Using a Semantic Multidimensional Approach to Create a Contextual Recommender System

Abdulbaki Uzun
Service-centric Networking
Deutsche Telekom Laboratories, TU Berlin
abdulbaki.uzun@telekom.de

Christian Räck
Competence Center FAME
Fraunhofer Institute FOKUS
christian.raeck@fokus.fraunhofer.de

Abstract

Item recommendations calculated by recommender systems mostly in use today, only rely on item content description, user feedback and profile information. In modern mobile services, however, contextual information and semantic knowledge can play a significant role concerning the quality of these recommendations. Therefore, the *SMART Recommendations Engine* of Fraunhofer FOKUS is extended by the *SMART Multidimensionality Extension* and the *SMART Ontology Extension* that enable the recommender to incorporate contextual and semantic data into the recommendation process. The demonstration of the *SMART Ontology Extension* visualizes that the preciseness of recommendations can be increased by exploiting implicit and indirect knowledge, classification and location information gained from ontologies when generating recommendations in the scope of an exemplary food purchase scenario.

1 Introduction

Today, people are confronted with a large amount of information in the World Wide Web [Shenk, 1998]. Receiving a small subset of desired and filtered content through standard search engines turns out to be very difficult. And it becomes quite impossible, if user specific needs and interests should be taken into consideration.

Recommender systems handle this issue by filtering relevant information and providing personalized content recommendations to users based on their profile and feedback. Numerous recommendation methods were designed over the years to improve the accuracy of recommendations. The most popular ones are content-based and collaborative filtering algorithms or hybrid approaches comprising both them [Adomavicius and Tuzhilin, 2005].

The results delivered by hybrid methods are often acceptable. However, they only depend on content descriptions and ratings given to items, and user profile information. In a time when mobile and location-based services become very popular, context information and semantic knowledge can play a decisive role in order to significantly improve the preciseness of personalized recommendations. If John, for example, is vegetarian, eats only organic food, tries to live economical and goes shopping nearby, it does not make sense to recommend him groceries in stores far away or only in discounters without taking his preference for vegetarian and organic food into consideration. This example shows that context as well as semantic information (e.g. implicit knowledge about which food products fit to certain eating preferences) are important to satisfyingly answer a user's grocery recommendation request.

In order to harness the potential of contextual and semantic information, the generic recommender system of Fraunhofer FOKUS, the *SMART Recommendations Engine* [Raeck and Steinert, 2010], has been extended by two new recommender extensions. Inspired by the work of [Adomavicius *et al.*, 2005], the *SMART Multidimensionality Extension* enhances the two-dimensional matrix representation of recommender data by a multidimensional recommendation model allowing the integration of additional contextual information when generating recommendations. The *SMART Ontology Extension*, on the other hand, enables the recommender to incorporate contextual and semantic information gained from ontologies (e.g. implicit and semantic knowledge about a user and his preferences, location, time or ontological classification information). For this purpose, the extension provides a tool to exploit semantic data stored in ontologies and perform context and semantic filtering on the recommender database using several filters. Both extensions can be used independently from each other or together depending on the given scenario and application.

This paper is organized as follows: First, an overview about related work in the field of context-aware and semantic recommender systems is presented. Afterwards, a background about the *SMART Recommendations Engine* is given. Following that, concepts for the *SMART Multidimensionality Extension* are described. Section 5 explains the *SMART Ontology Extension* including the automated mapping of ontological information into the recommender's data model. In section 6, the functionality of the *SMART Ontology Extension* is demonstrated within the scope of a food purchase scenario.

2 Related Work

Contextual and semantic information is incorporated in the field of recommendations in various ways. A context-aware collaborative filtering system is presented by [Chen, 2005], which generates item recommendations for a user based on different context situations. In order to achieve that, the traditional collaborative filtering is extended, so that feedback of like-minded users in a similar context can be used to recommend items to the active user in his current context.

[Adomavicius *et al.*, 2005] propose a multidimensional recommendation model, in which additional contextual di-

mensions can be added to the traditional *User x Item* matrix representation. Recommendations are calculated by using the *reduction-based approach*, which reduces the problem of multidimensional recommendations to a traditional two-dimensional matrix in the required context, so that widely known traditional recommendation techniques can be applied after the reduction is done.

Another methodology is suggested by [Farsani and Nematbakhsh, 2006], which recommends semantic products to customers in the context of eCommerce based on product and customer classification via OWL.

[Kim and Kwon, 2007], on the other hand, developed an ontology model with a multiple-level concept hierarchy for a grocery store scenario with four different ontologies: a *product*, *location*, *consumer* and *record* ontology. From the *product* ontology, most relevant products are taken using user information modeled in the *consumer* and *record* ontology. Recommended products are presented in a concept hierarchy ranging from most specific to most broad. When users select some of these concepts, the context of the request is subsequently refined enhancing the specificity of the provided results.

Another interesting approach is demonstrated by [Yu *et al.*, 2006]. They present a context-aware media recommendation platform called *CoMeR*, which uses an OWL ontology context description, a *N x M*-dimensional model and a hybrid processing approach to support media recommendations for smart phones.

In another paper by [Setten *et al.*, 2004], the integration of a context-aware recommender system into the mobile tourist application *COMPASS* is described, where users get touristic information and services recommended based on their interests and contexts.

COReS, which stands for *context-aware, ontology-based recommender system for service recommendation*, is another example. It was developed by [Costa *et al.*, 2007] and this recommender system extends the capabilities of the *INFRAWARE* [Pereira Filho *et al.*, 2006] service platform by supporting service selection and user needs satisfaction for a certain context.

Previous research activities are either focused on the integration of context or semantic information. However, incorporating both – context information and semantic domain knowledge – would increase the preciseness of recommendations decisively. The food scenario shows that the integration of both types of data is necessary to satisfyingly answer a recommendation request. For example, in order to generate accurate grocery recommendations, the system has to be aware of the location of the user (context) and has to be capable of using implicit knowledge in order to relate the user's eating preferences to the right groceries (semantic information). That's why, Fraunhofer's engine was extended using both types of data.

3 SMART Recommendations Engine

The *SMART Recommendations Engine* is a generic recommender system that enables internet businesses, rich media and entertainment services or SMEs deliver a more personalized experience by providing recommendations in their respective application domain. By offering a flexible, general purpose algorithmic model, the engine makes it possible to formulate application specific recommendation algorithms. These algorithms and the application specific data model are declared at configuration time by assembling selected components. By adding custom components through the provided API, the capabilities of the recommender can be extended in order to meet specific application demands. These custom components can be built from functional groups, such as basic mathematical operations, similarity and relevance computations, sorting and filtering, and data access.

In the system, data is represented in an entity-relationship-like data model. An entity type that includes a set of entities is named *domain*, whereas relations between domain entities are represented by *matrices* (see Figure 1). A user domain, for example, can contain all users, while an item domain can comprise all items in a specific application. Ratings given by a user to a certain item can be represented by a rating matrix, whose rows and columns are associated to the these domains.

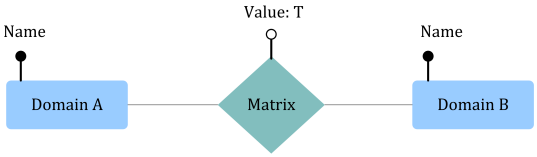


Figure 1: Recommender Data Model

Recommendation algorithms calculate relevance values for each *User x Item* pair. Differing on the given application, the recommendation algorithm is assembled at runtime configuration by defining a network of matrix transformation components (e.g. a *similarity* computation component). The matrix operations are applied on a data set in a hierarchical manner leading to the estimated utility function at the top node of the tree. Figure 2 shows such a generic algorithm hierarchy, where the nodes represent matrix operations and data is propagated along the edges in form of matrices.

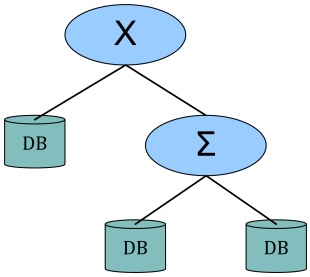


Figure 2: Generic Algorithm Hierarchy

The engine also provides a custom query language called *Sugar Query Language (SuQL)*, which is used to request recommendations and related data at runtime. Via *SuQL* various recommendation algorithms can be selected and combined. Constraints on the item set to be recommended can also be specified, so that only items that fulfill these constraints are included in the final recommendation. For this purpose, the recommender already offers a variety of sorting and selection filters, which can be used to alter the result set by certain properties.

One example for a filter is the *Proximity Filter*, which in combination with a Geo-Location typed domain, an *Item x Location* matrix, a given center position and a maximum range, is capable of selecting items (e.g. shops) in a given geographic region. By using the lookup capabilties of the *SMART Ontology Extension* (see section 5), items

can be filtered based on location constraints, like finding all grocery products sold in shops located within a given distance from the user's current location.

4 SMART Multidimensionality Extension

Contextual information can be exploited in the recommendation process by enhancing the level of dimensions from the traditional two-dimensional paradigm to multiple dimensions. Therefore the *SMART Recommendations Engine* is upgraded by the *SMART Multidimensionality Extension*, which is able to handle multidimensionality and hence provide more precise recommendations.

4.1 Enhanced Data Model

This extension enhances the generalized data model of the *SMART Recommendations Engine* for multidimensionality purposes by binding more than two domains to a rating matrix. Figure 3 shows the enhanced data model in an entity-relationship notation.

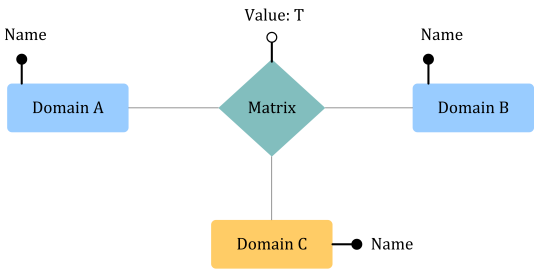


Figure 3: Enhanced Data Model

However, the recommender is designed to work two-dimensional; a matrix can only consist of two domains (one row domain and one column domain). That's why, the enhanced data model has to be adapted to the standard data model of the recommender. By merging several domains to one domain and hence mapping multiple dimensions to a two-dimensional matrix, the two-dimensional recommender data model can be kept with the advantage that multidimensionality features can be utilized at the same time (see Figure 4).

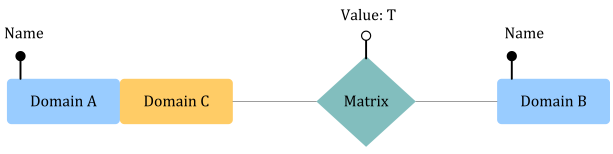


Figure 4: Merged Domains in the Enhanced Data Model

The two-dimensional matrix representation of multiple dimensions can be done through *serialization* by use of a hierarchical index. Each slice of the multidimensional cube is stored into one two-dimensional matrix making it possible to access the desired element with the help of an index (e.g. *Slice1.User#1*). Figure 5 shows a general MD-2D-Mapping for the enhanced data model.

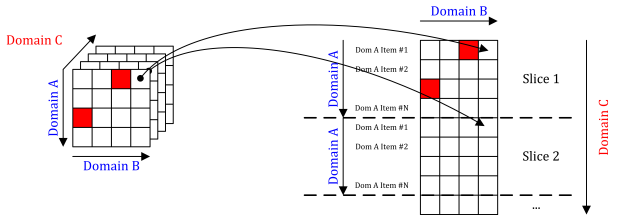


Figure 5: General MD-Mapping to 2D

4.2 Multidimensional Recommendation Algorithms

Various standard recommendation algorithms, such as user or item-based collaborative filtering, can be extended by the extension to multidimensional algorithms.

The standard *user-based collaborative filtering* algorithm applied by the *SMART Recommendations Engine*, for example, determines similar users by using ratings given for items by them. Based on this similarity and on the ratings given to items, predictions for new items are calculated. This algorithm can be upgraded to a *multidimensional user-based collaborative filtering* algorithm by merging several domains to one column domain.

Assume that there is a rating matrix *R[U :: C][I]* in the *User :: Context x Item* space, in which *User :: Context* displays users with a certain context.

$$R[U :: C][I] = User :: Context \times Item \tag{1}$$

The similarity between users in the same context is determined based on their ratings for items by applying a *similarity* transformation on *R[U :: C][I]*. This calculation also identifies similarities between users in different context situations, which can be useful for later recommendations.

$$Sim(R[U :: C][I], R[U :: C][I]) = User :: Context \times User :: Context \tag{2}$$

Having a similarity matrix *Sim[U :: C][U :: C]* and the rating matrix *R[U :: C][I]*, a *matrix product* transformation can be applied to calculate item predictions for users in a certain context.

$$PredictionP[U :: C][I] = Sim[U :: C][U :: C] * R[U :: C][I] = User :: Context \times Item \tag{3}$$

These predictions can be further processed and sorted to generate new recommendations for users in certain contexts.

To better illustrate the new *multidimensional user-based collaborative filtering algorithm*, an example query "*John wants to buy food for a soccer evening*" is used. This query includes three dimensions: the *user* (John), *food* and the *event* dimension (here: soccer evening). For that matter, the similarity between users, who bought food for a soccer evening, is calculated using the *User :: Event x Food* rating matrix twice for the similarity computation (see Figure 6).

By applying a matrix product transformation on the calculated similarity matrix *User :: Event x User :: Event* and the rating matrix, food predictions for John in the context of a soccer evening can be identified. These predictions can further be filtered and sorted based on John's eating habits, for example, and the rest result can

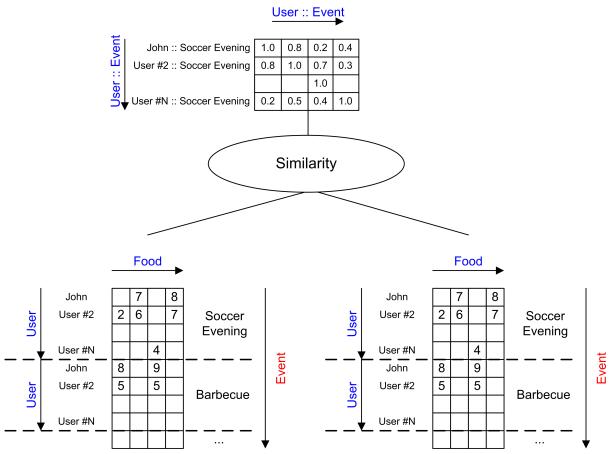


Figure 6: Exemplary Multidimensional User-based Collaborative Filtering Algorithm – Part 1

be presented as generated recommendations for John (see Figure 7).

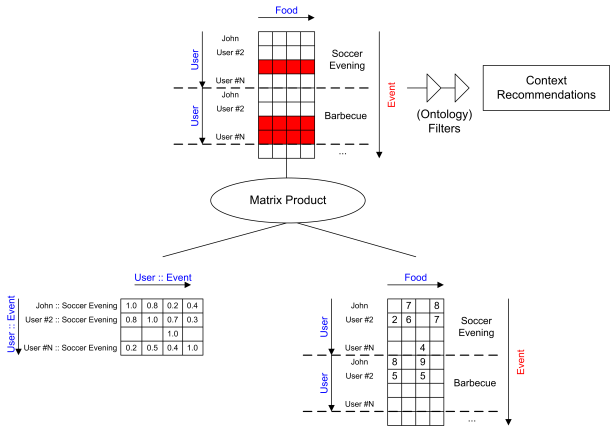


Figure 7: Exemplary Multidimensional User-based Collaborative Filtering Algorithm – Part 2

5 SMART Ontology Extension

Data that provides additional and useful information to the traditional *User x Item* representation, such as taxonomies, implicit and indirect knowledge about a user's preferences or location information can enhance the quality of recommendations.

Semantic web ontologies offer complex knowledge representation possibilites with which such semantic information can be modeled (the *Web Ontology Language (OWL)* [Dean and Schreiber, 2004]). Implicit knowledge, i.e. knowledge that is not directly included in an ontology, can be inferred using reasoning technologies.

The *SMART Ontology Extension* provides features with which the *SMART Recommendations Engine* is capable of exploiting semantic information from ontologies including information gained from reasoning mechanisms. In that way, implicit and indirect knowledge as well as taxonomies become available when generating recommendations.

The extension is divided into two main parts: The first part is the *Ontology Mapping*, where all relevant data available in pre-designed ontologies is mapped onto the data model of the recommender. The second part performs semantic filterings on the previously extracted data set using the *Ontology Filter* in order to generate semantic recommendations.

The functionality of the extension is explained using pre-designed ontologies for the food scenario. It comprises users with special eating habits (such as *vegetarian*, *vegan* or *organic*), who want to buy food from grocery stores nearby based on their eating preferences.

5.1 Ontology Mapping

The main constructs included in OWL ontologies are individuals, classes, a class hierarchy, object properties, datatype properties and restrictions. These constructs can be mapped onto the data model of the recommender, so that the recommendation engine becomes capable of handling ontology information.

Each class hierarchy in OWL is represented by a domain with the name of its respective root class (e.g. *Food*, *User* or *GroceryStore*). Individuals are items of a certain class and are mapped as items of their respective class hierarchy recommender domain. Furthermore, all classes in an ontology become elements of the general *Class* domain. While a *Food x Class* matrix, for example, shows to which classes a food product belongs to, taxonomies become clearly recognizable in a *Class x Class* matrix. Figure 8 shows an exemplary ontology class structure mapping for the food scenario.

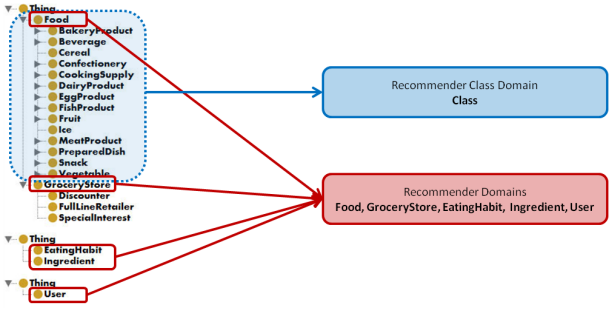
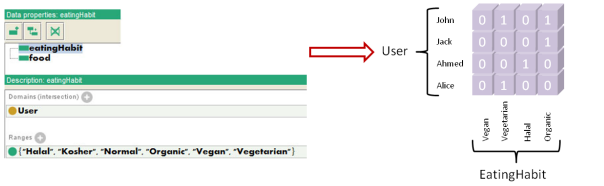


Figure 8: Root Class Representation

Object properties in OWL form a relation between individuals, whereas datatype properties build a relation between individuals and data types, such as *Integer*, *String* or *Boolean*. Matrices display relationships between domains and are therefore equal to object and datatype properties. The domain and range an OWL property has are both represented as domains in the recommender and their values are elements of these domains. If a property's domain or range is defined as an ontology class, the recommender domain name corresponds to the class's name as well. Otherwise, it corresponds to the property's name.

In the example of Figure 9, the property *eatingHabit* has a domain named *User* and data range values, such as *Vegan*, *Vegetarian* or *Halal*. Therefore the recommender domains are *User* and *EatingHabit* (name of the OWL property) defining a matrix. The values in the matrix are also directly mapped from the ontology and represent the assigned property values to individuals (e.g. *John eatingHabit Vegetarian*).

Based on the *Ontology Mapping* naming conventions, the *Ontology Mapping tool* automatically creates a recommender-compliant database including domains,



a) Datatype Property in the Ontology a) Matrix Representation of the Property

Figure 9: Datatype Property Representation

matrices and filters. Knowledge gained by reasoning mechanisms is also exported via the tool making it possible to exploit implicit knowledge in the recommendation process. As seen in the screenshot of the tool (Figure 10), the left part of the GUI represents the *Ontology Import* and *Ontology Export* functions, whereas the right part comprises all functions concerning the recommender.

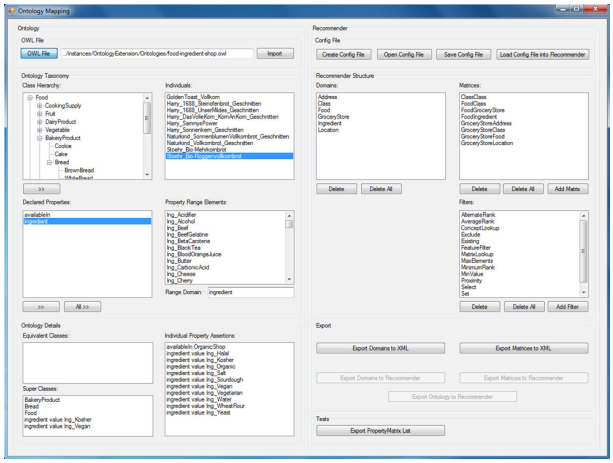


Figure 10: Ontology Mapping Tool

5.2 Ontology Filter

After successfully exporting ontology data into the recommender's database, the *Ontology Filter* can process the ontology information in the engine by performing a semantic filtering. This new filter extends the available filter list of the *SMART Recommendations Engine* and makes it possible to query implicit and indirect ontology knowledge at runtime.

Two different matrix operations are provided by the *Ontology Filter*, the *Concept Lookup* and the *Matrix Lookup*. In order to accomplish their task, both lookups use standard set operations, which are also features of the *Ontology Filter*. The *Concept Lookup* uses the *Union* and *Intersection* set operations, while the *Existential quantification* and *Universal quantification* are utilized by the *Matrix Lookup* operation. If needed, the result set delivered by the *Ontology Filter* can be inversed by the *Not* set operation. This can be helpful in order to determine all products that can be eaten by John instead of all that are forbidden for him, for example.

Ontology concepts can be looked up in the recommender by using the *Concept Lookup*, which needs at least two different matrices for the operation. Hereby, the column domain of the first matrix has to be the row domain of the second matrix. Applied on the first matrix, the *Concept Lookup* filters certain column elements for one single row element based on given constraints. These constraints can be generated by any available *SMART Recommendations Engine* filter, such as the *Existing* or *Feature* filters. The filtered column elements – now selected rows in the row domain of the second matrix – build the basis for further operations. All selected rows will be processed again individually depending on given filters. The calculated result sets of each selected row will be returned and will then be either unified or intersected based on the selected set operation (*Union* or *Intersection*). Figure 11 shows a *Concept Lookup* example, in which ingredients are looked up that are **not** allowed to be eaten by John.

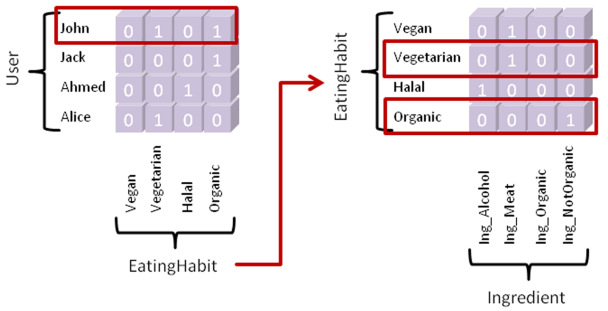


Figure 11: Concept Lookup Example

The *Matrix Lookup* filters information in a matrix based on a given column domain result set of another matrix. Therefore, it also requires the use of two different matrices, whereas the column domain of the first matrix remains the column domain of the second matrix. Rows of the second matrix will be filtered based on the given column domain result set and a predefined set operation (*existential quantification* or *universal quantification*). The result is one set of filtered row elements. In Figure 12, an exemplary *Matrix Lookup* can be seen. All food products are looked up that are **forbidden** for *organic* eating people.

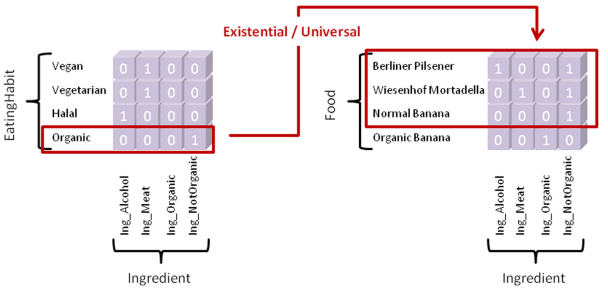


Figure 12: Matrix Lookup Example

An overview of all supported operations of the *Ontology Filter* is given in Table 1.

Complex recommendation queries require combining both lookups to single a *Concept and Matrix Lookup* operation, which is exemplary demonstrated in section 6.

6 Demonstration

Powerful contextual/semantic recommendations can be computed by the recommender using the *SMART Ontology Extension* in combination with the *Proximity Filter*. The ontology data present in the database of the *SMART Recommendations Engine* facilitates the recommendation query

Filter	Matrix Ops.	Set Ops.
Ontology Filter	Concept Lookup	Union
		Intersection
	Matrix Lookup	Existential quantification
		Universal quantification
		Not

Table 1: Ontology Filter Operations

requests of a user by reducing his effort to manually describing his personal needs and interests in a detailed way. In conjunction with the *Proximity Filter*, the recommendations are also filtered by their location leading to a great user experience when using a mobile shopping service, for example. This function is going to be demonstrated in this section by taking the following mobile grocery shopping service scenario as a basis:

Assume that John is vegetarian and eats only organic food products. He wants to buy groceries nearby his location and that's why he wants his shopping application to give him recommendations based on the food categories he prefers. After the selection of food categories for his shopping cart, John gets food products recommended for each category fitting his eating preferences and sorted by their relevance based on John's previous purchases. The grocery stores that sell these products and are in a certain range to John are also listed.

Eating preferences, such as eating vegetarian or organic, include some sort of implicit and indirect knowledge. For example, in order to be able to specify vegetarian preferences, the system must know what kind of food vegetarians do not eat. It further has to have the information, which ingredients are included in each grocery, so that non-vegetarian food can be filtered based on the list of non-consumable ingredients. Since semantic ontologies are predestined to model this kind of dependencies, three ontologies were designed including a complete data set to map relevant information given in the food scenario.

The *food-ingredient-shop* ontology consists of a classification of food categories, concrete grocery products including their ingredients and shops, in which they can be found. The *eatinghabit-ingredient* ontology, on the other hand, models relations between certain eating preferences and the ingredients that are either allowed or forbidden to eat. User profiles are stored in the *user-profile* ontology. All data modeled in these ontologies is exported into the recommender database via the *Ontology Mapping tool*. Figure 13 shows parts of the created data model after a successful export.

Using the *SuQL*, the *SMART Recommendations Engine* can now provide contextual/semantic recommendations based on the available data set and the implemented recommendation algorithms. For the food scenario, assume that John wants to buy snacks and bread in the range of 2 kms from his location and that he already purchased the brown bread product *Naturkind_SonnenblumenVollkornbrot_Geschnitten*. The *SuQL* query is constructed as follows: At first several semantic filterings are performed using the lookup operations several times in order to identify all snack and bread products that fit John's eating preferences and his location. Afterwards, these elements are sorted by their relevance and limited to a certain number depending on the relevance predictions calculated by the recommendation algorithms.

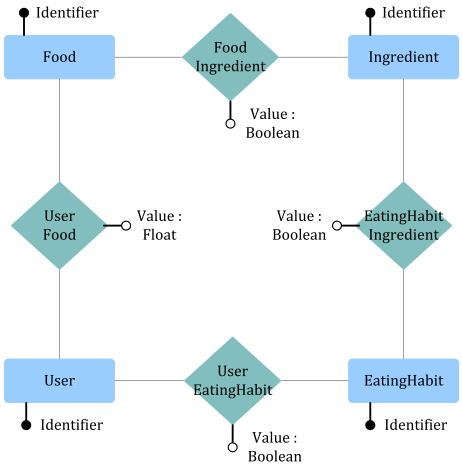


Figure 13: Food Scenario Ontology Data Model

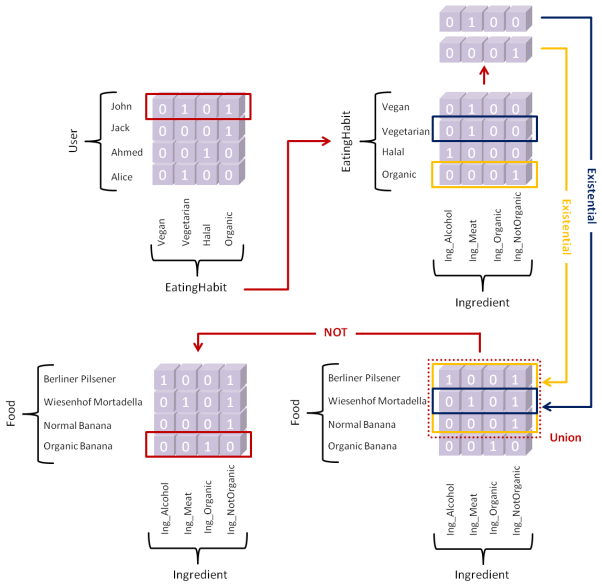


Figure 14: Concept and Matrix Lookup

Figure 14 shows a semantic filtering example for John, in which the *Ontology Filter* first performs a *Concept Lookup* in the *User x EatingHabit* matrix that looks up his eating preferences. In the second matrix (*EatingHabit x Ingredient*), John's eating preferences (*vegetarian* and *organic*) are mapped to the ingredients. (Note: Due to the fact that organic food products cannot be identified by their ingredients, artificial ingredients, such as *Ing_Organic* or *Ing_NotOrganic* were created and related to the food products.) The *Matrix Lookup* then looks up all groceries in the *Food x Ingredient* matrix for vegetarians and organic eating people individually. Finally, both result sets are unified to one single result set and inversed by the *Not* set operation in order to get a set of groceries, which can be eaten by John. These groceries are also filtered by their categories, so that only snacks and bread products remain.

Another semantic operation in form of a *Matrix Lookup* is utilized, so that all grocery stores near to John's location

that sell these food products can be looked up. This is done by using the *Proximity Filter*, where the location of John is specified as well as the range to look for. All grocery stores near to John's position are delivered to the *Matrix Lookup*, which then performs a lookup to find all filtered snacks and bread products that are available in these shops.

In a final step, the recommendation algorithm is used in the recommendation process. Even though the engine can also generate recommendations based on different *collaborative filtering* algorithms, this paper focuses on an extended version of the *content-based filtering* approach using the ontology taxonomy as content meta-data. This approach – named as the *ontology-based filtering* algorithm – calculates relevance predictions using the similarity of food products based on their cagetory as content meta-data (e.g. *brown bread* is more similar to *white bread* than to snacks) and the implicit user feedback given by users automatically when purchasing items. This user feedback affects the final recommendations in that way that groceries of a category the user has already bought products before become more relevant to him than products of other nodes in the tree.

The ontology taxonomy stored in the *Food x Class* matrix is used to compute a similarity between groceries by means of their categories leading to a food similarity matrix with the dimensions *Food x Food*. *User x Food* relevance predictions based on the taxonomy information can be gained by applying a matrix product transformation on the *User x Food* feedback matrix and the *Food x Food* similarity matrix (see Figure 15).

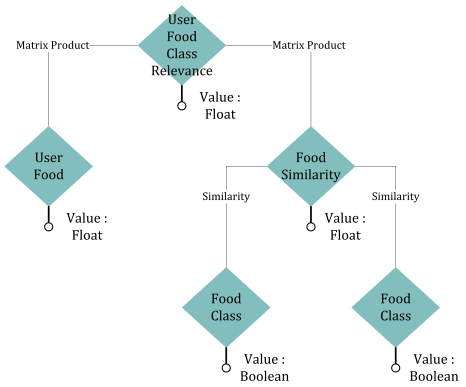


Figure 15: Ontology-based Filtering Algorithm

Figure 16 shows the *SuQL* recommender response to John's query. John bought *Naturkind_SonnenblumenVollkornbrot_Geschnitten* before, which is from the bread product category *BrownBread*. This and all other *BrownBread* products are most relevant to him. The relevance value decreases the more vegetarian and organic bread products are in other nodes of the *Bread* tree. There is only one snack fitting his eating preferences with a low relevance value since John did not purchase any snacks yet. As a result, five bread products and one snack are presented in combination with the grocery stores nearby that sell these products.

7 Discussion of the Approach

One way to compare the presented approach with widespread memory-based and model-based recommendation algorithms is to compare the accuracy of these approaches based on a common metric such as the mean average error (MAE). For that, a suitable set of data points for training and testing is needed. The chosen application domain of the presented demonstration scenario was predetermined by the project's context in which our approach was developed. This context did not provide us with a suitable data set for this kind of practical testing. However, a closer examination of our approach shows that a benchmark recommendation algorithm, which has been adapted to make use of the product hierarchy that is stored in the ontology, provides better relevance values by taking the relation of different food items into account. The relevance values of unrelated items decrease in less relevant or unsuitable nodes of the product hierarchy, so that the overall recommendation quality in terms of the achieved accuracy increases.

Figure 16: Ontology-based *SuQL* Recommender Response

8 Conclusion

This paper introduced the generic recommender system, the *SMART Recommendations Engine* of Fraunhofer FOKUS, which is capable of providing recommendations for different types of applications. In order to make the engine meet the recommendation quality requirements of modern mobile services, the recommender was extended by two new extensions. The *SMART Multidimensionality Extension* provides multidimensionality capabilities to the so far two-dimensional recommender system. This upgrade enables the recommender to deal with additional context dimensions in order to generate more accurate recommendations taking user's current contextual situation into account. The *SMART Ontology Extension*, on the hand, exploits semantic ontology data by exporting all relevant information given in pre-designed application-dependent ontologies into the data structure of the *SMART Recommendations Engine* enabling the recommender to use semantic knowledge or ontology taxonomy information for recommendation purposes.

The demonstration showed that in conjunction with the *Proximity Filter*, the *SMART Ontology Extension* provides an added-value to the engine by generating semantic and contextual recommendations.

References

[Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[Adomavicius *et al.*, 2005] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.

[Chen, 2005] Annie Chen. Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1110–1111, New York, NY, USA, 2005. ACM.

[Costa *et al.*, 2007] A.C. Costa, R.S.S. Guizzardi, G. Guizzardi, and J.G.P. Filho. Cores: Context-aware, ontology-based recommender system for service recommendation. *19th International Conference on Advanced Information Systems Engineering (CAISE07)*, 2007.

[Dean and Schreiber, 2004] M. Dean and G. Schreiber. Owl web ontology language reference, 2004. http://www.w3.org/TR/owl-ref/.

[Farsani and Nematbakhsh, 2006] H.K. Farsani and M. Nematbakhsh. A semantic recommendation procedure for electronic product catalog. *International Journal of Applied Mathematics and Computer Sciences*, 3:86–91, 2006.

[Kim and Kwon, 2007] S. Kim and J. Kwon. Effective context-aware recommendation on the semantic web. *International Journal of Computer Science and Network Security*, 7:154–159, 2007.

[Pereira Filho *et al.*, 2006] J.G. Pereira Filho, R.M. Pessoa, and C.Z. Calvi. Infraware: A support middleware to context-aware mobile applications (in portuguese). *In Proceedings of the 24th Simpsio Brasileiro de Redes de Computadores*, 2006.

[Raeck and Steinert, 2010] Christian Raeck and Fabian Steinert. Fraunhofer institute fokus, smart recommendations engine, 2010. http://tinyurl.com/3xdsffz.

[Setten *et al.*, 2004] Mark Van Setten, Stanislav Pokraev, Johan Koolwaaij, and Telematica Instituut. Context-aware recommendations in the mobile tourist application compass. In *In Nejdl, W. and De Bra, P. (Eds.). AH 2004, LNCS 3137*, pages 235–244. Springer-Verlag, 2004.

[Shenk, 1998] David Shenk. *Data Smog: Surviving the Information Glut*. Harper San Francisco, 1998.

[Yu *et al.*, 2006] Zhiwen Yu, Xingshe Zhou, Daqing Zhang, Chung-Yau Chin, Xiaohang Wang, and Ji Men. Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing*, 5(3):68–75, 2006.