help to consolidate the set of user-specific filter constraints in order to make the resulting recommendation-relevant filter constraints more compact. Consolidation will be achieved, for example, on the basis of redundancy detection algorithms [16].

Quality Management. The major task of quality management is to assure the quality of the dataset collected on the basis of different micro tasks. Quality assurance must be capable of detecting and preventing manipulations of the dataset (also under the assumption that anonymous users are allowed to complete micro tasks), it must also identify changes to the given set of user-specific filter constraints that help to improve the prediction quality of recommendation algorithms. Quality assurance is also responsible for the generation of micro tasks that need to be completed in order to improve the overall quality of the PEOPLEVIEWS datasets. The micro tasks generated by quality assurance are summarized as an *agenda* – this agenda is forwarded to micro task scheduling that is responsible for distributing micro tasks to the PEOPLEVIEWS user community.

6 Conclusions

In this paper we gave an overview of the PEOPLEVIEWS recommendation environment which exploits concepts of Human Computation to integrate domain experts more deeply into knowledge base development and maintenance processes. PEOPLEVIEWS knowledge bases can be exploited to generate learning applications which can be used in the STUDYBATTLE environment. A major focus of this paper was to show how PEOPLEVIEWS can be applied in the context of financial service recommendation. The concepts presented in this paper have the potential to avoid scalability issues which already exist in many knowledge-based environments due to the increasing size and complexity of knowledge bases.

REFERENCES

- [1] A. Bangor, P. Kortum, and J. Miller, 'An Empirical Evaluation of the System Usability Scale (SUS)', *International Journal of Human-Computer Interaction*, **24**(6), 574–594, (2008).
- [2] R. Burke, 'Knowledge-based recommender systems', *Encyclopedia of Library and Information Systems*, **69**(32), 180–200, (2000).
- [3] R. Burke, A. Felfernig, and M. Goeker, 'Recommender systems: An overview', *AI Magazine*, **32**(3), 13–18, (2011).
- [4] R. Burke and K. Hammond, 'The FindMe Approach to Assisted Browsing', *IEEE Expert*, 32–40, (1997).
- [5] R. Burke and M. Ramezani, 'Matching recommendation technologies and domains', in *Recommender Systems Handbook*, 367–386, Springer, (2010).
- [6] J. de Kleer, A. Mackworth, and R. Reiter, 'Characterizing diagnoses and systems', *AI Journal*, **56**(197–222), 57–95, (1992).
- [7] A. Fano and S. Kurth, 'Personal Choice Point: Helping Users Visualize What it Means to Buy a BMW', in *International Conference on Intelligent User Interfaces IUI'03*, pp. 46–52, Miami, FL, USA, (2003). ACM, New York, USA.
- [8] A. Felfernig and R. Burke, 'Constraint-based recommender systems: Technologies and research issues', in *IEEE ICEC'08*, pp. 17–26, Innsbruck, Austria, (2008).
- [9] 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).
- [10] A. Felfernig, S. Haas, G. Ninaus, M. Schwarz, T. Ulz, M. Stettinger, K. Isak, M. Jeran, and S. Reiterer, 'RecTurk: Constraint-based Recommendation based on Human Computation', in *RecSys 2014 CrowdRec Workshop*, pp. 1–6, Foster City, CA, USA, (2014).
- [11] A. Felfernig, K. Isak, K. Szabo, and P. Zachar, 'The VITA Financial Services Sales Support Environment', pp. 1692–1699, Vancouver, Canada, (2007).
- [12] A. Felfernig and A. Kiener, 'Knowledge-based Interactive Selling of Financial Services with FSAdvisor', in *17th Innovative Applications of Artificial Intelligence Conference (IAAI05)*, pp. 1475–1482, Pittsburgh, Pennsylvania, (2005).
- [13] A. Felfernig, M. Schubert, G. Friedrich, M. Mandl, M. Mairitsch, and E. Teppan, 'Plausible repairs for inconsistent requirements', in *21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pp. 791–796, Pasadena, CA, (2009).
- [14] A. Felfernig, M. Schubert, and S. Reiterer, 'Personalized diagnosis for over-constrained problems', in *23rd International Conference on Artificial Intelligence (IJCAI 2013)*, pp. 1990–1996, Peking, China.
- [15] A. Felfernig, M. Schubert, and C. Zehentner, 'An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets', *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, **25**(2), 175–184, (2012).
- [16] A. Felfernig, C. Zehentner, and P. Blazek, 'Corediag: Eliminating redundancy in constraint sets'.
- [17] G. Friedrich, 'Elimination of spurious explanations', in *European Conference on Artificial Intelligence (ECAI 2004)*, pp. 813–817, Valencia, Spain, (2004).
- [18] S. Hacker and L. VonAhn, 'Matchin: Eliciting User Preferences with an Online Game', in *CHI'09*, pp. 1207–1216, (2009).
- [19] D. Jannach and U. Bundgaard-Joergensen, 'SAT: A Web-Based Interactive Advisor for Investor-Ready Business Plans', in *Intl. Conference on e-Business (ICE-B 2007)*, pp. 99–106, (2007).
- [20] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender Systems*, Cambridge University Press, 2010.
- [21] 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*, pp. 709–715, (2012).
- [22] 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).
- [23] I. Pribik and A. Felfernig, 'Towards Persuasive Technology for Software Development Environments: An Empirical Study', in *Persuasive Technology Conference (Persuasive 2012)*, pp. 227–238, (2012).
- [24] R. Reiter, 'A theory of diagnosis from first principles', *AI Journal*, **23**(1), 57–95, (1987).
- [25] S. Reiterer, A. Felfernig, P. Blazek, G. Leitner, F. Reinfrank, and G. Ninaus, 'WeeVis', in *Knowledge-based Configuration – From Research to Business Cases*, eds., A. Felfernig, L. Hotz, C. Bagley, and J. Tiitonen, chapter 25, 365–376, Morgan Kaufmann Publishers, (2013).
- [26] L. VonAhn, 'Human Computation', in *Technical Report CM-CS-05-193*, (2005).