Using Appropriate Context Models for CARS Context Modelling

Christos Mettouris and George A. Papadopoulos

Abstract Most context-aware recommender systems in the literature that use context modelling have the tendency to develop domain and application specific context models that limit, even eliminate any reuse and sharing capabilities. Developers and researchers in the field struggle to design their own context models without having a good understanding of context and without using any reference models for guidance, often resulting in overspecialized, inefficient or incomplete context models. In this work we build upon prior work to propose an enhanced online context modelling system for Context-Aware Recommender Systems. The system supports CARS developers in the process of building their own context models from scratch, while it supports at the same time sharing and reuse of the models among developers. The system was tested with a real dataset with positive results, as it was able to support context model development with instructions to the developer, model comparison, useful statistics, recommendations of similar models, as well as alternative views of context models to aid the developer's task.

Keywords Context modelling system · Context-Aware recommender systems · Application context model · Context instance model · Context variables · Context dimensions

C. Mettouris (✉) · G.A. Papadopoulos
Department of Computer Science, University of Cyprus, 1 University Avenue, 20537, CY-2109 Nicosia, Cyprus
e-mail: mettour@cs.ucy.ac.cy

G.A. Papadopoulos
e-mail: george@cs.ucy.ac.cy

S. Kunifuji et al. (eds.), *Knowledge, Information and Creativity Support Systems*, Advances in Intelligent Systems and Computing 416,
DOI 10.1007/978-3-319-27478-2_5

1 Introduction

A well-known and effective solution to the information overload modern life experiences at all fields is the usage of Recommender Systems (RS). Information overload refers to the vast amount of information users have to access nowadays: users can get lost, disappointed and frustrated for failing to retrieve the desired and needed information at a given time. RS use a variety of filtering techniques and recommendation methods to provide personalized recommendations to their users, mostly by using information retrieved from the user profile, from user's usage history, as well as item related information [5, 9]. However, traditional RS use limited or none contextual information to produce recommendations, as opposed to the Context-Aware Recommender Systems (CARS) that focus in using contextual information to enhance recommendations [2]. Context was first utilized into the recommendation process by Adomavicius by proposing three approaches: the Pre-filtering approach, the Post-filtering approach and the Multidimensional Contextual Modelling approach [1, 2]. Context modelling is important for modelling the contextual information to be used during the recommendation process.

An important contextual modelling issue in CARS is the development of domain specific and application specific context models that only represent information on the particular application domain (e.g. recommendation of movies). Our review on RS [11] had revealed that most CARS and semantic RS in the literature are domain and application specific, meaning that they cannot be applied in other domains. By designing domain and application specific context models, many different and very specific models are produced with no reuse and sharing capabilities.

Another problem is that developers and researchers attempt to design their own models based on their own knowledge and skills and more importantly without using any reference model, without any guidance and strictly focused on the application at hand, often resulting in overspecialized, inefficient or incomplete contextual models.

We had partially addressed the above contextual modelling problem in prior works, at first by proposing a generic, abstracted contextual modelling framework for CARS which developers and researchers can use theoretically to be guided through the process of properly defining the context for their application [11], and later by developing an online "Context Modelling System and Learning Tool" [10] based on [11], which is able to teach and guide developers towards a more efficient, effective and correct selection and usage of context attributes for building their own application model, allowing at the same time for sharing and reuse of context models among applications, regardless of the domain they belong to.

The modelling framework in [11] was essentially a model template in UML built in the Eclipse Modelling Framework (EMF) [7]. Although this framework was developed as a UML class diagram, it was mainly a theoretical tool rather than a modelling tool since: (i) it was not an easy and straight-forward procedure for developers to extend or instantiate a UML class diagram in order to build their own context models, (ii) it was time consuming, (iii) it required programming

knowledge and skills and (iv) it did not offer guidance and learning of important concepts. The online "Context Modelling System and Learning Tool" [10] aimed at solving the aforementioned problems by specifying an easy to use UI for the developers and researchers to be able to effectively and efficiently build their models, share them with others, as well as reuse models of others. The above can be accomplished without any programming skills being required by the user. The system focuses also on learning, being able to introduce developers and new researchers with modern concepts from CARS research, as well as their role in a context model and a recommendation process [10].

In this work, we have extended and finalized the work conducted in [10] based on feedback received by experts, aiming to advance the functionality of the system towards a more effective context model development, sharing and reuse. The work was extended with important system functionality and the new *CARS Context Modelling System* [4] was tested by us in real settings by using a dataset released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems—HetRec 2011 [8]. More to the point, we have designed and implemented: (i) the validation of application contexts through context instances, (ii) the comparison of application context models regarding their common context variables, (iii) the recommendation of similar application context models to the user and (iv) the inclusion of the *context dimensions* concept [2] in the system and the enablement of a "context dimensions" view of the context models.

In Sects. 5–7 we discuss the above important additions, after presenting related work in Sect. 2, an introduction to the system concept in Sect. 3 and a presentation of its functionality in Sect. 4. Section 8 discusses system testing and Sect. 9 closes the paper with conclusions and future work.

2 Related Work

During our research [11] we have reviewed a number of CARS and semantic RS to opine whether the context models used were application/domain specific or generic. While domain specific models focus extensively on a particular domain, generic models do not and focus on being able to facilitate *any* application specific domain. This research revealed that most CARS and semantic recommenders in the literature use domain specific models [11], meaning that they cannot be applied for usage in other domains. A number of generic recommenders also exists that either apply to some generic application area, or can be applied to more than one domain by linking domain specific ontologies to their own data and knowledge pool in order to gain domain-aware knowledge and provide domain-aware functionality. Although some of the semantic and contextual models attempt to be more generic, the majority represent information that either concern a particular application domain (e.g. movies), or a more abstracted domain (such as products in general, web services, e-learning).

To the best of our knowledge no attempts have been made towards developing a context modelling tool that could facilitate the development of truly generic contextual models for CARS and the definition of their contextual entities, so that CARS developers be able to extend/update and reuse them to construct application specific models for their needs. Such a tool would simplify the process of contextual modelling in CARS and enable context uniformity, share and reuse; this is the motivation for this work.

3 System Concept

In our work we have followed the representational view of context [2, 6], meaning that, as in most CARS, the context of an application is defined through a predefined set of observable context attributes of static (not dynamic) structure which does not change significantly over time (as opposed to the interactional view of context where context is not necessarily an observable feature of an interaction [3]). Please note that by static structure we do not mean that the context itself is static, e.g. the context attribute *user location* has a well-defined static structure but is itself a dynamic context since it continuously changes values as the user changes locations. Therefore, we assume that there is a predefined, finite set of contextual attributes in a given CARS application and that each contextual attribute is defined in our system as a context variable. As an example of the interactional view of context the reader may refer to [3] in which the user is modelled based on human memory models proposed in psychology, where user preference models for previous interactions are stored within the user's long term memory, while the current user's model is stored in the user's short term memory. Then, the short term memory is used to retrieve information from the long term memory in order to be used to generate recommendations for the user.

The CARS Context Modelling System is presented online at [4]. On the right side of the main webpage, an image of the context modelling framework is presented for reference [11], while on the left side various options are presented. The red colour text throughout the system [4] is clickable and provides information on important system concepts that have to do with CARS research, as well as information on how to use the system. The information is provided in pop-up text boxes, as well as on the context modelling framework image on the right side for reference and easier comprehension of the information.

Many complicated concepts related to CARS research are used by the system in order to construct context models [4]. Concepts like "the multidimensional context-aware recommendations: *Users* × *Items* × *Context* *Ratings*", "context variables", "application contexts" and "context instances" need to be well understood by CARS developers and researchers in order to be able to use the modelling system to create their own application context models. To assist users on this difficult task the text provided within the text boxes is carefully selected from

important published research papers, while references are provided as links wherever needed for further reading.

The fundamental concept of CARS research is to include the context in the recommendation process to result from the 2D un-contextual RS: Users × Items → Ratings to the multidimensional CARS: Users × Items × Context → Ratings [2]. The latter represents a single complete recommendation process and is the main idea behind this CARS Context Modelling System. For each recommendation attempt, a RS must examine whether each item is suitable for a user *in a certain context*. This can be depicted through the question: *what is the rating a particular user would assign to a particular item under a certain context?* This rating score is what a recommender must calculate. Therefore, in order to examine whether an item is suitable for a user, the recommendation scheme must have exactly one *user*, exactly one *item* but one or more *context* entities, for each of which a *rating* score can be assigned.

The *context variable* concept contains the actual contextual information to be inserted into a context model [4]. Each context variable has a name and a value to describe both the context parameter and its particular value, e.g. "Temperature" is a context parameter and "high" is its value. Therefore, to define the context: "temperature can be high, medium or low", a total of three context variables will be needed. Via a weight property developers may denote a particular importance for their context variable. The "static" property refers to whether the context variable is static (cannot change dynamically, e.g. user's date of birth) or dynamic (can change, e.g. weather).

An *Application Context* is a context model for a particular RS, e.g. a movies RS. It is built by a CARS developer in order to model the context for this particular RS. An application context model contains all context variables that the developer will select, along with their values. Since each context variable has a name and a specific value ("Temperature: high"), a developer must select all variables with a particular name for the model to be accurate and complete (e.g. all of the following: "Temperature: high", "Temperature: medium", "Temperature: low").

Although an application context model is built by a CARS developer to model the context for a particular recommender, the system enables sharing and reuse of such models by supporting other developers that want to build similar models in using the same context model and enhance/update it as needed. The idea is: since all RS of a specific type/field (e.g. online movie recommenders) interact in similar context settings, why having one (often incorrect or incomplete) context model for each such recommender built by each developer, when we can have just one (correct and complete) context model for all recommenders. We argue that application context models of recommenders of the same or similar fields should be identical, or in any case similar to a great extent. The system suggests to developers to use pre-existing application context models of similar applications (if any) and build upon them, instead of building a model of their own. Eventually, only one application context model for each type of recommender system will exist in the system that should be able to satisfy all developers.

A *Context Instance* is a "screenshot" of the context during an event of interaction between the user and the item that is involved in the recommendation process. For

example, for a movie recommender, a context instance is the set of context variables that constitute the context during the event of a particular user (user = "Tom") watching a particular movie (item = "Rambo") at a particular time (we assume the first time that Tom watches Rambo). Such a context instance may have title: "Tom-Rambo C1" (C1 results from "Context1") and may be consisted of the context variables: the time of day, the day of the week, the IMDB ratings of the movie watched, whom did the user watch the movie with, etc. In a similar way, the context instance around the second time Tom watches Rambo will have a title "Tom-Rambo C2" and will again be consisted of a number of context variables. Another definition of the context instance is that it is the set of context variables with their corresponding values that constitute the context at the time a single recommendation is requested. In addition, a context variable may participate in a number of context instances, each of which is characterized by that context variable. For example, the context variable "time: morning" can participate in many context instances, all of which refer to morning time.

Based on the above, the context instances define all valid contextual information around a particular fact or event (in the example above around user Tom watching movie Rambo at a certain time). Ideally, a context instance should be automatically created using context sensing and retrieval during the occurrence of a fact/event and stored in the system. E.g. around the event of a user watching a movie the system should be automatically aware of: the time of day, the day of the week, the IMDB ratings of the movie watched, whom did the user watch the movie with, and any other context variables that could participate in the process. This is a very difficult process, in some way impossible to achieve with the current technology available (how is the system going to know whom did the user watch the movie with, unless the user states it?) and it is beyond the scope of this work. In this CARS Context Modelling System we provide the ability for CARS developers to create their own context instances for modelling purposes, in order to closely observe and study whether their application contexts are able to "catch" *any* context instance that may occur, and in that way *validate* their models. More on modelling validation follows in Sect. 5.

4 Building Context Models

In the main page in [4] an input form is provided in order for developers to add context variables. Context variables contain the actual contextual information that developers need to insert into their context model to populate it [11]. The set of all context variables currently available in the system constitute the *Generic Context Model*[1] (Fig. 1). The application context models, the context instance models and

[1]Due to limited space in the paper, not all context models are presented complete in figures; instead, we provide hyperlinks to the models on the online tool in footnotes for reference: http://www.cs.ucy.ac.cy/~mettour/phd/CARSContextModellingSystem/genericContextModel.php.

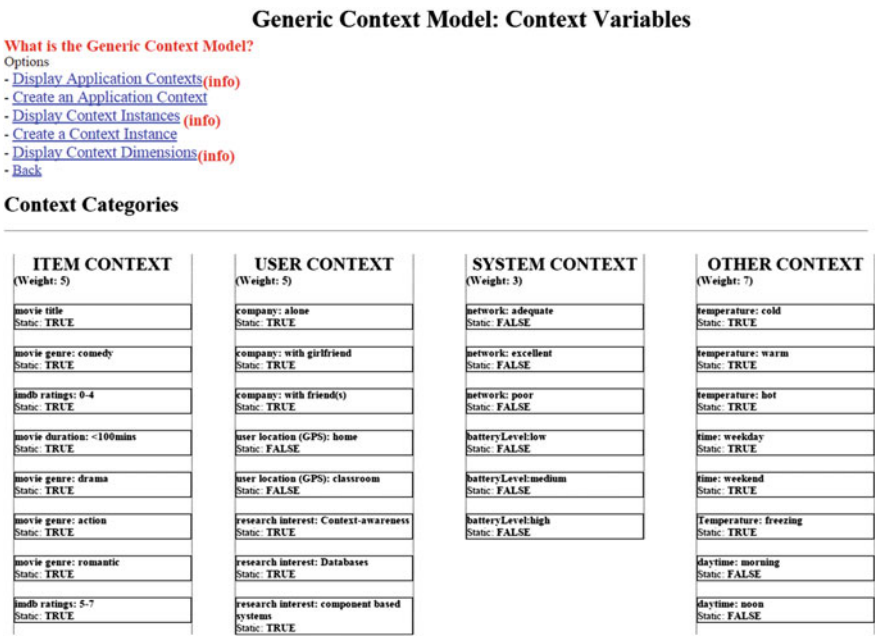

Fig. 1 The generic context model

the generic context model constitute the three types of context models supported by the system. The generic context model defines the basic contextual entities of RS, as well as their properties and associations in order for CARS developers to be able to extend it to construct application specific models for the needs of the application at hand. This generic context model simplifies the process of context model development and enables context uniformity, sharing and reuse.

All context models are based on the abstracted contextual modelling framework for CARS (refer to Fig. 2 in [11] and main page in [4]). These models constitute the "context" entity of this framework, as well as all the context related entities and relationships from that level downwards. In the generic context model[1] (Fig. 1), as in all context models, the four context categories are presented: "Item Context", "User Context", "System Context" and "Other Context" [4]. The rectangles depict the context variables (both name and value) and their parameters. Note that a context variable can belong to more than one context category; such decisions are made by the developer at the time of creation of the context variable or the context model. The "itemContext", "userContext", "systemContext" and "otherContext" constitute the four main context classes in the system and are meant to be perceived as the main context entities for any contextual model of CARS; any context information of any CARS should be able to be represented as a context property of one (or more) of the main context classes, as a context variable.

A simple and straightforward way is developed for CARS developers to build their own context models. They can either add a new context variable that is not

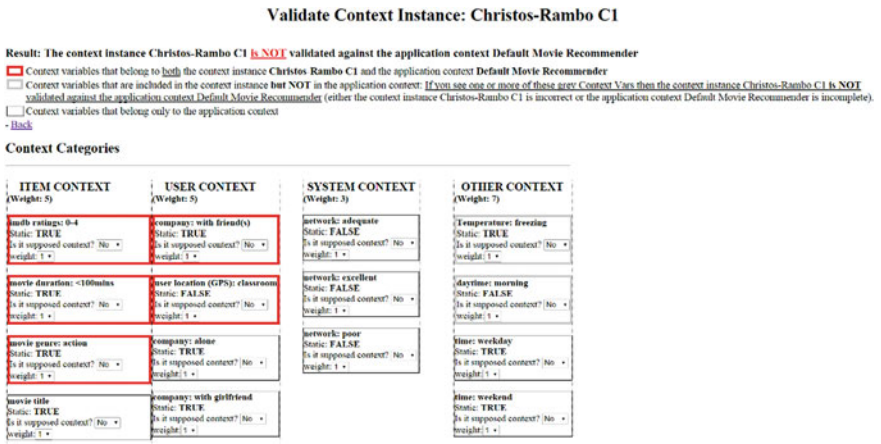

Fig. 2 The context instance "Christos-Rambo C1" is not validated against the application context default movie recommender

currently included in the system, or use the generic context model and simply select/unselect the context variables (already in the system) that are of interest to them by clicking on them. Before adding a new context variable in the system, developers are advised to first check whether the context variable they would like to add already exists in the system as part of the generic context model; if it does, developers are asked to use the existed variable in order to avoid redundant information in the system and confusion to other developers. In this way, the context variables created are universal among many applications and domains and hence they can be shared and reused in many context models of various CARS. For more information on the features and functionality offered by the developed online system, the reader is referred to the system [4].

5 Validating Application Contexts Through Context Instances

The system supports the validation of an application context model through one or more context instance models. As already stated in Sect. 3, an application context is a context model for a particular recommender application, e.g. a movies recommender system, while a context instance defines the context "screenshot" at the time a single recommendation is requested. It is evident that an application context model should be able to support any context instance related to the particular recommender; in other case, the application context model is incomplete. E.g. an application context model for a movie recommender should be able to support any movie recommender related context instance, such as "Tom-Rambo C1" (see Sect. 3) which can be consisted of a number of context variables such as the time of

day, the day of the week, the IMDB ratings, etc. This means that the application context model must include all context variables of the context instance model.

The possibility to validate an application context model through context instance models can be a useful tool for the CARS developer who has just created her application context model in the system (let's suppose an application context model for a movie recommender) and wants to ensure that her application context is able to properly model the context of movie RS (i.e. model any context instance of these recommenders). The context instance can be created preferably by a RS user who will reflect her experiences regarding the activity of watching movies in the context instance model. If this is not feasible, the context instance model can be created by another developer. Following, the CARS developer will be able to access the context instance and by clicking a button, validate this model against her application context model. The system then provides a justified answer whether the context instance model was validated against the application context. If yes, then the application context is also validated. If not, the system provides information as to why the model was not validated. A context instance may not be validated against an application context either because the context instance is incorrect or because the application context is incomplete. Figure 2 provides an example of the context instance "Christos-Rambo C1" not being validated against the application context "Default Movie Recommender" (you may use the online system [4] to view the actual coloured page).

6 Recommendation of Application Context Models and Model Comparison

During the creation of a new application context model by the CARS developer, the system is able to recommend the top N (currently N = 5) most similar application context models regarding the percentage of common context variables (refer to Sect. 8 and Fig. 4). This is very important for CARS developers who are in the process of creating their application context model and would like to be informed about similar context models in the system, as well as the level of similarity. As soon as a new application context is created, the system automatically provides recommendations.

Moreover, the system provides an easy way to compare two specific application context models. Through the usage of colours, the system depicts the common context variables of the two application contexts, as well as the context variables that belong only to one of the two application contexts. Besides common context variables, the system also provides statistics regarding the percentage that each application context participates in the other application context. For an example on the above you may refer to Sect. 8.

If two application context models have many common variables, then the system proposes a merge. It is certainly a situation where the CARS developers must

decide whether both application contexts are needed. This is especially interesting in the case where each application context concerns a different type of recommender system, e.g. a movie recommender and a book recommender, as context models of recommenders of different fields are very rarely similar.

7 Including Context Dimensions in the System

The basic concept of CARS research is to include the context in the recommendation process so that to result from the 2D un-contextual recommenders: *Users* × *Items* → *Ratings* to the multidimensional context-aware recommenders: *Users* × *Items* × *Context* → *Ratings* [2]. The term *Context* appears to be a single dimension itself, but in essence it represents the *Context Dimensions*, i.e. all the additional context-related dimensions that are being used in the recommendation process besides the user and the item (refer to Fig. 2 in [2] for an example of a three dimensional model for the recommendation space: User × Item × Time).

In our work, a context dimension can only be a context variable with all of its possible values. We use the following definition of the context dimension: *each individual context variable with a unique name and with all of its values can be perceived as a context dimension*. Therefore a context dimension name is a unique context variable name.

The system provides the "context dimensions view" option for each context model in the system: the generic model, an application context model or a context instance model. The "context dimensions view" is an alternative view of a model, besides the default "context variables view" of the system presented in Sect. 4. The "context dimensions view" is of particular importance for: i. observing the context dimensions of a model and ii. observing the context variables of a model with all of its values aggregated. Figure 3 presents the "context dimensions view" of the application context model Movie Recommender (refer to Sect. 8).

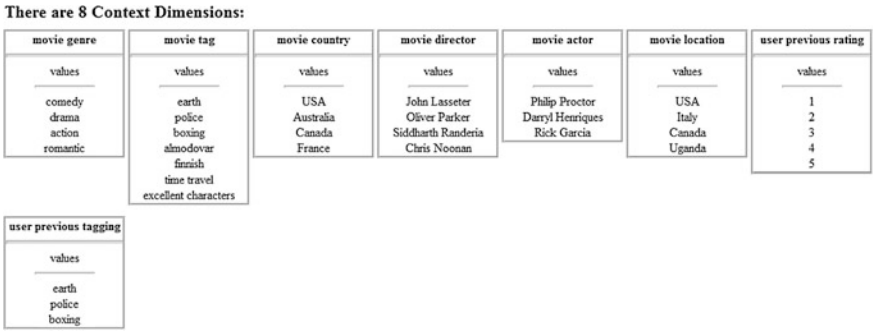

Fig. 3 Movie recommender application context model: "context dimensions view"

8 System Testing

The CARS Context Modelling System was tested by us in real settings by using a dataset aimed for usage with movie RS which was released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems—HetRec 2011 [8]. The dataset is an extension of MovieLens10 M dataset, which contains personal ratings and tags of users about movies. In the dataset, the movies are linked to Internet Movie Database (IMDb) and RottenTomatoes (RT) movie review systems. The dataset includes more than 2000 users, more than 10000 movies, more than 800000 ratings, as well as many related data such as movie genres, directors, actors, countries, locations and tags. It also includes information on user ratings on movies, as well as on user tagging of movies.

For system testing purposes we have played the role of a CARS developer who would like to model the context for building a movie recommender system based on this dataset. Similar datasets are frequently used by RS developers and researchers to develop their systems, both for commercial as well as research purposes. By using this dataset for building a context model in our system we can closely observe how a developer can be assisted in real settings.

We have created an application context model named Movie Recommender in the system based on this dataset and then, by using the available system functionality, we have compared this dataset with other application context models in the system to find similarities, as well as opine whether the context model built from the dataset was adequate to be used in a CARS for movies.

The first step was to (manually) extract the context attributes from the dataset. The context attributes we have identified are: “movie genre”, “movie director”, “movie actor”, “movie country”, “movie location”, “movie tag”, “user (previous) rating” and “user (previous) tagging”. The first 6 context attributes were assigned under the context category “item context”, while the final two under the context category “user context”. Each context attribute is inserted into the system as a context variable in the form: “name: value”. E.g. “movie country: France”. Since the amount of raw data included in the dataset is huge (10000 movies) at this point we have decided to insert only a sample of the data in the system’s database. This is adequate for our “proof of concept” type of testing. The context variables of the created application context model Movie Recommender can be seen via the online system,[2] while Fig. 3 shows the context dimensions of the same model (there are 34 context variables and 8 context dimensions in this model).

As we can observe from the online model[2] and Fig. 3, we were able to extract only 8 context dimensions from the dataset, populating the application context model with a sample of 34 context variables. If we had populated the model with all the information of the dataset, the context variables in the model would be hundreds

[2] http://www.cs.ucy.ac.cy/~mettour/phd/CARSContextModellingSystem/displayAppInstancesModel.php?appCont=Movie%20Recommender.

Fig. 4 Top 5 recommendations of similar application context models

more; however, the distinct context variable names (and hence the context dimensions) would still be only 8. Another important observation is that the application context model[2] includes information in only two of the four context categories. This is certainly a limitation of the model, since it does not utilize the context entirely.

After creating the application context model, we have been provided with system recommendations of similar application context models already in the system (created earlier by us, by colleagues of ours and by experts on context-aware recommenders). The system provides the top 5 most similar models to our model. The most similar application context model is one of another CARS for movies (named Default Movie Recommender[3]) with 58 % similarity, following other models of other recommenders that are similar to ours in percentages between 8–2 % (Fig. 4). The similarities of the two application context models can be seen via the online system[4] (Fig. 5). In red colour the common context variables are depicted. In green are the context variables that belong only to the Movie Recommender context model, while in blue are shown the context variables that are included only in the Default Movie Recommender context model (you may use the online system[4] to view the actual coloured page).

It is quite easy to observe that the context model for the Movie Recommender is inferior to the Default Movie Recommender application context model, since it only includes a percentage of the context attributes of the Default recommender in all context categories. In Item Context category there is 65 % of common context, in

[3]http://www.cs.ucy.ac.cy/~mettour/phd/CARSContextModellingSystem/displayAppInstancesModel.php?appCont=Default%20Movie%20Recommender.

[4]http://www.cs.ucy.ac.cy/~mettour/phd/CARSContextModellingSystem/compareApplicationContexts2.php?sentData=$Default%20Movie%20Recommender$Movie%20Recommender.

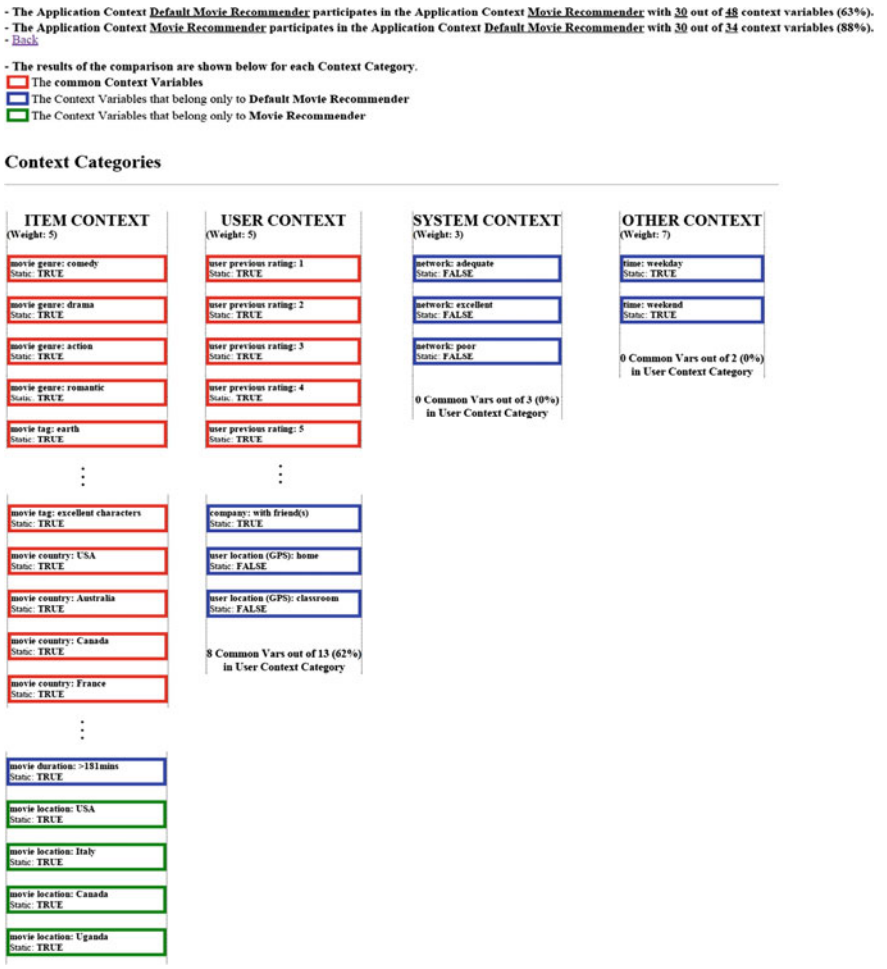

Fig. 5 Combined context model for the default movie recommender and the movie recommender

User Context category there is 62 % of commons context, while in the System Context and Other Context categories the common context is 0 %.

Switching to the "context dimensions view" (on the online system), we observe that the Movie Recommender context model includes the context dimension "movie location" consisted of 4 context variables which is not included in the Default Movie Recommender system. This is a deficiency of the Default model. However, there is a number of context dimensions that were included in the Default model but not in the Movie Recommender application context model: movie title, imdb ratings (the imdb ratings of a movie), movie duration, company (with whom the user watches the movie with), user location (GPS), network (describes the

network connectivity/bandwidth) and time (time of day of watching the movie). We can observe that, while our context model includes only 8 context dimensions, the Default model includes 14.

9 Conclusions and Future Work

The purpose of the CARS Context Modelling System presented in this work is to serve as a tool for CARS developers (and researchers), enabling them to efficiently, effectively and correctly select and use context attributes for building their own application models, allowing at the same time for sharing and reuse of context models and information among applications, regardless of the domain they belong to. We have presented the system concept and its functionality, as well as discussed additional important features such as the validation of application contexts through context instances, the comparison of application context models, the recommendation of similar application context models to the user and the inclusion of the context dimensions concept in the system (via the "context dimensions view" of context models).

The system testing we have conducted in real settings by using a real dataset [8] showed that the system was able to depict important limitations of the application context model created based on the dataset, in comparison with other pre-existing CARS application context models in the system (built by us, colleagues and experts).

Based on the above, we conclude that if CARS developers were to define the context for a movie RS based on this dataset (which is a valid dataset for movie RS used by many developers) and without using the CARS Context Modelling System, the result would be an inefficient and incomplete context model. Instead, by using the CARS Context Modelling System we were able to build our application context model with the aid of system tools, compare our model with other models, as well as be recommended with similar models that could assist us in developing and enhancing our application context model.

Currently, the system provides all context models as visual images and will support the extraction of the models in xml and txt formats. As future work, we aim to support CARS developers in incorporating the context models within their RS by providing appropriate tools. An idea is to extent the system by providing automatic transformation of the context models to code. The idea is to support CARS developers in attaching their context models to their recommendation methods via an easy and straightforward way that will also support context model sharing and reuse.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. (TOIS) **23**, 103–145 (2005)
2. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook, pp. 217–253 (2011)
3. Anand, S.S., Mobasher, B.: Contextual recommendation. WebMine LNAI **4737**, 142–160 (2007)
4. CARS Context Modelling System. http://www.cs.ucy.ac.cy/~mettour/phd/CARSContextModellingSystem/
5. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. **22**, 143–177 (2004)
6. Dourish, P.: What we talk about when we talk about context. Personal Ubiquitous Comput. **8** (1), 19–30 (2004)
7. Eclipse Modeling Framework Project (EMF). http://www.eclipse.org/modeling/emf/
8. hetrec2011-movielens-2k.: Dataset released in the framework of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011) at the 5th ACM Conference on Recommender Systems (RecSys 2011). http://ir.ii.uam.es/hetrec2011/datasets.html (2011)
9. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the tenth international conference on Information and knowledge management, pp. 247–254 (2000)
10. Mettouris, C., Achilleos, A.P., Papadopoulos, G.A.: a context modelling system and learning tool for context-aware recommender systems. In: Hernandez-Leo, D., Ley, T., Klamma, R., Harrer, A., (eds.) Scaling up Learning for Sustained Impact, LNCS, vol. 8095, pp. 619–620. Springer Berlin Heidelberg (2013)
11. Mettouris, C., Papadopoulos, G.A.: Contextual modelling in context-aware recommender systems: a generic approach. In: Haller, A., Huang, G., Huang, Z., Paik, H.-Y., Sheng, Q.Z. (eds.) WISE 2011 and 2012 Combined Workshops. LNCS, vol. 7652, pp. 41–52. Springer, Heidelberg (2013)