Explaining contextual recommendations: Interaction design study and prototype implementation

Joanna Misztal
Jagiellonian University
Cracow, Poland
joanna.misztal@uj.edu.pl

Bipin Indurkhya
Jagiellonian University
Cracow, Poland
bipin.indurkhya@uj.edu.pl

ABSTRACT

We describe an architecture for generating context-aware recommendations along with detailed textual explanations to support the user in the decision-making process. CARE (Context-Aware Recommender with Explanation) incorporates a hierarchical structure, in which independent modules embodying different aspects of the context cooperate together to generate recommendations for the user with accompanying rationales. We follow the Interaction Design principles to develop personas, goals and user scenarios, based on which a prototype system is developed. We present here two examples of its performance when processing movie-ratings data set with contextual information. We argue that our architecture is extensible in that more modules can be added as needed, and the approach can be applied to other domains as well.

Keywords

context-aware recommender system, recommendations explanations, interaction design

1. INTRODUCTION

An increasing number of available resources, and easy online access to diverse goods has resulted in data overload, making it difficult for many users to decide what items to select, which often slows down their decision-making process. A growing number of choices is leading to an emerging interest in the development of decision-support systems to help users in finding the most interesting or suitable items for their personal needs. Most of the research in this domain is focused on improving the accuracy and precision of recommendations. However, it is equally important to provide the user with some rationale for why a particular item is being recommended to them. Moreover, in some domains such as legal decision-making or moral and ethical reasoning, the justifications for recommendations are very crucial. Hence, the main focus of our work is to design a system that can explain why the user should select particular items.

As observed in [3], the user's preferences may be influenced by factors as diverse as time of the day, day of the week, the season or the weather at the moment, and so on. In our system, we incorporate different independent modules such that each module implements a particular approach to generating a recommendation based on a single contextual feature. This architecture allows generating a number of diverse recommendations, as each piece of contextual information is analyzed separately and the most approporiate items are recommended by choosing from among the various recommendations generated by different modules.

As the main focus of our research is to improve user's experience and understanding during the interaction with the system, we designed a Context-Aware Recommender with Explanation (CARE) system, following the Interaction Design paradigm [10]. Personas, user goals and scenarios are developed after interviewing potential recommendation-system users and a domain expert, based on which the prototype of the system is designed.

We motivate here our approach in the context of the current state of the art, summarize the interaction design process, and present examples of persona and scenarios. Then we describe the system architecture and present some results generated by our prototype implementation.

2. BACKGROUND

As defined in [20], the main goal of a recommender system is to support the user in a decision-making process by suggesting items that they might find interesting. Since information overload is a growing problem for Web users, because of an exponential increase in the amount of web content that is being generated, development of such tools has become a thriving research area in recent years. Consequently, several techniques have been developed for predicting users' preferences.

2.1 Content-based recommender systems

Content-based recommenders try to find items similar to what the user previously liked [17]. These work by identifying key features of the items highly rated by the user in the past, and by building a user-preference model from those characteristics [5].

A significant problem in many recommendation techniques is the *cold-start* problem, which occurs when a new user or a new item is presented to the system and there is not enough data to perform a reliable prediction - the user has not rated enough products to define his or her preferences, or an item has not been rated by a sufficient number of users.

Content-based recommenders deal well with situations when a new item is added to the system. It also maintains independence among the users as a particular user's ratings are sufficient to perform the recommendation process for that user. For our research, a major advantage of content-based recommenders is their transparency — the features that triggered the recommendation results may be listed along with the output.

However, content-based recommenders suffer from overspecialization: i.e. they recommend items similar to those seen in the past, preventing a serendipity of recommendations. Also, when a new user, who does not have a previous history with the system and so lacks any ratings, enters the system, the *cold-start* problem may occur.

Content-based analysis operates on vectors representing features of each object. Two basic techniques for filtering similar items are similarity calculation (such as cosine similarity) and distance measurement (such as Euclidean distance).

2.2 Collaborative filtering recommender systems

In collaborative filtering (CF), a user's preferences are predicted based on modelling other users behaviors [24, 16]. The basic idea behind this approach is that the rating of a user for a new item should be close to the ratings of users who have similar tastes.

This approach suffers from the data-sparsity and new-item problems. Another disadvantage is that collaborative-filtering recommenders mostly work as black-box systems, therefore they lack transparency and cannot explain why certain items are being recommended. However, this approach is proving to be an effective technique for making recommendations and is widely used in commercial applications. It has an advantage of being able to recommend items with unknown content. CF also supports serendipity in recommendations, for recommended items may differ significantly from the previous ones.

Common approaches to CF for recommendations use neighbourhood-based or model-based methods. Neighbourhood-based methods try to find the most similar items (item-based approaches) or most similar users (user-based approaches) when predicting the rating of an item for a particular user. Basic technique may incorporate correlation measures (such as Pearson's similarity) when comparing the vector of ratings for users or items [14]. Model-based methods work by finding the latent features that characterize the user's ratings, and build a predictive model of their preferences. Such methods may employ Matrix Factorization algorithms, Bayesian models, Support Vector Machines or other such techniques.

2.3 Context-aware recommender systems

As observed in [3], a person's preferences may be influenced by factors as diverse as time of the day, day of the week, the season, the weather at the moment, and so on. Context-aware recommender systems (CARS) try to model user's preferences considering changing contexts that may affect user's moods and tastes [4]. Contextual data may be collected explicitly by asking the user some questions, or implicitly from the environment (information such as time, day of week, season or location). Some information may also be statistically inferred from the other data (such as the companion or mood). In CARS, the input data from the standard recommendation approach in the form $< user, item, rating >$ is extended by an additional parameter of $context$. Some standard approaches to recommendations have been adapted to model the additional dimension of context. In [15], the authors present a *Tensor Factorization* model-based technique using N-dimensional tensor of User-Item-Context instead of the 2D User-Item matrix.

Standard approaches for CARS implementation incorporate contextual pre-filtering (items filtered by context before recommendation), post-filtering (context applied to recommendation results) and contextual modeling (context as a part of ratings prediction). Common approaches based on standard pre-filtering techniques represent item and user-splitting algorithms [28, 6]. In these methods, items (or users) in different contexts are treated as separate objects for the recommendation algorithm. Context-aware systems are known to increase the accuracy of recommendations [4]. However, they face the data sparsity problem, as the number of ratings is restricted to given context. As discussed in [7], the most efficient approach to context-splitting is *single split*, where objects are split considering a single contextual feature.

2.4 Other approaches

Some recommenders are implemented as knowledge-based systems [8], where the recommendation algorithm is performed by a set of constraints representing the knowledge about the domain. Such systems may be applicable to the domains for which historical data is not available, or when the user does not perform the action often enough, so there is little data to make a prediction (e.g. buying a car).

Another approach is to use demographic information about the users to predict their tastes based on their social group. Such recommendations may depend on user's age, gender, nationality, and so on.

2.5 Hybrid solutions

Each of the techniques mentioned above has some advantages as well as some drawbacks, and each may be effective for a certain domain or a certain type of problem [13, 9]. In order to build a more general recommender system, or to improve the quality of recommendations, *hybrid systems* combine diverse recommendation algorithms. There are different ways to combine outputs of various recommendation strategies, which are classified by Burke [9] as follows:

Weighted: the scores from several recommenders are weighted into one result.

Switching: the most appropriate technique is selected depending on the input data.

Mixed: outputs from diverse algorithms are presented simultaneously.

Feature combination: features of different algorithms are combined into a new feature.

Cascade: the recommendation is performed hierarchically and the outputs are refined by the subsequent recommenders.

Feature augmentation: output from one system is the input to the following one.

Meta-level: the model created by one system is used by another.

2.6 Multi-agent approaches

Hybrid recommender systems are often implemented based

on diverse multi-agent system (MAS) architectures. Distributed approaches to recommendations have been previously studied in [27], where a collaborative recommendation algorithm is implemented using cloud computing. Sabater, Singh and Vidal [22] proposed a protocol in which a group of selfish agents can decide how to share their recommendations with the others. In [26], the authors introduce *recommender agents* to enable the user's interaction with the system and to combine the outputs of the recommendation algorithms with other techniques such as other users bookmarks and tags.

Another example of MAS architecture that may be used for implementing a recommender system is the *blackboard architecture.* This architecture may be visualized by the metaphor [11] of a group of independent experts with diverse knowledge who are sharing a common workspace (the blackboard). They work on the solution together and each of them adds some contribution to the blackboard, whenever possible, until the problem is solved.

The blackboard model provides an efficient platform for problems that require many diverse sources of knowledge. It allows a range of different *experts* represented as diverse computational agents, and provides an integration framework for them. It seems a promising platform for recommendation tasks, and has already been incorporated in [12, 21].

3. OVERVIEW OF CARE SYSTEM

3.1 Architecture

CARE (Context-Aware Recommender with Explanation) is built as a hybrid architecture that adapts a *mixed* approach to recommendations, and incorporates some features of the multi-agent blackboard architecture. We implemented each module as an independent subsystem that uses some particular approach to generating a recommendation by incorporating a particular contextual feature. As some of those factors may be non-deterministic, we present the final result as an array of alternate choices and allow the user to choose from the recommendations generated by analyzing diverse contextual factors. Hence the diversity of final recommendation outputs is ensured by presenting the analysis from multiple points of view. This approach also incorporates serendipity, and gives the users a choice of possible actions. The users actions can be noted and used for ordering future recommendations.

Our solution also embodies some aspects of the *feature augmentation* approach — we perform the recommendations hierarchically on different levels of abstraction. We introduce some inter-level recommenders that are responsible for defining the features of items that are most liked by the users in a given context. Their outputs are used to filter the data with identified characteristics, which is then sent as an input to the higher layer of recommendations.

3.2 Evaluation

Most of the solutions for automatic recommendations are focused on development of techniques improving the overall performance and accuracy of ratings prediction. Accordingly, most popular evaluation approaches incorporate precision metrics for the estimations. However, there are other factors that impact the effectiveness of recommendations and influence the user experience.

A major limitation of the existing recommendation systems is overspecialization and a lack of diversity in recommender outputs [29, 19]. As described in [23], during the challenge on Context-Aware Movie Recommendation (CAMRa 2010), competing systems were evaluated according to diverse factors divided into two groups. The first set of criteria consisted of precision metrics while the other set contained the following *Subjective Evaluation* Criteria: Context, Contextualization of recommendations, Extensibility, Serendipity, Creativity, Scalability, Sparsity, Domain dependence, and Adoptability

In our research, we aim to address some of these subjective criteria to improve user experience, and also develop an architecture that is easily adaptable to other domains. We use contextual filtering to model user preferences. Our architecture enables one to implement flexible and generic recommender systems that can easily be extended with new independent modules in a hierarchical structure. Our approach promotes diversity and serendipity among the recommendation outputs as each module processes information from a different point of view.

3.3 Explaining recommendations

We mentioned above that aspects such as user satisfaction play an important role in the evaluation of a recommender system. As noted by [25], a limitation of many recommenders is that they work as black-box systems and do not provide the users with any reasons for providing a particular recommendation. Some of the commercial systems are striving to overcome this limitation by producing a rationale accompanying each recommendation. A number of diverse styles have emerged to provide this rationale [25]: *Case-Based* (*... because you highly rated Item A...*, used in Netflix [2]), *Collaborative* (Customers who bought this Item were also interested in... used by Amazon), *Content-Based* (We are playing this music because it has a slow tempo by Pandora [1]). In [26], the authors use information visualization techniques to improve interaction with their recommender system. Such system transparency not only increases the user satisfaction, but also helps the users in making easier and faster decision in selecting an item.

In CARE, each module is provided with an explanation-generating function, which produces a description of the features that determined the recommendation. The style of this message is dependent on the module's implementation. The final explanation may combine different styles of messages produced independently by separate modules. We incorporate modules that process information on different levels of abstraction, hence the final description contains rationales at multiple granularity levels. Our goal is to generate a rationale explaining to the user why she or he should find certain items interesting (contextual reason, e.g. *because it is rainy* and what features make this particular item a relevant choice (e.g. *because you like rock music when it is rainy*).

4. GOAL-DIRECTED DESIGN

In designing the CARE system, we follow the principles of Interaction Design [10], according to which the design of a system is developed iteratively through continuous interaction with the user. In the subsequent sections, we summarize the conclusions from the interviews with three users and a movie-domain expert.

4.1 Domain expert's opinion

Following the Interaction Design paradigm, we consulted a film analysis academic to find out what factors may influence the popularity of a movie among the users.

The expert noted that the genre is not the only feature that the users consider when deciding which movie to watch. Other factors which may determine their preferences are narrative description, its tempo, atmosphere and tension.

Moreover, the users often want to watch the same kind of movies that they have already watched. Thus, they select well-known names, plots or recognizable brands. Such a *brand* may be defined by the director, movie star or award such as Oscar or some Film Festival.

Considering all these factors, our system design should embody modules that analyze relevant movie features such as genre, cast, director, awards won as well as information about the atmosphere of the movie. The prototype implementation incorporates the genre filtering modules, and we plan to extend it with modules that will analyze other factors as well.

Another major factor that influences a user's choice is the current trend or fashion. There are some *must-see* movies that many users desire to watch. Moreover, some people rely on the public opinion more than on their own impressions. In our design, the public opinion is modelled by a collaborative filtering recommendation algorithm.

4.2 Defining User Goals

We interviewed three potential users to determine their expectations from a recommender system. The volunteers were technical faculty graduates who are familiar with using recommender systems to find items of their interest. Each of them was interviewed separately, in their natural environment. They were asked to describe their experiences in one of the three domains of recommendations: books, movies and music. First, each person was asked to describe his or her general preferences in the given area and if they could identify some factors that may influence it. Then they were asked to describe some particular situations in which they use recommendations, considering the context details. The final question was about the expected recommendation output in these situations. We are planning to extend this research by incorporating interviews with a broader group of users with more diverse backgrounds.

It was observed that every person has some general tastes, but particular preferences change according to the time and the mood. Thus, the system should consider some contextual information in generating the recommendations. However, some of these factors, such as the user's mood, who they are with, and so on, may be difficult to predict. Hence the approach we chose is to give the user a choice of possible actions considering different contextual or affective states so that the user can decide what she or he needs at the moment.

Finally, we found that the users like to know why any particular item is recommended to them. Hence the system should aggregate information from different levels of abstraction, and present it to the user in an intuitively understandable way. We plan to present a rationale for each recommendation as an accompanying text message.

4.3 User scenarios

Persona: Mark

Goals: getting movies recommendations; finding uncommon yet interesting movies when alone; finding lights comedies to watch with his girlfriend

Scenario 1

It is a cold winter Friday and Mark and his girlfriend want to spend the evening with a light movie and a glass of wine. Mark opens CARE and a message pops out:

Hi Mark! Finally, it's the weekend! It's freezing, isn't it? Are you dreaming of little holidays? What about Woody Allen's "Vicky Cristina Barcelona" to warm you up a little? I know you like this director. Or maybe you had a tough week and feel like watching something to cheer you up with a bit of dark humor, like "Grand Budapest Hotel"?

Scenario 2

On Monday, Mark's girlfriend is off for a ladies night with her friends so finally he can choose a movie on his own. CARE greets him:

Hi Mark! Maybe something positive for the new week? How about "Intouchables"? Or maybe you're fed up with the city life in Krakow and want to watch the story of a man in the heart of nature, like "Into the wild"? You like non-fiction movies!

5. SYSTEM ARCHITECTURE OF CARE

General architecture of the CARE system is presented in Figure 1. Modules in our prototype system work on different levels of abstraction.

First group of modules perform *contextual features filtering* based on the input with current context, user information and ratings history with context. The goal of this processing phase is to determine the most relevant item features for a given context. Each component on this level analyzes the information about a single contextual information. To address the problem of sparsity, we also incorporate a module that considers all users' ratings without any contextual filtering. The output of each filter is a list of items characterized by the identified features. This architecture is extensible with different types of filters that analyze other aspects of recommendations (such as demographic data). In this paper we focus only on the contextual information processing.

In the next stage of recommendation process, we incorporate recommender algorithms that select items that should be most liked by the user, considering each of the item groups received from the former stage as a separate recommendation problem. The modules on this stage may represent diverse recommendation techniques and algorithms, however our prototype implementation includes collaborative filtering algorithms. The result of recommendation is the best choice of items for each set of items.

Each component generates a short description of its results and the reason of recommendation. The messages are finally composed by the *explanation templates* module and presented to the user along with the recommended items.

The hierarchical structure of the system and the inter-level filtering of item features enables a more thorough explanation of the process in generated outputs. Hence the output does not only provide the user with the information *Item A was recommended because it is summer,* but also emphasizes the feature that was crucial for this choice (*Item A was recommended because you seem to like this type of items during summer.*).

The system is being developed using Django, a Web framework for Python. The system interface will be provided as a web application, however at present the system output is

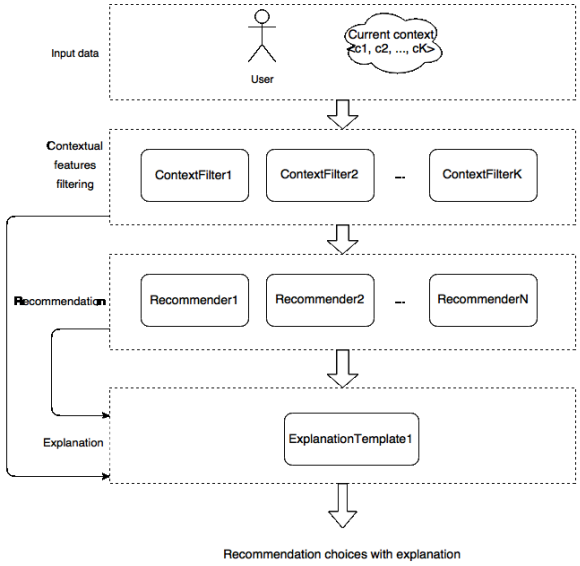

Figure 1: General architecture of the CARE system.

in plain textual form as presented in the results section.

6. EXPERIMENT

We present phases of the recommendation algorithm along with an illustrative example of recommendations for user John.

6.1 Testing data

CARE system architecture is applicable to many recommendation domains where the context may influence user preferences. Possible domains of application include books, music or movies as well as restaurants recommendations as user's choice may depend on aspects such as changing weather or time. Here we present the results from testing our prototype on the LDOS-CoMoDa dataset [18] that contains movie ratings along with contextual information, and user and item characteristics. Contextual data contain information about the season, type of day (weekend, working day or holiday), time of day (morning, afternoon, evening), companion, and so on.

The dataset also contains information about the *mood* of the movie and it could be interesting to consider this data as well. However, the dataset only provides the *dominant* and *end* emotional values, without any information about the user's mood *before* watching the movie. Since we treat contextual factors as facts known at the moment of recommendation (as the initial data), we cannot make a recommendation considering user's mood after or during watching the movie. In future work, we plan to add a feature to query user's mood at the moment of recommendation, and consider this information as a contextual parameter.

Table 2 contains a part of John's ratings for the analyzed example along with contextual information.

6.2 Recommendation process

We present below a brief description of the different stages of our algorithm along with illustrative examples. The main steps of algorithm are listed below and described in the subsequent sections:

1. Initialization with contextual input data.
2. Contextual type-splitting - identifying significant contextual factors and relevant data types.
3. Collaborative filtering items recommendation for each of the types defined in 2.
4. Explanation generation for each of the recommended types and a corresponding contextual factor.

6.2.1 *Input data*

Input for the recommendation is the user data and the information about the context in which the recommendation takes place. This data may contain information explicitly provided by the user (such as whether they are alone or with a companion) or implicit information extracted automatically from the date and location data (such as day of week, time of the day, weather, and so on).

Example:

Context:
Season: Summer
Day type: working day
Time: evening
Weather: sunny
Companion: alone

6.2.2 *Contextual type-splitting*

For the context-aware recommendation process, we introduced *contextual type-splitting* algorithm which is an adaptation of the standard *contextual item-splitting* approach. We incorporated an additional abstraction level and treated the item features as the recommendation objectives. Table 2 illustrates the difference between approaches.

- Contextual item-splitting

  In the first phase of the basic algorithm, each item is associated with the most relevant contextual feature that diversifies its ratings. Then the ratings for this item are split according to this division. The mean rating values for each item are compared for the situation where particular circumstance occurs and otherwise.

  For example, we could compare ratings for a particular movie that were given during the weekend with ratings from all other days. If they are significantly different, we can infer that the user preferences for this item are influenced by the day of the week.

- Contextual type-splitting

  In our approach, we perform analogical computations, but instead of comparing the contexts for each movie, we analyze each context for a group of movies with a common feature (such as a genre). As a result we can, for example, find out that the user prefers to watch horror movies during the weekends than on other days. This step may be generalized to recommend items grouped by other features, such as the director, country of origin, etc.

movie id	genre	rating	daytype	time	weather	season	companion
23	action movie	1	working day	evening	sunny	autumn	alone
160	action movie	2	working day	evening	sunny	autumn	alone
2755	action movie	4	working day	evening	rainy	winter	alone
3898	action movie	5	weekend	night	cloudy	spring	with family
3942	action movie	3	working day	evening	sunny	spring	alone
4020	action movie	5	weekend	night	rainy	summer	alone
3992	action movie	5	working day	afternoon	sunny	summer	with family
3962	animated movie	4	weekend	evening	rainy	spring	alone
65	comedy	3	working day	evening	sunny	autumn	alone
101	comedy	2	working day	afternoon	cloudy	autumn	alone
4025	comedy	5	working day	evening	sunny	summer	alone
4054	comedy	5	holiday	night	cloudy	summer	alone
30	crime movie	4	weekend	evening	rainy	autumn	alone
227	crime movie	2	working day	night	cloudy	autumn	alone
3715	crime movie	1	working day	evening	cloudy	winter	alone
248	crime movie	5	weekend	night	sunny	winter	with family
149	drama	5	weekend	evening	sunny	autumn	alone
61	drama	2	weekend	afternoon	sunny	autumn	alone
239	drama	3	holiday	night	rainy	autumn	in public
242	drama	4	holiday	evening	cloudy	autumn	alone

Table 1: User's ratings with contextual information.

Exemplary ratings with a context				
User	Item	Item type	Rating	Context
U1	I1	T1	1	C1
U1	I3	T1	3	C1
U1	I2	T2	5	C2
U1	I3	T1	5	C2
U1	I2	T2	5	C1

Contextual items splitting		
User	Item	Rating
U1	$I3_{C1}$	3
U1	$I3_{C2}$	5

Contextual types splitting		
User	Item type	Rating
U1	$T1_{C1}$	2
U1	$T1_{C2}$	5

Table 2: Contextual splitting - basic approach and proposed modification.

This approach allows us to give more transparent and intuitive recommendations for the user by presenting the features that lead to the final recommendation. It also addresses a major drawback of the context pre-filtering approaches, namely the data sparsity after applying the filter. As the number of recommendations for a particular type of movies is significantly higher than for each movie separately, we expect the results to be more accurate. Reducing number of comparisons to groups of items also contributes to decreasing complexity of computations and speeds up the recommendation process. In subsequent steps, the recommendations are performed for selected groups only.

We verify the significance of each contextual feature using the two-tailed Student's t-test, assuming the p-value threshold of 10%. Additionally, we consider the significance of only those features where the mean rating is higher when a particular context occurs. The t-test is a basic approach (as presented in [28]), however its use is limited for normally distributed data. In other cases, it may be replaced by other statistical methods such as Wilcoxon signed-rank test.

The features significance testing in a context is performed as follows:

- For each type t_i of items (eg. for each of the genres) we compare the mean rating for items of this type when each of the input contextual circumstances c_i occurs and otherwise (eg. ratings for comedies in summer and other seasons).
  $significance(c_i, t_j) = ttest(Ratings_{t_j|c_i}), Ratings_{t_j|\neg c_i})$
- If the statistical test for both groups of ratings split by the contextual feature c_i indicates a significant difference in mean ratings, and the mean rating for type when c_i occurs ($Ratings_{t_j|c_i}$) is higher then otherwise ($Ratings_{t_j|\neg c_i}$), we conclude that the type t_i is a relevant recommendation in given context c_i.

Example:

Contexts significance testing:
Input context: summer, working day, evening, sunny, alone
Ratings in summer vs other seasons: mean ratings for comedies are significantly higher during summer.
Ratings on working days vs other days: no relation found.
Ratings in the evening vs other time: mean ratings for comedies are significantly higher in the evening.
Ratings on sunny days vs other weather: no relation found.
Ratings for movies watched alone and other companion: no relation found.

6.2.3 *Types pre-filtering*

After identifying the types of movies that are most relevant for a given context, we filter the set of ratings for each type separately. For example, if we consider a recommendation for Saturday morning and we find out that horror movies are the most preferred genre during the weekend, and comedies get the highest ratings in the mornings, we

first consider recommendations for horror movies and then for comedies separately.

We also calculate general recommendations considering the user preferences of all the items, without any pre-filtering. This addresses the sparsity problem for context recommendations and deals with the situation when no relevant contextual information is provided.

6.2.4 Items recommendation

After filtering the items by their types, we perform a standard collaborative-filtering recommendation algorithm to find the most suitable choices considering a particular user's taste. We calculate the similarity between users using Pearson's correlation measure. Then we consider the ratings of the most similar users with the K-Nearest-Neighbours algorithm. In further research we plan to address the problem of scalability by users clustering.

For each of the identified categories we perform a separate recommendation process, hence the final output contains the recommendation results from diverse perspectives. For the current implementation, we incorporated the standard user-based CF algorithm. However the system may be easily extend by other modules performing different recommendation algorithms since the calculations are performed independently.

Example:

Performing user-based collaborative filtering algorithm to find expected highest-rated movies for each category and general user's preferences without any context:
crime story: "The Usual Suspects"
action movie: "Pirates of the Caribbean: At World's End"
all movies: "Le Concert"

6.2.5 Explanations generation

A major goal of our research is to develop a recommendation system that can present the recommendations along with accompanying explanations. Hence, we incorporated modules responsible for generating textual messages to give rationales for recommendations. Each sentence of the accompanying message consists of the following information: item type; recommended item; context.

The message is generated in the form of a textual template. Future improvements of CARE will consider increasing the serendipity aspect of the recommendations by generating more advanced and surprising commentaries for the outputs.

The messages generated by all the modules that analyzed the situation from different perspectives are aggregated in one template and presented to the user as a message.

Example:

Producing a textual explanation containing descriptions from all former steps of algorithm:

Hi John! You might like a comedy "Le Concert" as it is something in your taste. You might like a drama like "Shutter Island" because it is evening. Maybe you feel like watching a comedy like "Intouchables" because it is summer?

7. CONCLUSIONS AND FUTURE WORK

We proposed an architecture to generate context-aware recommendations along with accompanying rationales to help the user choose the most interesting item. In CARE (Context-Aware Recommender with Explanation), the recommendation process is performed hierarchically, and with transparency at each abstraction level so as to produce detailed explanations for the suggested choices. Our approach promotes a diversity of recommendation results since each piece of contextual information is analyzed separately, and the most appropriate items are recommended with a rationale accompanying each suggestion.

Our architecture enables one to implement a flexible and generic recommender systems that can easily be extended with new independent modules in a hierarchical structure. In the current stage of our research, we have tested the performance of the CARE prototype on a movie-ratings dataset. Following the suggestions of a domain expert, we plan to extend the system with modules that incorporate other diverse movie features such as the director, cast, atmosphere and so on. We also plan to follow the Interaction Design principles during the evaluation of our system and will perform user testing with a working system. In future work, we will also address the evaluation of results quality with standard methods such as RMSE or nDCG.

Our approach is applicable to other domains as well. Currently we are working on adapting this architecture for supporting legal decision making, and moral and ethical reasoning.

8. REFERENCES

[1] Pandora. http://www.pandora.com, 2006.

[2] Netflix dataset. http://www.netflixprize.com/, 2009.

[3] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM.

[4] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. 2011.

[5] M. Balabanović and Y. Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, Mar. 1997.

[6] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 245–248, New York, NY, USA, 2009. ACM.

[7] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. 24(1-2):7–34, 2014.

[8] D. Bridge, M. H. G oker, L. McGinty, and B. Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20:315–320, 9 2005.

[9] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[10] A. Cooper, R. Reimann, and D. Cronin. *About Face: The Essentials of Interaction Design*. John Wiley & Sons, Inc., New York, NY, USA, 2014.

[11] D. D. Corkill. Blackboard systems. *AI Expert*, 6, 1991.

[12] A. H. Dong, D. Shan, Z. Ruan, L. Zhou, and F. Zuo. The design and implementation of an intelligent apparel recommend expert system.

[13] B. S. Francesco Ricci, Lior Rokach. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.

[14] G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2619–2625. AAAI Press, 2013.

[15] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[16] Y. Koren and R. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer US, 2011.

[17] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer US, 2011.

[18] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. In G. Adomavicius, editor, *Proceedings of the 4th Workshop on Context-Aware Recommender Systems in conjunction with the 6th ACM Conference on Recommender Systems (RecSys 2012)*, volume 889, 2012.

[19] L. Qin and X. Zhu. Promoting diversity in recommendation by entropy regularizer. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2698–2704. AAAI Press, 2013.

[20] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[21] A. Ruiz-Iniesta, G. Jiménez-Díaz, M. Gómez-Albarrán, et al. A framework for the rapid prototyping of knowledgebased recommender systems in the learning domain. *Journal of Research and Practice in Information Technology*, 44(2):167, 2012.

[22] J. Sabater, M. Singh, and J. M. Vidal. A Protocol for a Distributed Recommender System, 2005.

[23] A. Said, S. Berkovsky, and E. W. De Luca. Introduction to special section on camra2010: Movie recommendation in context. *ACM Trans. Intell. Syst. Technol.*, 4(1):13:1–13:9, Feb. 2013.

[24] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer Berlin Heidelberg, 2007.

[25] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 479–510. Springer US, 2011.

[26] K. Verbert, D. Parra-Santander, P. Brusilovsky, and E. Duval. Visualizing recommendations to support exploration, transparency and controllability. In *Proceedings of the 2013 international conference on Intelligent user interfaces - IUI '13*, page 351. ACM Press, 2013.

[27] Y. ZHANG, H. Liu, and S. Li. A Distributed Collaborative Filtering Recommendation Mechanism for Mobile Commerce Based on Cloud Computing. 2011.

[28] Y. Zheng, B. Mobasher, and R. D. Burke. The role of emotions in context-aware recommendation. In L. Chen, M. de Gemmis, A. Felfernig, P. Lops, F. Ricci, G. Semeraro, and M. C. Willemsen, editors, *Decisions@RecSys*, volume 1050 of *CEUR Workshop Proceedings*, pages 21–28. CEUR-WS.org, 2013.

[29] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.