Incorporating Context Into Recommender Systems: An Empirical Comparison Of Context-Based Approaches

Umberto Panniello and Michele Gorgoglione

Politecnico di Bari (Italy), Viale Japigia 182

Tel. +39 080 5962765

Fax: +39 080 5962766

u.panniello@poliba.it, m.gorgoglione@poliba.it

Abstract — Recently, there has been growing interest in recommender systems (RSs) and particularly in context-aware RSs. Methods for generating context-aware recommendations were classified into the pre-filtering, post-filtering and contextual modeling approaches. This paper focuses on comparing the pre-filtering, the post-filtering, the contextual modeling and the un-contextual approaches and on identifying which method dominates the others and under which circumstances. Although some of these methods have been studied independently, no prior research compared the relative performance to determine which of them is better. This paper proposes an effective method of comparing the three methods to incorporate context and selecting the best alternatives. As a result, it provides analysts with a practical suggestion on how to pick a good approach in an effective manner to improve the performance of a context-aware recommender system.

Keywords: Recommender systems, context-aware, collaborative filtering, algorithms.

RSs: recommender systems; CARS: context-aware recommender systems; 2D: 2-dimensional; PreF: contextual pre-filtering; PoF: contextual post-filtering; CM: contextual modeling; CTRs: click-through rates; EPF: exact pre-filtering; CF: collaborative filtering; MAE: mean absolute error; RMSE: root mean square error.

1 Introduction

Most of the traditional recommender systems provide recommendations of items to users and vice versa and do not take into consideration the circumstances and other contextual information when recommendations take place. For example, when an online travel agency recommends a vacation package, it is important to know when the person plans to go on vacation. Recently, some companies started taking into account the contextual information. For example, Sourcetone

interactive radio (www.sourcetone.com) asks the listener to specify her mood (the context) before recommending a song. In academia, several studies, such as [1], demonstrated that context induces important changes in a customer purchasing behavior. Experimental research on customer modeling suggests that including context in a customer behavior model improves the ability to predict her behavior in some cases because it allows the identification of more homogeneous patterns in the data describing the purchasing history of a customer [2]. This observation is consistent with the view maintained by marketing researchers that the purchasing process is contingent upon the context in which the transaction takes place since the same customer can adopt different decision strategies and prefer different products or brands depending on the context [3, 4]. According to [5], "consumers vary in their decision-making rules because of the usage situation, the use of the good or service (for family, for gift, for self) and purchase situation (catalog sale, in-store shelf selection, and sales person aided purchase)". Therefore, the accuracy of predicting consumer preferences should depend on the degree to which we have incorporated the relevant contextual in-formation.

The usage of contextual information in context-aware recommender systems (CARS) can be broadly classified into two groups: (1) recommendation via context-driven querying and search, and (2) recommendation via contextual preference elicitation and estimation. The approach based on the context-driven querying and search has been used by several mobile and tourist recommender systems, e.g., [6, 7], that typically use contextual information to query or search a certain repository of resources (e.g., restaurants) and present the best matching resources (e.g., nearby restaurants that are currently open) to the user.

The other approach to CARS, that we follow in this paper, is based on contextual preference elicitation and estimation, e.g., [8, 9, 10, 11, 12], including elicitation and estimation of ratings of various items provided by different users. This approach can be traced back to the work in [9] and [13]. In [13], Herlocker and Konstan hypothesized that the inclusion of knowledge about the user's task into the recommendation algorithm in certain applications can lead to better recommendations. In [9], Adomavicius and Tuzhilin described a way to incorporate the contextual information into recommender systems by using a multidimensional approach in which the traditional 2-dimensional (2D) User/Item

paradigm was extended to support additional contextual dimensions, such as Time, Location and Company. Since then, several contextual preference elicitation and estimation approaches to CARS have been proposed, all of them emphasizing the need to model and learn user's context-sensitive preferences. Many of these methods are reviewed in [14] and [10].

Once the context-sensitive preferences of users are learned, recommendations are generated by either adapting the existing collaborative filtering, content-based, or hybrid recommendation methods to context-aware recommendation settings or by developing novel intelligent data analysis techniques from data mining and machine learning. Furthermore, several scholars [8, 10, 11, 12] have shown that adding contextual information helps to improve estimations of unknown ratings in this approach. For example, [8] and [9] described a way to include the contextual information by using a multidimensional approach, as it was mentioned above. Also, it was shown in [8] that the multidimensional contextual information does matter and can lead to better recommendations in comparison to the traditional un-contextual 2D recommender systems. Moreover, [11] presents another contextual recommender system based on some of the machine learning techniques, and experimentally shows that the context-based approach significantly outperforms the corresponding un-contextual one in terms of the accuracy and user satisfaction with recommendations.

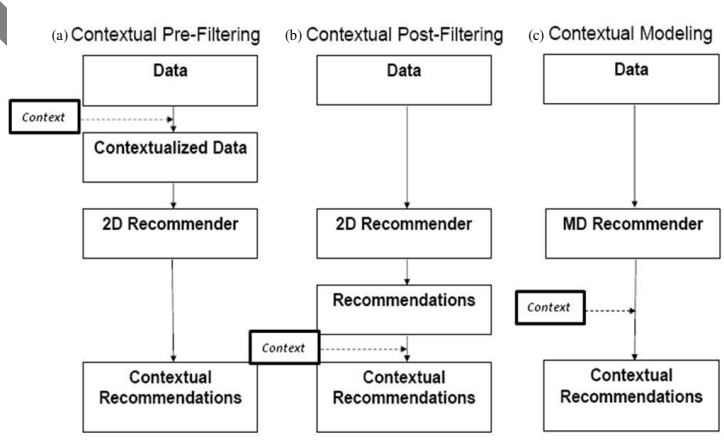

Fig. 1 How to use context in the recommendation process

In [14] (and also shown in Fig. 1), the following three different algorithmic paradigms for incorporating contextual information into the recommendation process, that starts with "Data" on users, items, ratings and contextual information

("Context") and results in generating context-specific recommendations, are presented:

1. Contextual pre-filtering (PreF): contextual information is used to filter out irrelevant ratings before they are used for computing recommendations with standard (2D) methods.
2. Contextual post-filtering (PoF): contextual information is used after the classical (2D) recommendation methods are applied to the standard recommendation data.
3. Contextual modeling (CM): contextual information is used inside the recommendation-generating algorithms.

The work presented in [14] helped researchers to understand different aspects of using the contextual information in the recommendation process. However, [14] did not examine which of these methods are more effective for providing contextual recommendations, and therefore, left this important topic un-addressed and without any prescriptive recommendations for which method to use and under which circumstances.

In this paper we empirically compare the three contextual approaches, i.e., pre-filtering, post-filtering and contextual modeling among themselves to determine which one is better, and also with the un-contextual approach when contextual information is ignored and not used in the recommendation methods. We also present several types of contextual post-filtering and contextual modeling methods and evaluate their performance in order to identify the best-performing methods in these two categories. We show that one particular post-filtering method dominates other approaches. However, this dominance comes with a certain complexity "cost" that prevents that method to be a clear winner in all the practical settings. Therefore, we propose an alternative approach to selecting the best-performing method that involves the comparison of the pre-filtering with the un-contextual methods and selecting the specific approach to including context (pre-filtering, post-filtering or contextual modeling) depending on the outcomes of this comparison. We argue that this approach constitutes a practical alternative method of including context in a recommendation process and advocate using it in the "real-world" settings.

However, before describing all these comparisons, we first provide some background information on CARS in the next section.

2 Background information on CARS

Traditionally, recommender systems (RSs) deal with two types of entities: users (e.g., customers) and items (e.g., products or Web pages). In the *rating-based* RSs, users rate the items that they have seen in the past, thus, specifying the utility of these items, i.e., how much the users liked the items. Alternatively, in the *transaction-based* RSs, the utility of an item for a user is not measured with a user's rating but is specified either by a Boolean variable indicating if the user bought a particular item or not, or with the purchasing frequency of an item, or with the click-through rates (CTRs) of various Web objects (URLs, ads, etc.) [15, 16, 17]. In this paper, we follow this approach and measure the utility of product *j* for user *i* with the purchasing frequency x_{ij} specifying how often user *i* purchased product *j*.

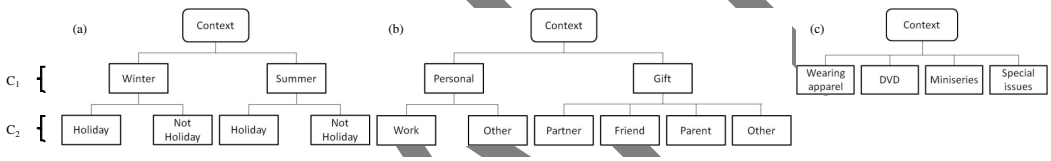

Fig. 2 Hierarchical structure of the contextual attributes (a) TimeOfTheYear, (b) IntentOfPurchase and (c) Store

Unlike the traditional two-dimensional (2D) recommender systems that try to estimate unknown ratings in the $Users \times Items$ matrix, where *Users* and *Items* are the sets of users and items respectively, context-aware recommender systems (CARS) also take into account contextual information. Therefore, CARS require specification of the following rating or utility function[1]:

$$R = Users \times Items \times Context \rightarrow Value \quad (1)$$

where *Context* is a set of contextual attributes C each having a hierarchical structure defined by a set of q atomic attributes, i.e., $C = (C_1,\ldots, C_q)$ [14]. Further, the values taken by attribute C_q define finer (more granular) levels, while C_1 coarser (less granular) levels of contextual knowledge [18]. For example, Fig. 2

[1] For simplicity, we will use the terms "utility" and "rating" interchangeably in this paper, despite the fact that there are certain differences between these two notions.

presents the hierarchy for the contextual attributes used in Section 4. In particular, Fig. 2(a) presents the contextual attribute "period of the year", Fig. 2(b) presents the contextual attribute "intent of the purchase" and Fig. 2(c) presents the contextual attribute "store" recommended to the target user, as defined in the application described in Section 4 below. Note that we have defined two levels of contextual knowledge, C_1 (less granular) and C_2 (more granular), for the two contextual attributes presented in Fig. 2(a, b). In contrast, we have defined just one level of contextual knowledge in Fig. 2(c).

Function R in (1) is initially specified as partial because it contains a partial list of ratings provided by some users for some items in certain contexts. Therefore, one of the main tasks of CARS is to make this function total by estimating all the unknown ratings. As shown in Fig. 1, this estimation can be done via contextual pre- and post-filtering and using contextual modeling. We describe these three methods further in Section 3, based on [14].

3 The pre-filtering , post-filtering and contextual modeling approaches

In this section we present the contextual approaches that we used in our studies. Section 3.1 describes the pre-filtering, Section 3.2 the two post-filtering, and Section 3.3 the contextual modeling approaches.

3.1 The Pre-Filtering Approaches

According to the pre-filtering approach (Fig. 1(a) and [14]), the contextual information is used as a label for filtering out those ratings that do not correspond to the specified contextual information. This filtering is done *before* the main recommendation method is launched on the remaining data that passed the filter. In other words, if a particular context of interest is C, then the pre-filtering method selects from the initial set of all the ratings only those corresponding to the specified context C. As a result, it generates the $Users \times Items$ matrix containing only the data pertaining to context C. Then the 2D recommendation method (e.g., collaborative filtering) is launched on this remaining dataset that passed the filter to generate recommendations for context C.

Further, [14] distinguishes between the two types of pre-filtering: *exact* and *generalized pre-filtering*. *Exact pre-filtering* (*EPF*) selects all the data referred to the exactly specified context (e.g., purchases made in January 2010). In contrast, *generalized pre-filtering* selects the data referred to some generalization of the specified context (e.g., winter 2010 purchases, as opposed to January 2010 purchases). Note that this generalization from the exactly specified context can be done in several different ways, and [14] discusses how to do it. In contrast, exact pre-filtering is uniquely defined by context *C*. We focus only on the exact pre-filtering in the rest of this paper.

3.2 The Post-Filtering Approach

According to the post-filtering (PoF) approach (see Fig. 1(b) and [14]), we first ignore all the contextual information in the data and apply a traditional 2D recommendation method, such as collaborative filtering, on the *whole un-contextual* data set, where the contextual information is dropped. Once the unknown ratings are estimated using the 2D method on this data and the un-contextual recommendations are produced, we "contextualize" these recommendations as follows.

Although there exist various methods for contextualizing the 2D recommendations, as mentioned in Section 1 and described in [14], in this paper we consider the following two approaches, called "*Weight*" and "*Filter*". Both approaches analyze data for a given user in a given context to calculate the probability with which the user chooses a certain item in the given context. After that, the recommendations obtained using this 2D method are "contextualized" by using the contextual probabilities. In the following paragraphs we describe the calculation of the contextual probabilities and the "contextualization" process.

The contextual probability $P_C(i,j)$, with which the *i*-th customer purchases the *j*-th item in context *C*, is computed as the number of neighbors (customers similar to *i*) who purchased the same item in the same context, divided by the total number of neighbors. We identified the neighborhood necessary to compute the probability $P_C(i,j)$ by using the cosine similarity applied only to transactions pertaining to context *C*. The neighborhood size *N* was selected empirically, as described in Section 4, and was set at $N = 80$ customers in our studies. The *Weight* and *Filter*

approaches differ in the way recommendations are contextualized. In particular, the *Weight PoF* multiplies each 2D rating by $P_C(i,j)$, i.e.,

$$Rating_c(i,j) = Rating(i,j) \times P_c(i,j) \quad (2a)$$

whereas, *Filter PoF* filters 2D ratings out based on a threshold value of $P_C(i,j)$:

$$Rating_c(i,j) = \begin{cases} Rating(i,j) & if \quad P_c(i,j) \geq P^* \\ 0 & if \quad P_c(i,j) < P^* \end{cases} \quad (2b)$$

where P^* is the threshold value. We varied the threshold value from 0.9 to 0.1 and obtained the best results for $P^* = 0.1$. The intuition behind the *Filter PoF* heuristic is that if only few similar users purchased an item in a particular context, then it is better not to recommend this item to the user, even though the un-contextual estimation of the rating is high and suggests recommending that item to the user (again, without the knowledge of the context).

3.3 The Contextual Modeling Approach

In this section, we present a new CM method called *contextual-neighbors* CM. This approach is based on user-based collaborative filtering (CF) and works as follows. First, for each user *i* and context *c*, we define the user profile in context *c*, i.e. the *contextual profile Prof(i, c)*. For example, if the contextual variable *c* has two values (e.g., Winter and Summer), then we have two contextual profiles for each user, one for the Winter and the other for the Summer.

Note that these contextual profiles can be defined in many different ways, some of which are presented in [2], and our approach does not depend on any particular choice of a profiling method. However, in the experimental study described in Section 4 we use the following specific contextual profiling technique. As explained in Section 2, we follow the transaction-based approach to RSs and measure the utility r_{ijc} of product *j* for user *i* in context *c* with the purchasing frequency x_{ijc} specifying how often user *i* purchased product *j* in context *c*. Then we use this measure to define contextual profile as $Prof(i, c) = (r_{i1c}, \ldots, r_{ikc})$.

We use these profiles to define similarity among users and also to define and find *N* nearest "neighbors" of user *i* in context *c*, where "neighbors" are determined using contextual profiles *Prof(i', c')* and similarity measures between the profiles.

Although this similarity can be defined in various ways, we use a popular CF approach and define the similarity measure *d* using the cosine measure as:

$$d(Prof(i',c'), Prof(i,c)) = \frac{Prof(i,c) \bullet Prof(i',c')}{\|Prof(i,c)\|^2 \times \|Prof(i',c')\|^2} = \frac{\sum_{s \in S_{ii'c}} r_{isc} r_{i'sc}}{\sqrt{\sum_{s \in S_{ii'c}} r_{isc}^2} \sqrt{\sum_{s \in S_{ii'c}} r_{i'sc}^2}} \quad (2c)$$

where r_{isc} and $r_{i'sc}$ are the ratings of item *s* assigned by user *i* and user *i'* respectively in context *c*. $S_{ii'c}=\{s \in Items \mid r_{isc} \neq \varnothing \wedge r_{i'sc} \neq \varnothing\}$ is the set of all items co-rated by both user *i* and user *i'* in context *c*.

Then we find *N* nearest neighbors for the *(i, c)* pair by identifying pairs *(i', c')* such that *d(Prof(i', c'), Prof(i, c))* is the largest among all the candidate pairs *(i', c')* subject to the following constraints:

- Mdl_1: There are no constraints on the set of *(i', c')* pairs, and we select *N* pairs that are the most similar to *(i, c)*.
- Mdl_2: we select an equal proportion of pairs *(i', c')* corresponding to each context *c* (e.g., if the contextual variable has only two values, Winter and Summer respectively, and the neighborhood size is 80, we select 40 neighbors from Winter and 40 from Summer).
- Mdl_3: we select *N* pairs *(i', c')* that are the most similar to *(i, c)* corresponding to each context *c* at the same level of the context of interest (e.g., if the context of interest is "Winter Holiday" in Fig. 2(a), we select the neighborhood by using only profiles referred to level C_2 of that contextual variable).
- Mdl_4: we select an equal proportion of pairs *(i', c')* corresponding to each context *c* at the same level of the context of interest (e.g., if the context of interest is "Winter Holiday" in Fig. 2(a) and the neighborhood size is 80, we define the neighborhood by using 20 users from the context "Winter Holiday", 20 users from the context "Winter Not Holiday", 20 users from the context "Summer Holiday" and 20 users from the context "Summer Not Holiday").

Once we introduced the contextual neighbors approach and its four implementations Mdl_1, Mdl_2, Mdl_3 and Mdl_4, we next want to (a) compare them to determine which one is the best among them, and (b) see how it compares against the pre- and post-filtering methods. Therefore, the rest of the paper focuses on the comparison of contextual modeling, contextual pre-filtering, post-filtering and the un-contextual methods in order to determine which method performs better. Since

we do these comparisons empirically, we present the experimental setup in the next section before presenting the actual results of our experiments in the subsequent sections.

4 Experimental Setup

We compared pre-filtering vs. post-filtering vs. contextual modeling vs. un-contextual recommendations across a wide range of experimental settings. First, we selected three different data sets having contextual information. The first dataset (DB1) comes from an e-commerce website commercially operating in a certain European country which sells electronic products to approximately 120,000 users and contains about 220,000 purchasing transactions during an observation period of three years. For this dataset, we selected the time of the year as a contextual variable. Its hierarchical structure is presented in Fig. 2(a). The classification into Summer or Winter and Holiday or Not Holiday is based on the experiences of the CEO of the e-commerce website that we used in our study. He defined June, July, August, April, May and September as "Summer". The first three months of this period are considered as "Holiday" while the remaining as "Not Holiday". Also he defined October, November, December, January, February and March as "Winter". The first three months of this period are considered as "Holiday" while the remaining as "Not Holiday". According to this definition, a purchase made, for example, on December 1 is labeled as "Winter Holiday". The data was pre-processed by excluding about 80,000 customers who made only one single transaction (for these customers, it is hard to generate any meaningful recommendations due to the lack of preference data), around 500 customers who had any kind of abnormal behavior such as buying the same product for 1,000 times at the same time (this was probably a retailer), and around 38,000 customers who had transactions either only in the first two years or only in the third year. The reason for this last elimination is that we used the transactions in the first two years as training set and those in the third year as validation set, as explained below. The resulting dataset contained about 1,500 users and about 10,000 transactions.

The second dataset (DB2) is taken from the study described in [2]. First, a special purpose browser was developed to help users navigate Amazon.com website and

purchase products on its site. This browser was made available to a group of students who were asked to navigate and simulate purchases on Amazon.com during a period of four months based on the incentive scheme developed for this study. While navigation was real on Amazon.com, purchasing was simulated. Once a product was selected by a student to be purchased, the browser recorded the selected item, the purchasing price and other useful characteristics of the transaction and this information was stored in the database. In addition, the student was asked at the beginning of each browsing session to specify its context, which was the intent of a purchase in our case, i.e., whether the purchase would be intended for personal purpose or as a gift, for which specific personal purpose, and for whom the gift was intended. The structure of this contextual variable IntentOfPurchase is presented in Fig. 2(b). Further, the data was pre-processed by excluding the students who made less than 40 transactions and eliminating the students who had any kind of misleading or abnormal behavior. The resulting number of students was 556, and the total number of purchasing transactions for the students was 31,925.

The third dataset (DB3) comes from an e-commerce website which sells comics and comics-related products, such as T-shirts, DVDs and various gadgets. It contains about 50,000 transactions and 5,000 users. In this case, we used the store (i.e., the section in the Web site where products are bought), as a contextual variable, distinguishing whether the product is bought in "Wearing apparel", "DVD", "Miniseries" or "Special issues" section (store) of the website (see Fig. 2(c)). This contextual variable store specifies the immediate browsing activity in which the customer was engaged just before the recommendation by identifying the location of the customer on the website. The importance of this contextual variable comes from the expectation that customers' behavior changes when navigating and buying products in different sections of the Web site. For instance, purchasing behavior of a comics book can be very different from the purchasing behavior of clothes (such as T-shirts). In a real-time recommender system, when a customer enters a specific store of the website, the system should use this context (the store type) to focus mainly on the recommendations pertinent to that store.

In our study, we recommend product categories instead of individual items because the e-commerce applications that we consider have very large numbers of

items (hundreds of thousands or even millions). Therefore, if single items were used, the conversion from implicit to explicit ratings would not work due to the low amount of rated data (e.g., many of the products were not purchased at all). We tried different item aggregation strategies and found that the best results are for 14 categories for DB1, 24 categories for DB2 and 136 for DB3. For our three datasets, we aggregated items into categories of products according to the classification provided by the Web site product catalogue. In particular, for the comics dataset, DB3, items are aggregated into the main character associated with these items. For example, a "Spider Man" comics book and a DVD "Spider Man" are aggregated into the category "Spider Man". Similarly, all DVDs related to "Spider Man" are aggregated into one category. Moreover, there is no correlation between aggregated items and contexts because each main character is included in each store. The results described below are referred to these item aggregation levels.

The utility of items for the customers were measured by the purchasing frequencies, as described in Section 2 for the transaction-based RSs. This measure serves as a proxy for the rating of how much a customer likes a product and how useful that product is to the customer. Estimations of unknown utilities were done by using a standard user-based collaborative filtering (CF) method [19]. According to the CF approach, the neighborhood was formed using the cosine similarity [20], which is given by:

$$sim(i,z) = cos(\vec{i},\vec{z}) = \frac{\vec{i} \bullet \vec{z}}{\|\vec{i}\|_2 \times \|\vec{z}\|_2} = \frac{\sum_{s \in S_{iz}} r_{i,s} r_{z,s}}{\sqrt{\sum_{s \in S_{iz}} r_{i,s}^2} \sqrt{\sum_{s \in S_{iz}} r_{z,s}^2}} \quad (3)$$

where $r_{i,s}$ and $r_{z,s}$ are the ratings of item s assigned by user i and user z respectively. $S_{iz}=\{s \in Items \mid r_{i,s} \neq \varnothing \wedge r_{z,s} \neq \varnothing\}$ is the set of all items co-rated by both user i and user z, and $\vec{i} \bullet \vec{z}$ denotes the dot-product between the vectors $\vec{i}$ and $\vec{z}$.

The neighborhood size N was set to N = 80 users as follows. We performed an experiment where we varied the neighborhood size, moving from 30 to 200 users, and we measured the F-measure. We performed this experiment for each dataset. In general, the F-measure increased as we increased the number of neighbors; however, these improvement gains stopped when we set the neighborhood size

around 80 users, while the F-measure decreased when we set the neighborhood size over 80 users. So, we decided to set $N = 80$ users as the optimal neighborhood size for our experiments.

The value of the unknown ratings $r_{i,j}$ for user i and item j was computed as:

$$r_{i,j} = k \sum_{i' \in \hat{I}} sim(i,i') \times r_{i',j} \quad (4)$$

where $\hat{I}$ denotes the set of N users most similar to user i and who have rated item j. The multiplier k serves as a normalizing factor computed as:

$$k = 1 / \sum_{i' \in \hat{I}} |sim(i,i')| \quad (5)$$

We used the two post-filtering approaches (*Weight* and *Filter*) described in Section 3.2 and the four contextual modeling approaches (Mdl_1 to Mdl_4) described in Section 3.3. For the pre-filtering case, we used the exact pre-filtering (*EPF*) method described in Section 3.1. Furthermore, we used the same user-based CF method for estimating unknown ratings in both the pre- and the post-filtering cases to make sure that we were comparing "apples with apples". We decided to use a classical, popular and proven CF method, namely a user-to-user algorithm. The reason is that the focus of this research is the comparison among methods to incorporate context in a recommendation process and between these methods and the un-contextual method. Therefore, it is important to apply the same CF method across all of the experimental settings when comparing the relative performance of different contextual and un-contextual methods. We have applied this method not only to the un-contextual version but as the core CF method across all of our contextual and un-contextual methods.

In summary, in this study we compared seven contextual methods, i.e., pre-filtering (EPF), 2 post-filtering and 4 contextual modeling methods, and an un-contextual method to determine if one dominates another and in which circumstances.

The experiments were performed for datasets DB1, DB2 and DB3 for all the levels of contextual knowledge (un-contextual, C_1 and C_2), as presented in Fig. 2. Further, we have performed t-tests in order to determine if the chosen contextual variables matter. To do this, we split the ratings based on the values of the contextual variable (e.g. winter/summer, gift/personal) and performed the pairwise t-tests [21] to see if there are statistically significant differences in the

rating distributions. The results of these tests were statistically significant (at 95% level) and demonstrated that the contextual variables TimeOfTheYear, IntentOfPurchase and Store matter.

In our study, we used the following two recommendation strategies: "find all good items" and "recommend top-k items". In the "*find all good items*" approach, the recommender system suggests all the "recommendable" items, i.e., the items having the rating value above a certain threshold. In our study, we set this threshold value to 1. Therefore, if the system predicts a rating greater or equal to 2, we decide to "recommend" that item, otherwise we do not. In the "*recommend top-k items*" approach, only the "top-*k*" items having the *k* highest ratings for a particular user are recommended to that user. In our study, we varied the number of top-*k* recommended items from 1 to 4.

Finally, we used Precision, Recall, F-Measure, Mean Absolute Error (*MAE*) and Root Mean Square Error (*RMSE*) [22] as the performance measures in our experiments, as done in [8]. We computed MAE and RSME in a standard way [21] by taking absolute and squared differences between the estimated and actual values of estimated utilities, as defined by equation (4), since the utilities are measured using discrete variables [23, 24, 25], as described above. We also computed Precision and Recall as follows. For the "find all good items" strategy, we set the threshold between relevant and irrelevant items equal to 1, thus, assuming that if an item is purchased more than once, it is relevant ("good"), and we recommend it; otherwise, we do not. Then, we verify if the recommended item was actually purchased in the validation set. If it was, we consider that as a "good" recommendation, otherwise as a "bad" one. For the "recommend top-k items" strategy, we determined the top-k items as "good" items to be recommended to a user. Then we compared those with the actual items purchased by the user to compute Precision and Recall in a standard manner.

Finally, we divided each dataset into the training and the validation sets, the training set containing 2/3 and the validation set 1/3 of the whole dataset. For the DB1 dataset, the first two years were the training set and the third year was the validation set. For the DB2 dataset, we randomly split it in 2/3 for the training set and the remaining 1/3 for the validation set (in this case, it was impossible to make a good temporal split because all the transactions were made within a

couple of months). For the DB3 dataset, the first nine months were the training set and the last three months were the validation set.

To summarize, we performed numerous experiments across three datasets, levels of item aggregation, neighborhood sizes, recommendation methods (un-contextual, *EPF, Filter PoF, Weight PoF, Mdl_1, Mdl_2, Mdl_3, Mdl_4*), contextual levels (C_1 and C_2), recommendation strategies ("find all good items" and "top-k"), the number of recommended items (for the "top-k" strategy moving from k=1 to k=4), and different performance metrics. If we consider each combination of these experimental settings as one individual experiment, we had 1,872 total experiments. Furthermore, each such individual experiment required on the order of 3 or 5 computations of the performance measures (depending if it was possible to compute also MAE and RMSE), thus resulting in 8,208 computations of the performance results in total.

5 Comparison Between Un-Contextual and Contextual Recommender Systems

As a first step in our empirical studies, we compared the un-contextual with the contextual recommendations for the pre-filtering, the post-filtering and the contextual modeling cases. We did this for two reasons. First, we wanted to determine which method dominates the other. Second, we needed the results of this analysis later when comparing the contextual approaches, as described further in Sections 6 and 7. In Section 5.1 we compare the un-contextual and the pre-filtering, in Section 5.2 the un-contextual and the post-filtering approaches, in Section 5.3 the un-contextual and the contextual modeling approaches.

5.1 Un-Contextual Vs. Pre-Filtering

In this section, we compare the traditional un-contextual 2D collaborative filtering and the exact pre-filtering (*EPF*) methods (as described in Section 3.1). This type of comparison has been studied in [8] before, where it was shown that in certain cases the un-contextual approach dominates, while in other cases the contextual one dominates the un-contextual method. However, the work reported in [8] was done in the context of multi-dimensional recommendations and only for one small dataset having 62 users, 202 movies and 1457 ratings. In this study, we wanted to

provide a more extensive comparison of the un-contextual and the pre-filtering methods across several and bigger datasets and across numerous other experimental settings in order to either further support the claims of [8] or identify some caveats where our conclusions would differ from [8]. Furthermore, we also needed to conduct this comparison for the uniformity reasons because we also do the comparison of the post-filtering and the contextual modeling approaches to the un-contextual approach in this paper, and this has not been done in [8] or anywhere else in the literature.

As described in Section 4, we compared the un-contextual and the *EPF* contextual methods for the case of user-based collaborative filtering, across the datasets DB1, DB2 and DB3, across multiple levels of classification hierarchy (C_1 and C_2), across different recommendation strategies ("find all good items" and "top-k items") and different performance measures. We split the data into the training and testing sets as described in Section 4. The results of the comparison are presented in Fig. 3, Fig. 4 and Fig. 5 for "find all good items", "top-k=1" and "top-k=4" strategies respectively, where different values of contextual variables are plotted on the x-axis. Fig. 3 compares the Precision, Recall, F-measure, *MAE* and *RMSE* performance measures of the contextual and un-contextual RSs for the datasets DB1, DB2 and DB3 when the "find all good items" recommendation strategy is used. Fig. 4 presents the comparison of Precision, Recall and F-measure for the datasets DB1, DB2 and DB3 when the "top-k=1" recommendation strategy is used [2]. Fig. 5 shows the same comparison when the "top-k=4" recommendation strategy is used. We made experiments with k=2 and k=3 as well, however for the sake of brevity, we only present two of the four cases.

[2] Note that the *MAE* and *RMSE* measures are not applicable to the "top-k" strategy because these measures are calculated on the *whole* matrix of predicted ratings.

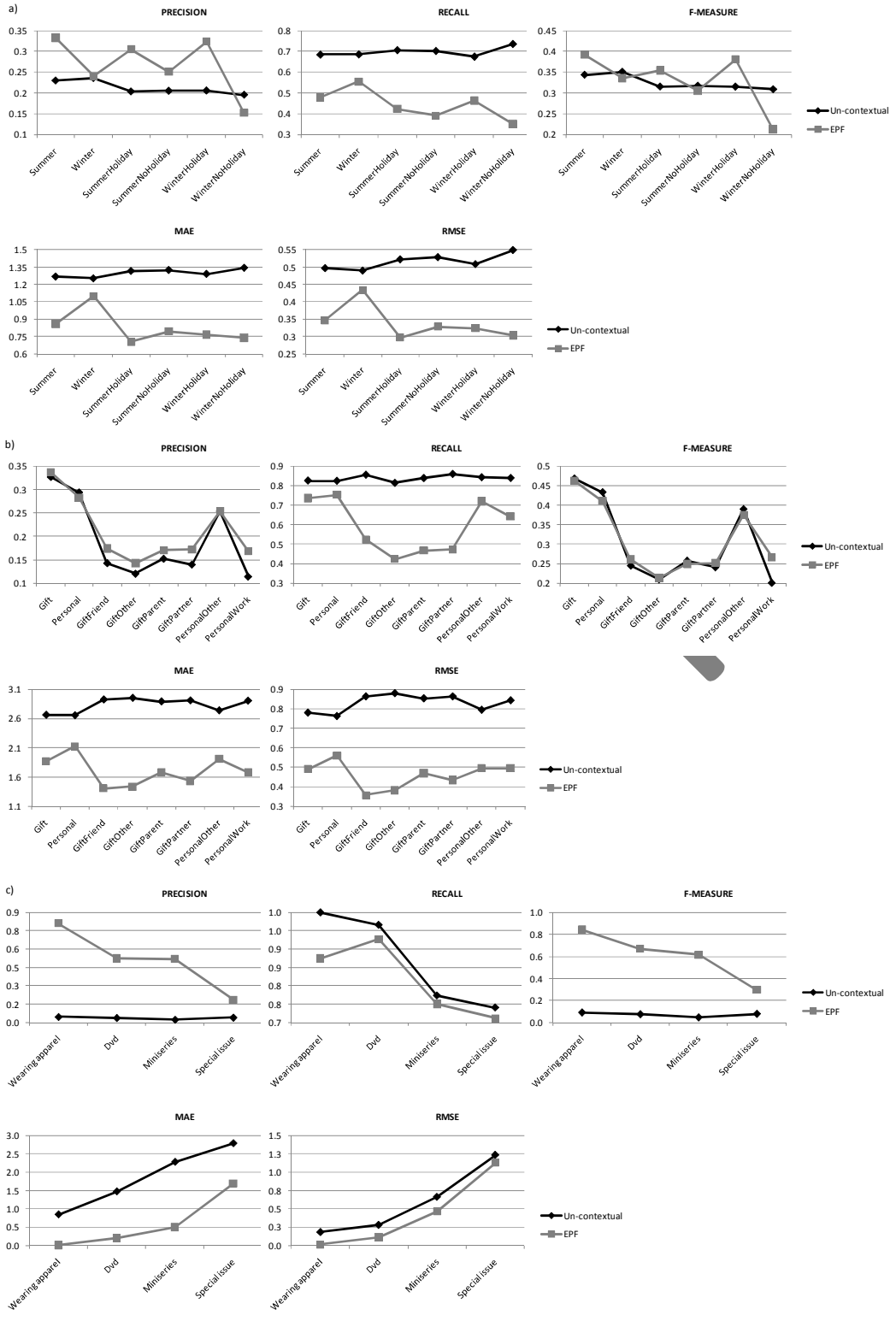

Fig. 3 Comparison between EPF and un-contextual using "find all good items" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

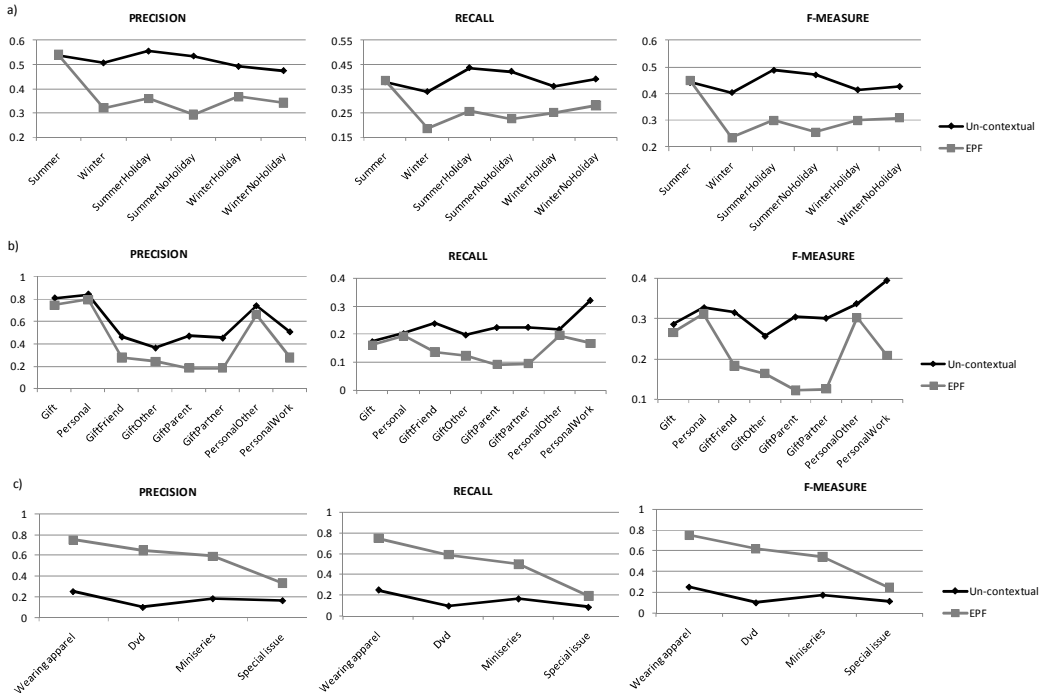

Fig. 4 Comparison of Precision, Recall and F-measure of EPF and un-contextual using "top-k=1" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

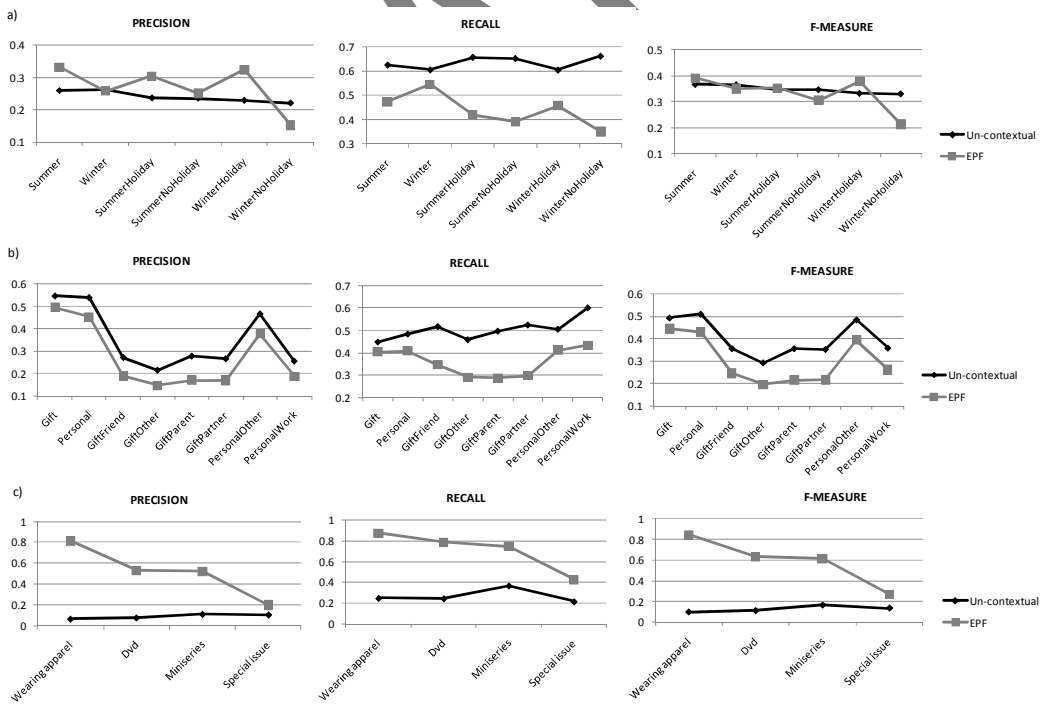

Fig. 5 Comparison of Precision, Recall and F-measure of EPF and un-contextual using "top-k=4" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

Fig. 3, Fig. 4 and Fig. 5 demonstrate that the un-contextual method dominates the contextual *EPF* method in some cases and is inferior in other cases. For example, if we consider the "find all good items" recommendation strategy, Fig. 3(a) shows that the *EPF* method outperforms the un-contextual method in terms of F-measure in one case out of two at the level C_1 and in two cases out of four at the level C_2. Fig. 3(b) shows that the *EPF* method and the un-contextual one have nearly the same performance, and Fig. 3(c) shows that the *EPF* method outperforms the un-contextual method. If we consider the "top-k=1" recommendation strategy, Fig. 4(a) and Fig. 4(b) show that the un-contextual outperforms the *EPF* in terms of F-measure and Fig. 4(c) shows that the *EPF* outperforms the un-contextual in terms of F-measure.

If we consider the "top-k=4" recommendation strategy, Fig. 5(a) shows that the *EPF* method outperforms the un-contextual method in terms of F-measure in one case out of two at the level C_1 and in two cases out of four at the level C_2. Fig. 5(b) shows that the un-contextual outperforms the *EPF* in terms of F-measure and Fig. 5(c) shows that the *EPF* method outperforms the un-contextual method. From all this, we conclude that the *EPF* dominates the un-contextual method more often when the strategy is "find all good items" and "top-k" with k=4; but rarely with the "top-k" strategy for k=1.

This conclusion that *EPF* dominates the un-contextual case in some cases in terms of the F-measure and is dominated in other cases is consistent with the finding reported in [8] and has the following intuitive explanation. When the contextual information is very granular, this results in a more homogeneous rating prediction model but also leads to data sparsity issues (only few context-specific ratings are used to build the model). In contrast, when the contextual information is too coarse, we can have enough data to build the model, but it is more heterogeneous. This conflict between homogeneity of the model and rating sparsity produces mixed results when comparing *EPF* and the un-contextual model. Furthermore, as Fig. 3, Fig. 4 and Fig. 5 show, *EPF* outperforms the un-contextual model clearly in terms of *MAE* and *RMSE*, usually in terms of precision, but seldom in terms of recall measures. This conclusion is especially interesting for the *MAE* and *RMSE* errors (as reported in Fig. 3) because [8] did not do the comparison in terms of these performance measures.

5.2 Un-Contextual Vs. Post-Filtering

In this section, we compare the un-contextual and the post-filtering methods. Unlike exact pre-filtering, there exist many post-filtering methods. It is important to select appropriate post-filtering methods because the comparison can depend very significantly on the choice. As explained in Section 3.2, we proposed to use the *Weight PoF* and *Filter PoF* methods in our studies. We will show in this section that this was an appropriate selection in the sense that one method is "good" whereas another one is "bad" in the sense explained below.

We compared the *Weight PoF* and *Filter PoF* and the un-contextual method used in Section 5.1 across various experimental conditions described in Section 4. Fig. 6, Fig. 7 and Fig. 8 present the results of comparison across various performance measures, all the levels of context for the *TimeOftheYear*, *IntentOfPurchase* and *Store* contextual variables, different datasets (DB1, DB2 and DB3) and recommendation strategies ("find all good items", "top-k=1" and "top-k=4").

As Fig. 6, Fig. 7(c) and Fig. 8(a, c) show, the post-filtering method *Filter PoF* dominates the un-contextual case across all the levels of context for the F-measure: for DB1, the difference between contextual and un-contextual models is 18% on average, for DB2 it is 28% on average and for the DB3 it is 89% using a "find all good items" strategy. This constitutes a substantial performance improvement for all the three datasets.

Furthermore, *Filter PoF* dominates the un-contextual RS not only in terms of F-measure, but also in terms of other measures, including Precision, *MAE* and *RMSE* (but not in terms of Recall). This observation is always valid when the recommendation strategy is "find all good items", and "top-k" with k=4 (except the case shown in Fig. 8(b), where k=4 and Filter PoF dominates the un-contextual RS in several contexts though not all of them). The only cases in which *Filter PoF* does not dominate the un-contextual RS are those of using a "top-k" strategy with k=1 in two datasets shown in Fig. 7(a) and Fig. 7(b).

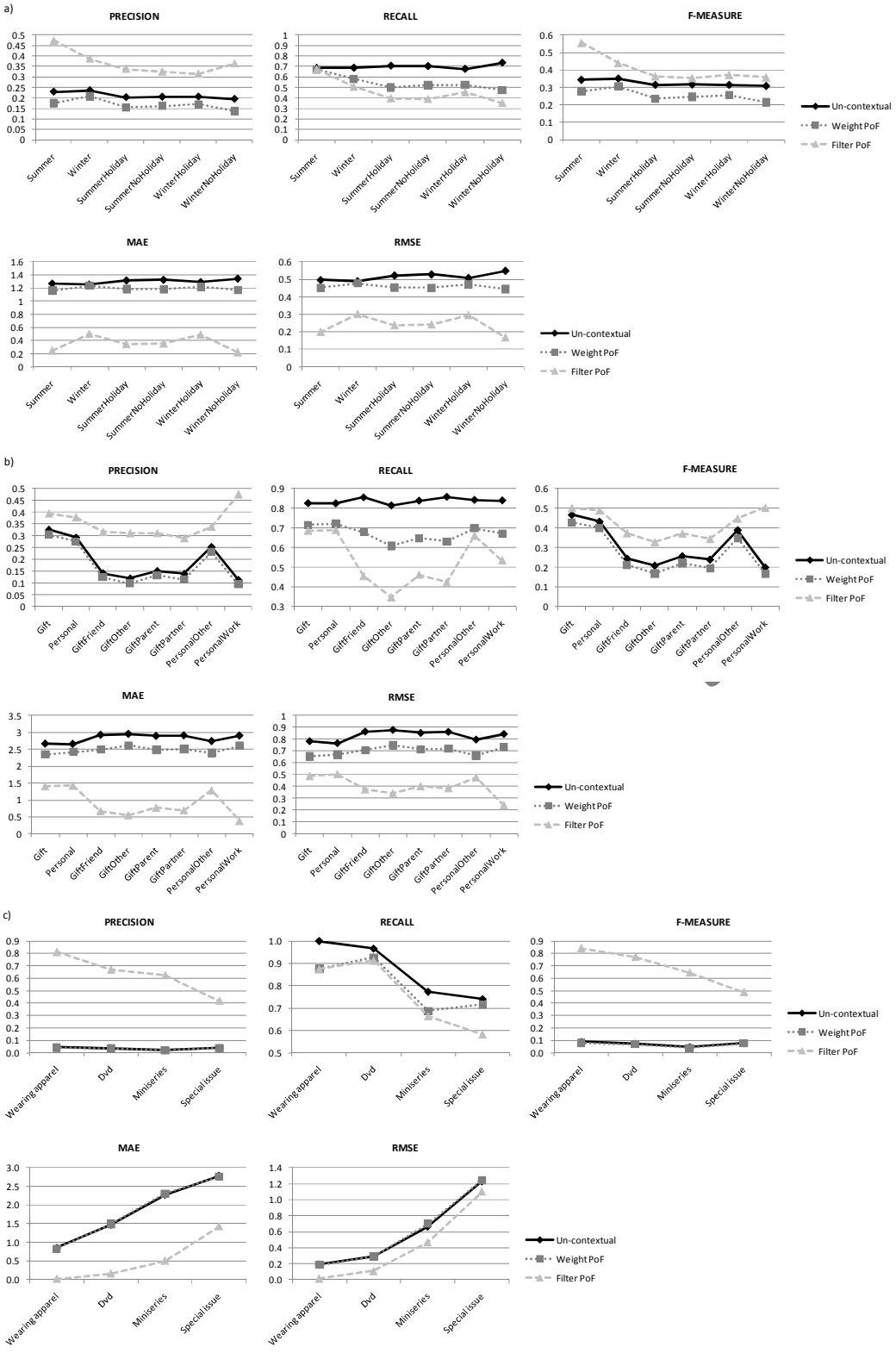

Fig. 6 Comparison between Weight Pof, Filter Pof and un-contextual using "find all good items" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

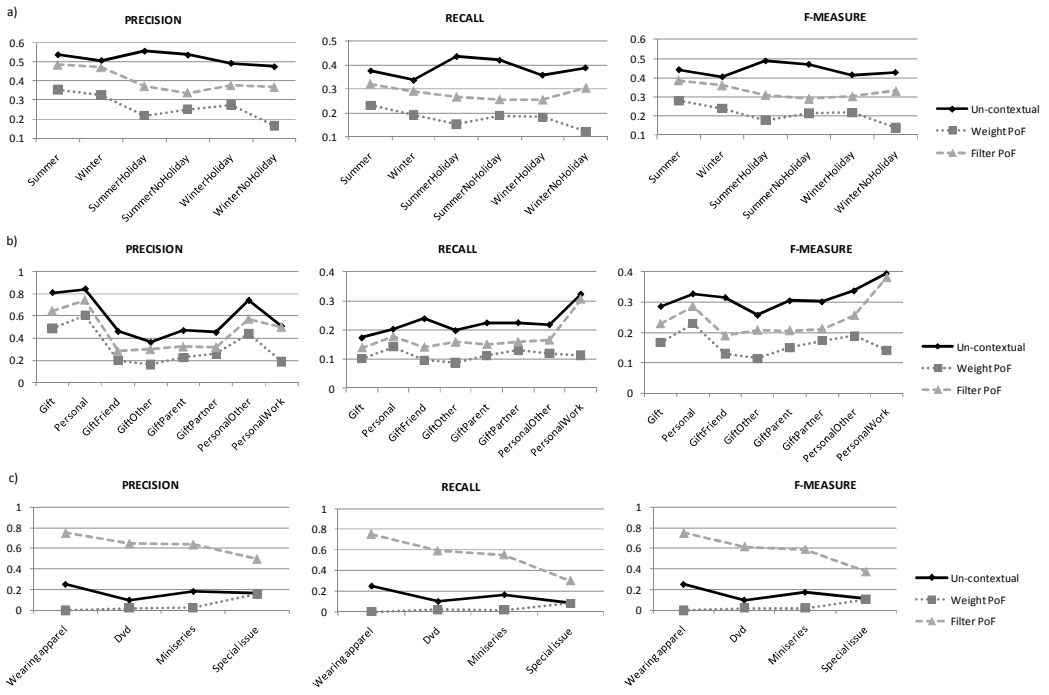

Fig. 7 Comparison of Precision, Recall and F-measure of Weight PoF, Filter PoF and un-contextual using "top=1" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

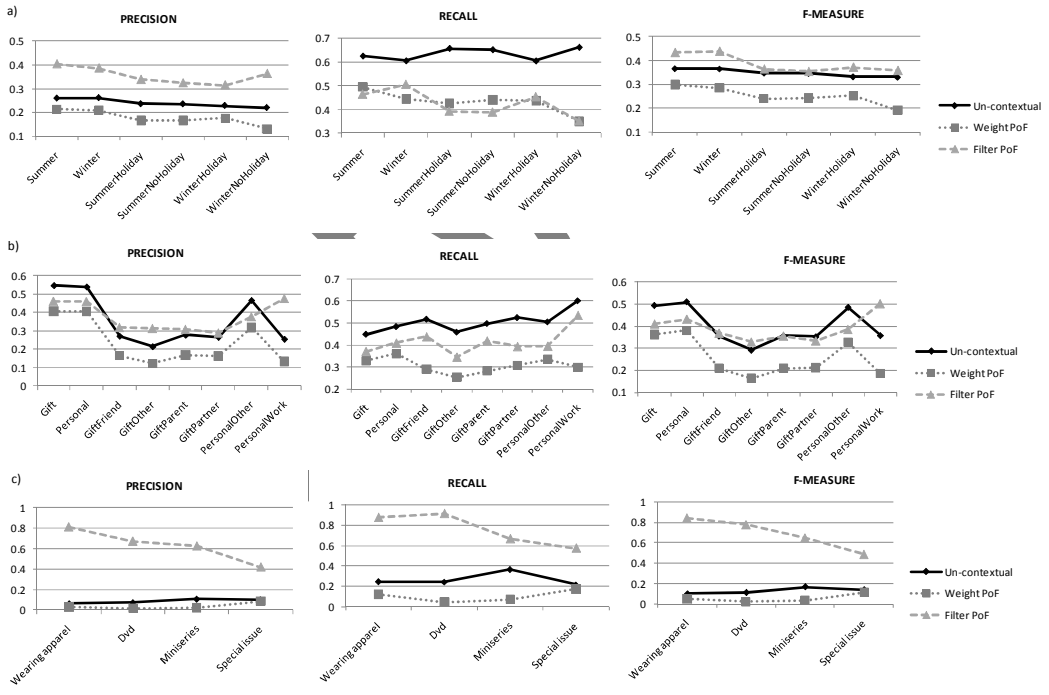

Fig. 8 Comparison of Precision, Recall and F-measure of Weight PoF, Filter PoF and un-contextual using "top=4" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

However, using this kind of recommendation strategy (top k=1) is seldom used in most business settings, where the number of recommended items is usually greater than one (e.g., recommender systems at Amazon, Netflix and other companies usually show several books, movies and other items at a time). In contrast to this, the un-contextual method predominately dominates the *Weight PoF* in terms of Recall, Precision and F-measure, as is evident from Fig. 6, Fig. 7 and Fig. 8.

All this makes us to conclude that the comparison of the post-filtering and un-contextual methods depends very significantly on the type of the post-filtering method being used, i.e., *Filter PoF* is a "good" method because it mostly outperforms the un-contextual case, while *Weight PoF* is a "bad" one because it is inferior to the un-contextual case.

This observation is not surprising because the post-filtering method takes the results of the same 2D un-contextual method and reorders them based on the context-based post-filtering heuristic. If the heuristic is "good" (such as *Filter*), then this re-ordering improves recommendation quality; otherwise, it can make the results even worse (as for heuristic *Weight*).

5.3 Un-Contextual Vs. Contextual Modeling

In this section, we compare the un-contextual and the contextual modeling methods. As explained in Section 3.3, we consider four methods Mdl_1, Mdl_2, Mdl_3, Mdl_4 in our study and compare them to the un-contextual method used in Section 5.1 across various experimental conditions described in Section 4. Fig. 9, Fig. 10 and Fig. 11 present the results of the comparison across all five performance measures (Precision, Recall, F-measure, *MAE* and *RMSE*) for all the levels of context for the contextual variables time of the year, intent of a purchase and store, for the three datasets DB1, DB2 and DB3 and for each recommendation strategy ("find all good items", "top-k=1" and "top-k=4").

As Fig. 9 shows, when using a "find all good items" strategy, the contextual modeling approaches dominate the un-contextual in terms of Precision, F-measure, *MAE* and *RMSE*, while do not in terms of Recall. When using a "top-k" strategy, as Fig. 10 and Fig. 11 show, the contextual modeling approaches

dominate the un-contextual in some cases, whereas they are dominated in other cases.

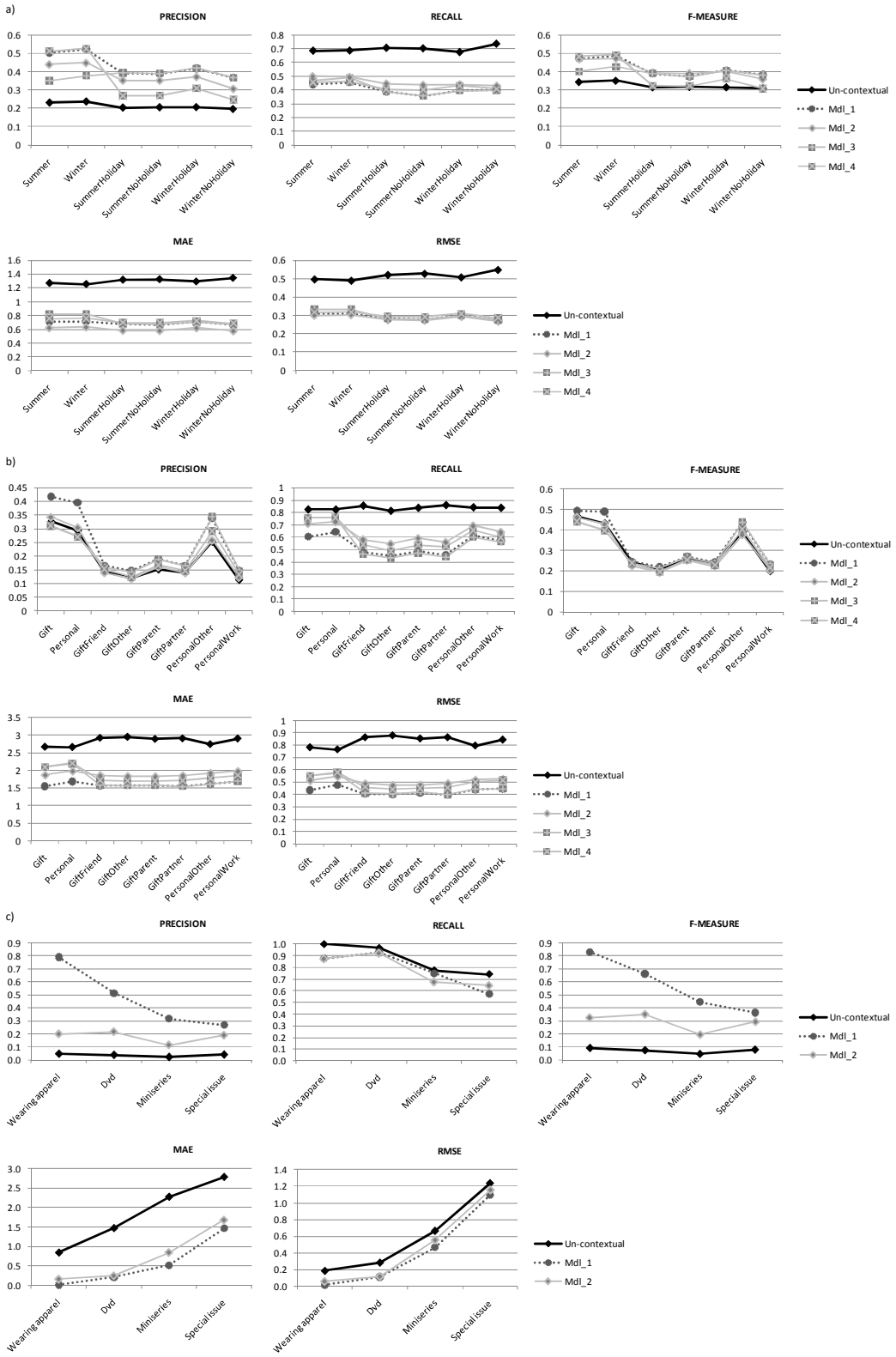

Fig. 9 Comparison between Mdl_1, Mdl_2, Mdl_3, Mdl_4 and un-contextual using "find all good items" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

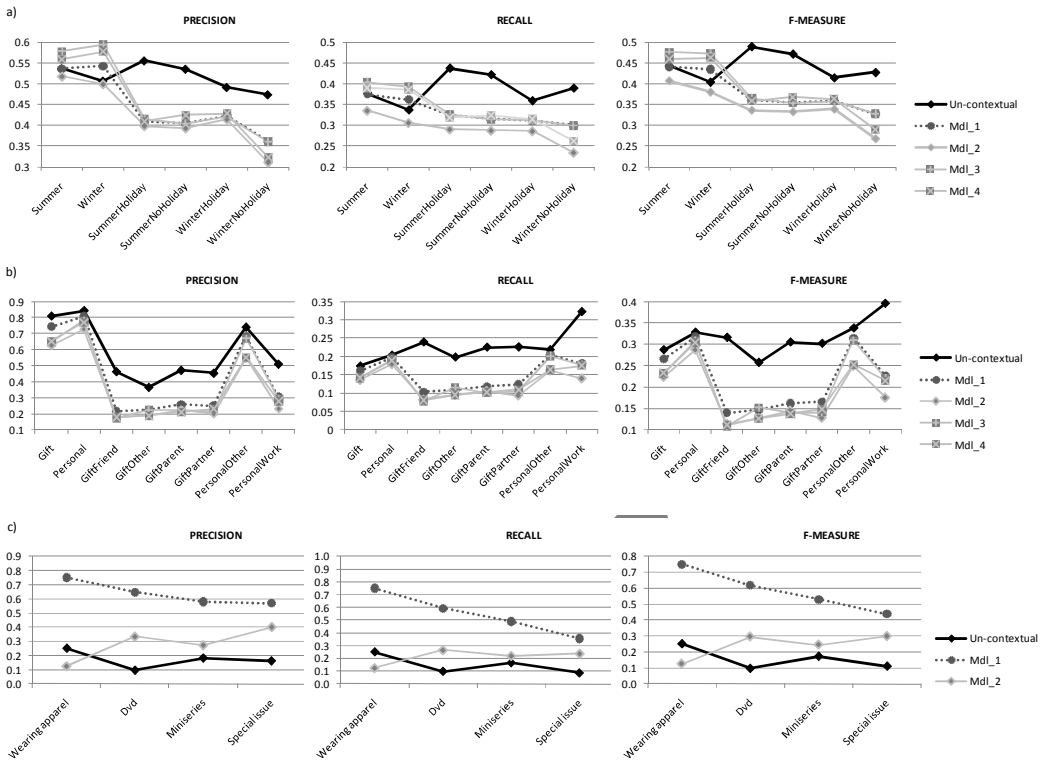

Fig. 10 Comparison of Precision, Recall and F-measure of Mdl_1, Mdl_2, Mdl_3, Mdl_4 and un-contextual using "top=1" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

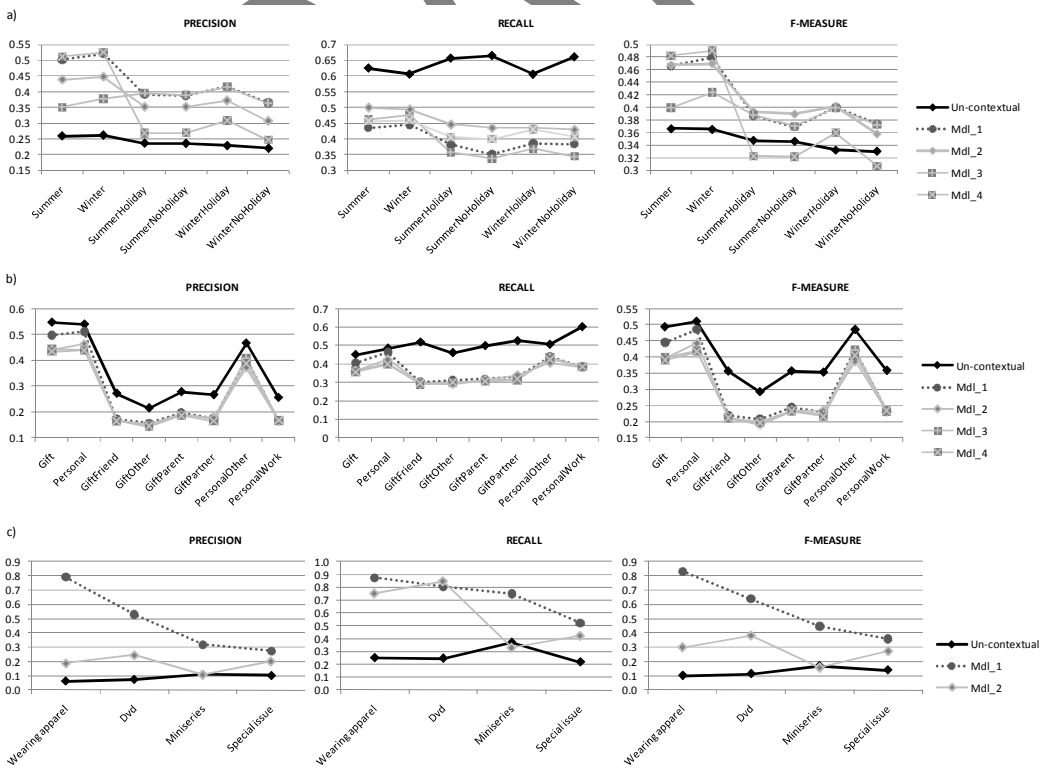

Fig. 11 Comparison of Precision, Recall and F-measure of Mdl_1, Mdl_2, Mdl_3, Mdl_4 and un-contextual using "top=4" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

The results also demonstrate that the performances of the four CM approaches are not significantly different. For instance, the difference between Mdl_1 and Mdl_2 for DB1 is 0.008, 0.13 and 0.02 in terms of F-measure, *MAE* and *RMSE* measures, respectively; for DB2 is 0.09, 0.17, 0.18; and for DB3 is 0.45, 2 and 0.7, respectively, which is not very significant.

In comparison, the performance differences between various post-filtering methods, as reported in Section 5.2, are much more pronounced in comparison to these differences. Also, Mdl_1 slightly dominates other CM methods in some cases (see Fig. 9(c)) and is dominated in other cases (see Fig. 10(a)). This makes sense because the N neighbors are selected for Mdl_1 in an unconstrained manner, whereas they are selected based on various constraints for the other three approaches.

6 Comparison of the Pre-Filtering, Post-Filtering and Contextual Modeling Methods

In this section, we compare performance of the pre-filtering, post-filtering and contextual modeling methods. As explained in Section 5.2, the performance of the post-filtering methods may significantly depend on the type of the post-filtering approach being used. Therefore, we decided to use two post-filtering methods in our experiments, *Weight PoF* and *Filter PoF*, to account for these differences. In contrast, as explained in Section 5.3, the performances of the four CM approaches are not significantly different. For this reason, we decided to keep only one CM method, namely Mdl_1, when presenting the comparison results among all the approaches since using all the four CM methods would not add anything to the results but would only make the graphs more "crowded".

Fig. 12, Fig. 13 and Fig. 14 present the comparison results among the two post-filtering methods *Weight PoF* and *Filter PoF*, the exact pre-filtering (*EPF*) and the contextual modeling method Mdl_1 across each contextual level and each dataset (DB1, DB2 and DB3) and using the "find all good items" (Fig. 12), "top-k =1" and "top-k =4" recommendation strategies (Fig. 13 and Fig. 14).

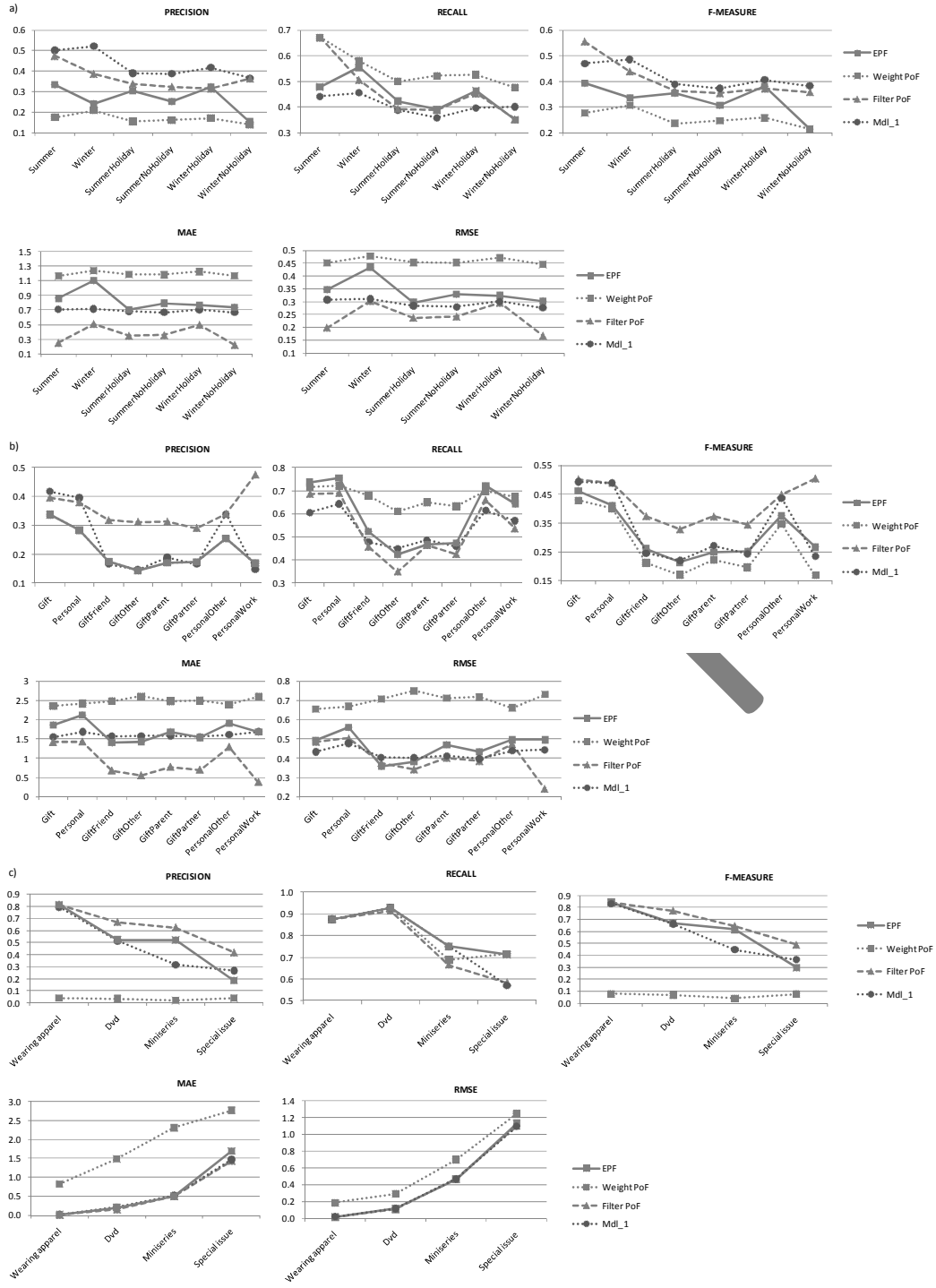

Fig. 12 Comparison between EPF, Weight PoF, Filter PoF and Mdl_1 using "find all good items" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

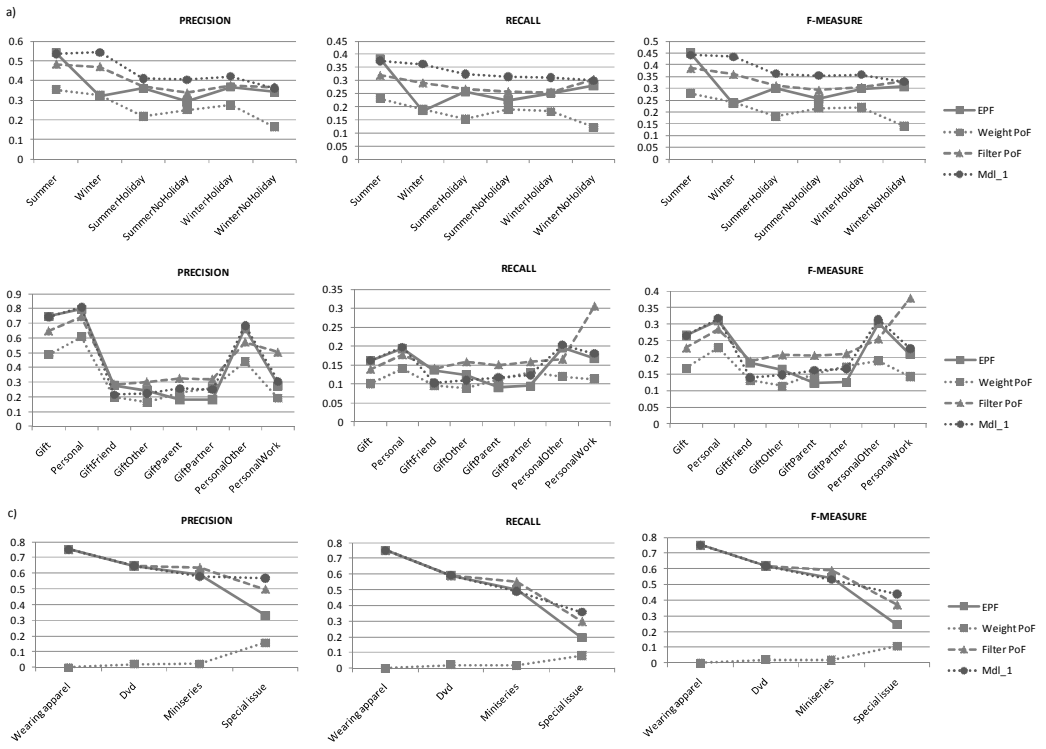

Fig. 13 Comparison of Precision, Recall and F-measure of EPF, Weight PoF, Filter PoF and Mdl_1 using "top=1" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

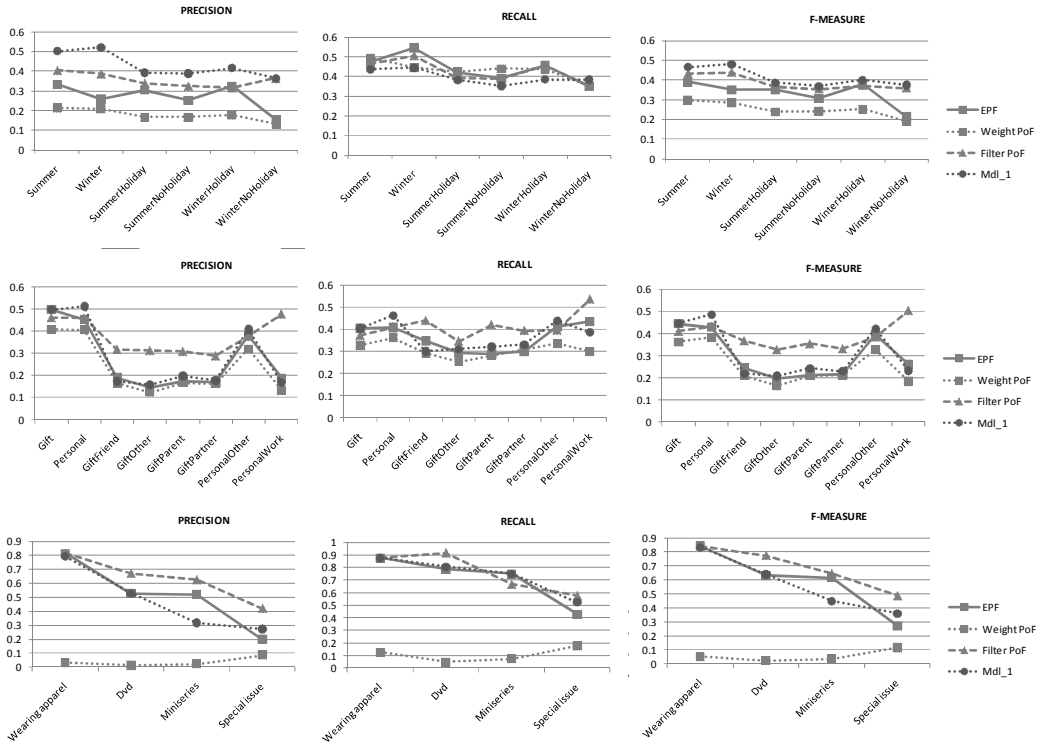

Fig. 14 Comparison of Precision, Recall and F-measure of EPF, Weight PoF, Filter PoF and Mdl_1 using "top=4" strategy for (a) DB1, (b) DB2 and (c) DB3 datasets

As Fig. 12 shows, when using a "find all good items" strategy, the *Filter PoF* method outperforms any other contextual approach in terms of *MAE* and *RMSE*. It is often superior in terms of Precision and F-measure (only in two cases it is dominated by Mdl_1). The *Weight PoF* is almost always the worst approach, except in terms of Recall, where Mdl_1 and *EPF* show very similar values. If the graphs reporting F-measure and Precision are considered, in two datasets *Filter PoF* dominates the other approaches (DB2 and DB3, see Fig. 12(b, c)). In these cases, the performance of *EPF* and Mdl_1 are similar, while the worst system is *Weight PoF*. In the third dataset the *Filter PoF* is very similar to Mdl_1 and both dominate the *EPF*, while the *Weight PoF* is still the worst approach (DB1, see Fig. 12(a)). Similar observations can be made for the graphs reporting *MAE* and *RMSE*. In these cases the *Filter PoF* is the best approach, the *Weight PoF* the worst, *EPF* and Mdl_1 are in the middle. The comparison looks different only in terms of Recall. As Fig. 12 demonstrates, the *Weight PoF* method is the best in two cases (DB1 and DB2, see Fig. 12(a, b)), while all the other approaches have similar performances. In the third dataset all methods seem equivalent (DB3, see Fig. 12(c)). The difference between *Filter PoF* and Mdl_1 in terms of F-Measure is 3% on average for DB1, 37% for DB2 and 24% for DB3.

The difference between *Filter PoF* and *EPF* in terms of F-measure is 21% on average for DB1, 40% for DB2 and 21% for DB3, while the difference between *EPF* and *Weight PoF* in terms of F-measure is 20% on average for DB1, 16% for DB2 and 87% for DB3. The difference between Mdl_1 and *Weight PoF* models in terms of F-measure is 38% on average for DB1, 19% for DB2 and 87% for DB3, while the difference between Mdl_1 and *EPF* is 21% on average for DB1, 3% for DB2 and 5% for DB3. These results mean that the performance of the CM approach (as represented by Mdl_1) is very similar to that of the *EPF* method.

When using a "top-k=1" and "top-k=4" recommendation strategies and considering Precision and F-measure, *Filter PoF* is the best approach in many cases. It is outperformed by Mdl_1 only for DB1, in terms of Precision for top-k=4 and in both Precision and F-measure for top-k=4 (see Fig. 13(a) and Fig. 14(a), respectively). The CM and *EPF* approaches are often the second best and their performance is quite similar. *Weight PoF* either is the worst approach or is equivalent to other approaches. In the case of top-k=1 for DB2 and DB3, the

differences among the various approaches are quite low. Considering top-k=1, the difference between *Mdl$_1$* and *Filter PoF* models in terms of F-Measure is 13% on average for DB1, -19% for DB2 and 1% for DB3, whereas the difference between *Filter PoF* and *EPF* in terms of F-measure is 7% on average for DB1, 13% for DB2 and 11% for DB3. The difference between *Mdl$_1$* and *EPF* is 19% on average for DB1, 2% for DB2 and 10% for DB3. The difference between *Mdl$_1$* and *Weight PoF* models in terms of F-measure is 44% on average for DB1, 22% for DB2 and 92% for DB3, while the difference between *EPF* and *Weight PoF* models in terms of F-measure is 29% on average for DB1, 16% for DB2 and 87% for DB3. Considering top-k=4, the difference between *Mdl$_1$* and *Filter PoF* models in terms of F-Measure is 6% on average for DB1, -38% for DB2 and -25% for DB3, while the difference between *Filter PoF* and *EPF* in terms of F-measure is 14% on average for DB1, 23% for DB2 and 17% for DB3. The difference between *Mdl$_1$* and *EPF* is 19% on average for DB1, 2% for DB2 and -3% (it means that *EPF* is better than *Mdl$_1$*) for DB3. Finally, The difference between *Mdl$_1$* and *Weight PoF* models in terms of F-measure is 39% on average for DB1, 16% for DB2 and 88% for DB3, while the difference between *EPF* and *Weight PoF* models in terms of F-measure is 23% on average for DB1, 14% for DB2 and 86% for DB3.

Table 1 F-measure gains (reductions) across recommender systems (RS on the row vs. RS on the column) for DB1, DB2 and DB3.

		Un-contextual	EPF	Filter PoF	Weight PoF	Mdl$_1$
DB1	Un-contextual	0%	-2%	-20%	27%	-22%
	EPF	2%	0%	-19%	29%	-21%
	Filter PoF	20%	19%	0%	59%	-3%
	Weight PoF	-27%	-29%	-59%	0%	-39%
	Mdl$_1$	22%	21%	3%	39%	0%
DB2	Un-contextual	0%	-2%	-27%	14%	-7%
	EPF	2%	0%	-26%	16%	-5%
	Filter PoF	27%	26%	0%	57%	28%
	Weight PoF	-14%	-16%	-57%	0%	-18%
	Mdl$_1$	7%	5%	-28%	18%	0%
DB3	Un-contextual	0%	-88%	-90%	8%	-88%
	EPF	88%	0%	-12%	819%	5%
	Filter PoF	90%	12%	0%	942%	19%
	Weight PoF	-8%	-819%	-942%	0%	-89%
	Mdl$_1$	88%	-5%	-19%	89%	0%

Table 1 reports all the accuracy gains (in terms of F-measure) across each recommender systems for DB1, DB2 and DB3 (negative values mean performance reduction) using a "find all good items" strategy. For example, its first row shows the performance gains (reductions), in terms of F-measure, for the un-contextual RS vis-à-vis the EPF, Filter PoF, Weight PoF and Mdl1 methods. As you can see, the matrix in Table 1 is anti-symmetric, as should be the case when two methods are compared in terms of their relative performance. Table 1, accordingly with the results discussed in previous sections, shows that *Weight PoF* is the worst recommender system (it never outperforms any other RS) followed by the un-contextual one (it outperforms only the *Weight PoF* while it is dominated by all the other context-aware RSs). On the other side, the *Filter PoF* is the best recommender system (it outperforms all the other systems and it is dominated by the Mdl_1 only for the DB1) followed by the Mdl_1 (it generally outperforms all the other RSs with the exception of *Filter PoF*). The *EPF* always outperforms the un-contextual recommender system, while in some cases dominates the other context-aware RSs and in other cases is dominated by them.

In summary, the answer to the question of which contextual approach (pre-filtering, post-filtering, or contextual modeling) is better depends on the type of the post-filtering method used. In fact, although the *Filter PoF* does not outperform all the other approaches in every setting, it does in most settings, while the *Weight PoF* is often outperformed by all the other context-based approaches. Overall, the *Filter PoF* is the best approach in 78% of comparisons in terms of F-measure (i.e., seven times out of 9), 67% in terms of Precision, 100% in terms of *MAE* and *RMSE* (i.e., three times out of three). In terms of F-measure, the CM approach (Mdl_1) is the best in 44% of comparisons (percentage values do not sum to 100% because in some cases approaches are equivalent), while *EPF* only 11%. The alternative post-filtering method, *Weight PoF*, has never the highest performance in terms of F-measure, only in one case it has the highest Precision (when top-k=1 for DB2, Fig. 13(b)), but in this case all approaches are equivalent, and in four cases *Weight PoF* has the highest Recall, but only in two cases the difference are relevant in some contexts (Fig. 12(a, b)). In contrast, *Weight PoF* has the lowest F-measure and Precision in 78% of cases, the lowest

Recall in 33% of comparisons, and it is always the worst in terms of *MAE* and Recall.

Although we demonstrated that the post-filtering method *Filter PoF* dominates alternative contextual approaches considered in the paper in terms of better recommendation performance, it does not mean that it should always be used in practice for the following reasons. First, using *Filter PoF* entails finding the right parameters for the method, such as the size of the neighborhood *N* and the threshold value *P**. Determining good values of these parameters in a specific application can be time consuming and expensive. Also, different post-filtering methods, such as *Weight* vs. *Filter*, have different types of parameters, and this complicates the selection of the best post-filtering method even further. Finally, post-filtering methods are computationally more expensive than the pre-filtering method *EPF* because only a relatively small subset of data is used in estimations of unknown ratings for *EPF*.

Therefore, the selection of the best contextual modeling method in a given application is more complicated in practice than a simple rule "always use *Filter PoF* in all the settings." Therefore, it is important to provide some practical guidelines about which method should be used and in which circumstances. In the next section, we present such guidelines.

7 An Alternative Approach to Selecting Context-Based Methods

Based on the results reported in Sections 5 and 6, we propose the following alternative approach to selecting when to use which contextual approach that is less extensive and time consuming than the direct comparison method reported in Section 6. First of all, before doing any comparison, one needs to check if context matters. If it does not, i.e. there are no statistically significant differences across the ratings associated with different contexts, then the whole comparison becomes meaningless. As explained in Section 4, we used a pairwise t-test in order to detect such statistical differences. If context does matter (e.g., statistically significant differences exist), then we propose to proceed as follows.

First of all, we compare the exact pre-filtering (*EPF*) and the un-contextual methods. If the un-contextual method outperforms pre-filtering, then we do not

consider pre-filtering at all (since it is inferior) and there is a choice between the following two alternatives:

1. Use a contextual modeling method to achieve the performance results that are "reasonable" in the following sense: (a) the contextual modeling is better than the un-contextual method in almost all the cases, (b) it is a much simpler and cheaper solution (no huge performance variations among similar CM methods) that avoids an expensive and uncertain search for a good post-filtering method.
2. Identify the best-performing post-filtering method and use it. This involves identification of the right method (such as *Filter PoF*) and selecting the right values of parameters for it. This approach is necessary if there is a need to achieve the best recommendation performance at any cost.

If, on the other hand, pre-filtering dominates the un-contextual case [26], then there is a choice between the following two alternatives:

1. Use pre-filtering to achieve the performance results that are "reasonable" in the following sense: (a) pre-filtering is better than the un-contextual case, as stated above; therefore, it already constitutes an improvement over the baseline, and (b) it is a much simpler and cheaper solution that avoids an expensive and uncertain search for a good post-filtering method. Also, as shown in Section 5, the CM and *EPF* methods are comparable in terms of their performance and, therefore, the CM methods can be used in addition to the *EPF* approach. However, since *EPF* is computationally less expensive than CM, we suggest using *EPF* as the first choice.
2. Identify the best-performing post-filtering and use it. This identification step is necessary because only the best post-filtering method may outperform the pre-filtering approach. As our results demonstrate, only the best post-filtering methods (such as *Filter PoF*) outperform the pre-filtering and modeling approaches, and one needs to identify such methods from the whole range of post-filtering approaches. Needless to say, this is an expensive and uncertain endeavor, because it is not guaranteed that such a better-performing post-filtering method can be found in a particular application. In our experiments, the performance gain of a good post-filtering method (such as *Filter PoF*) vis-à-vis a bad method (such as *Weight PoF*) can be up to 70% (e.g., F-measure of *Weight PoF* vs. *Filter PoF* for the context "Wearing apparel" in Fig. 14(c). This example demonstrates again

the importance of selecting a good post-filtering method, which can be a time-consuming and an expensive task.

The tradeoff between using the pre-filtering vs. the post-filtering vs. contextual modeling option lies in the "cheapness" of the pre-filtering method vis-à-vis potentially better performance of the post-filtering that comes at a price of significantly higher costs and the risks associated with choosing the right method with the properly chosen parameters among different post-filtering approaches that have large performance variations among them (again, better performing post-filtering method may not be found in a given application). This is in contrast to the low performance variations among different CM approaches of the same type (such as neighborhood type *Mdl* methods).

However, regardless whether the post-filtering or the pre-filtering or the contextual modeling method is used, the overall approach described in this section combines "the best of both worlds" in the sense that it provides a clear guidance of when to select the pre-filtering, the post-filtering and the contextual modeling methods in a way that is less expensive than the direct comparison method described in Section 6. Therefore, we recommend using this approach in practice.

8 Conclusions

In this paper, we compared the un-contextual and contextual recommender systems, for which we considered the pre-filtering, the post-filtering and the contextual modeling methods of generating contextual recommendations. In particular, we used the exact pre-filtering (*EPF*) and the *Weight* and the *Filter* post-filtering methods for the first two approaches. Moreover, we proposed a new type of contextual modeling, that we called contextual neighbors, and four specific types of contextual neighbors methods, called Mdl_1, Mdl_2, Mdl_3 and Mdl_4, each of them selecting contextual neighborhoods in a somewhat different way.

We compared the un-contextual with the contextual methods across various experimental settings, including three datasets, different level of item aggregation, different neighborhood sizes, seven recommendation engines (un-contextual, *EPF*, *Filter PoF*, *Weight PoF*, Mdl_1, Mdl_2, Mdl_3 and Mdl_4), different contextual levels (C_1 and C_2), two recommendation strategies ("find all good items" and "top-k"), different number of recommended items (k = 1, 2, 3 and 4) and several

performance measures (Precision, Recall, F-Measure, *MAE* and *RMSE*). We showed that the contextual *Filter* method dominates the un-contextual one and that the un-contextual method dominates the *Weight* method. We also showed that *EPF* dominates the un-contextual method in some cases and is inferior in other cases on the datasets (and the corresponding applications) used in our study. We also compared the contextual neighbors methods to identify the best performing one. Although Mdl_1 slightly outperforms the others, we have shown that there are no significant performance differences among them. This result is not surprising because different ways of selecting contextual neighborhood do not fundamentally change recommendation results. We have then selected Mdl_1 as the best-of-breed contextual modeling method and compared it with the pre-, post-filtering and un-contextual methods. We showed that Mdl_1 dominates the traditional un-contextual approach and is comparable to the pre-filtering method (*EPF*). We have also shown that Mdl_1 dominates some of the less advanced post-filtering methods (such as *Weight PoF*) but is inferior to the best post-filtering methods (such as *Filter PoF*).

This implies, among other things, that, in order to decide which approach should be used in a particular recommendation application, various post-filtering methods should be carefully compared which is a laborious and a time-consuming strategy. To overcome this problem, we proposed a more practical and effective way to decide which contextual approach should be deployed in a specific application. We propose, first, to compare the pre-filtering with the un-contextual filtering methods. If the un-contextual case dominates pre-filtering, then one should either choose the contextual modeling as a reasonable solution, or try to identify the best-performing post-filtering method as a more expensive but a potentially better alternative. Otherwise, one should either choose the pre-filtering as a "quick" and reasonable solution, or try to identify the best-performing post-filtering method as a more expensive but a potentially better alternative.

References

1. Bettman, J., Luce, M., & Payne, J. (1998). Constructive Consumer Choice Processes. *Journal of Consumer Research, 25*(3), 187-217.

2. Palmisano, C., Tuzhilin, A., & Gorgoglione, M. (2008). Using Context to Improve Predictive Modeling of Customers in Personalization Applications. *IEEE Transactions on Knowledge and Data Engineering, 20*(11), 1535-1549.

3. Bettman, J. R., Luce, M. F., & Payne, J. W. (1991). *Consumer Decision Making: A Constructive Perspective* (Consumer Behavior and Decision Making).

4. Lussier, J. G., & Olshavsky, R. W. (1979). Task Complexity and Contingent Processing in Brand Choice. *Journal of Consumer Research, 6*(2), 154-165.

5. Lilien, G. L., Kotler, P., & Moorthy, S. K. (1992). *Marketing Models*: Prentice Hall.

6. Cena, F., Console, L., Gena, C., Goy, A., Levi, G., Modeo, S., et al. (2006). Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Commun., 19*(4), 369-384.

7. van Setten, M., Pokraev, S., & Koolwaaij, J. (2004). Context-Aware Recommendations in the Mobile Tourist Application COMPASS. In *Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 235-244).

8. Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *ACM Transactions on Information Systems, 23*, 103-145.

9. Adomavicius, G., & Tuzhilin, A. (2001). Multidimensional Recommender Systems: A Data Warehousing Approach. In *Electronic Commerce* (pp. 180-192).

10. Anand, S., Mobasher, B., Berendt, B., Hotho, A., Mladenic, D., & Semeraro, G. (2007). Contextual Recommendation. In *From Web to Social Web: Discovering and Deploying User and Content Profiles* (Vol. 4737, pp. 142-160): Springer Berlin / Heidelberg.

11. Oku, K., Nakajima, S., Miyazaki, J., & Uemura, S. Context-Aware SVM for Context-Dependent Information Recommendation. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06), 2006*: IEEE Computer Society

12. Yu, Z., Zhou, X., Zhang, D., Chin, C.-Y., Wang, X., & Men, J. (2006). Supporting Context-Aware Media Recommendations for Smart Phones. *Pervasive Computing, IEEE, 5*(3), 68-75.

13. Herlocker, J., & Konstan, J. (2001). Content-Independent Task-Focused Recommendation. *IEEE Internet Computing, 5*(6), 40-47.

14. Adomavicius, G., & Tuzhilin, A. (2010). Context-Aware Recommender Systems. In Springer (Ed.), *Handbook on Recommender Systems*.

15. Chen, L. S., Hsu, F. H., Chen, M. C., & Hsu, Y. C. (2008). Developing recommender systems with the consideration of product profitability for sellers. *Information Sciences, 178*(4), 1032-1048.

16. Huang, Z., Chung, W., & Chen, H. A graph model for e-commerce recommender systems. In *Journal of the American Society for Information Science and Technology, 2004* (Vol. 55, pp. 259-274, Vol. 3)

17. Huang, Z., Li, X., & Chen, H. Link prediction approach to collaborative filtering. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries, 2005* (pp. 141-142): ACM Press

18. Kwon, O., & Kim, J. (2009). Concept lattices for visualizing and generating user profiles for context-aware service recommendations. *Expert Systems with Applications, 36*(2), 1893-1902.

19. Resnick, P., & Varian, H. R. (1997). Recommender systems. *Commun. ACM, 40*(3), 56-58, doi:10.1145/245108.245121.

20. Breese, J., Heckerman, D., & Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence 1998* (Vol. 461, pp. 43-52)

21. Kachigan, S. K. (1986). *Statistical Analysis*: Radius Press.

22. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems, 22*(1), 5-53, doi:10.1145/963770.963772.

23. Nichols, D. M. Implicit rating and filtering. In *Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering, 1998* (pp. 31-36)

24. Oard, D., & Kim, J. Implicit Feedback for Recommender Systems In *Proceedings of the AAAI Workshop on Recommender Systems, 1998* (pp. 81-83)

25. Adomavicius, G., Huang, Z., & Tuzhilin, A. (2008). Personalization and Recommender Systems. In Z.-L. C. a. S. Raghavan (Ed.), *State-of-the-Art Decision Making Tools in the Information-Intensive Age* (pp. 55-100): Tutorials in Operations Research.

26. Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., & Pedone, A. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems, New York, New York, USA, 2009* (pp. 265-268): ACM