OLAP Queries Context-Aware Recommender System

Elsa Negre[1(✉)], Franck Ravat[2], and Olivier Teste[2]

[1] Paris-Dauphine University, PSL Research Universities, CNRS UMR 7243, LAMSADE, Paris, France
elsa.negre@dauphine.fr
[2] Université de Toulouse, IRIT, CNRS UMR 5505, Toulouse, France
{ravat,teste}@irit.fr

Abstract. It becomes hard and tedious to easily obtain relevant deci-sional data in large data warehouses. In order to ease user exploration during on-line analytical processing analysis, recommender systems are developed. However some recommendations can be inappropriate (irrele-vant queries or non-computable queries). To overcome these mismatches, we propose to integrate contextual data into the recommender system. In this paper, we provide (i) an indicator of obsolescence for OLAP queries and (ii) a context-aware recommender system based on a contextual post-filtering for OLAP queries.

1 Introduction

A substantial effort of the scientific community in the last decades has been to develop data warehousing and on-line analytical processing (OLAP) [1]. Data warehouses (DWs) are built using materialized views [2] based on the multidimensional model, which consists in describing data as a data cube [3]. Contemporary DWs form large multidimensional networks where it becomes hard and tedious to explore and easily obtain relevant decisional data [4].

Context. One possible opportunity is to extend decision support systems with recommender system approaches [5]. The recommendation approaches are popular to provide personalized results into large volumes of data [6]. However, there is few research in the field of OLAP query recommendation [7,8].

Paper Issue. Although these approaches allow recommending queries for a given user, the recommended queries may be not satisfactory because the solution space is limited to queries that existed in the past. The *problem of out of date recommended queries* is the dissatisfaction of the recommender system users. We identified two reasons for the dissatisfaction:

– Irrelevant queries for users, which are not used for long time. These old queries may be considered out of date by users.

https://doi.org/10.1007/978-3-319-98812-2_9

– Non-computable queries due to data updating: stored data are periodically refreshed, and possibly updated.

Motivating Example. The illustrating example concerns a product company. Users analyze quantities of sales of products according to time and stores. To support this analysis, they query the DW (Fig. 1) where the analysis subject is *Quantity* and the three available analysis axes are *Store*, *Product* and *Time*. Each of them is organized according to a hierarchy of attributes, which represent various granularities. OLAP analysis consists in applying different sessions of queries on the schema defined above. After applying a user query, a recommender system proposes different possible queries that can help the user in his/her analysis.

Given a user query q_c and a set of past queries Ω that the recommender system kept, recommending queries aims to provide a subset $Q_R \subseteq \Omega$ similar to q_c. $Q_R = \{(q_{ri}, \sigma_i) \mid \sigma_i \in [0..1]$ is a similarity value between q_{ri} and $q_c\}$ is an ordered set of recommended queries q_{ri} ranked to the more similar to the less similar regarding to q_c.

Suppose that, today, a user wants to know the quantity of beverages of the range 'Fanta' sold in New York City and San Francisco in 2016 via the query q_c such that: q_c = "Sales of Fanta in San Francisco and New York City in 2016". OLAP queries that can complete the analysis. To suggest additional queries, the OLAP recommender system (RS) calculates similar queries denoted Q_R from the set of stored past queries denoted Ω (previous launched queries between 2005 and 2017). The recommendations returned by the system may be $Q_R = \{(q_{r1}, 0.95), (q_{r2}, 0.94), (q_{r3}, 0.92)\}$ where:

– q_{r1} = "Sales of Cola in New York City in 2012"
– q_{r2} = "Profits related to Fanta in New York City and San Francisco in 2016"
– q_{r3} = "Sales of Fanta in the store M1 of New York City"

Due to periodically refreshing, the DW only contains now products sold in US from 2012 to 2017. In classical RS, the stored queries is kept with non-calculable queries or it is cleaned by deleting out of date queries.

– q_{r1} is out of date because the query handles data related to 2012, which are out of date for the user who focuses on 2016.
– q_{r3} is out of date because the store M1 of New York City is closed since 2011. The M1 data no longer exist in the current DW.

Only $Q'_R = \{(q_{r2}, 0.94)\}$ is usable. Interesting queries, e.g. $Q_R \backslash Q'_R$ are lost because part of data is not kept, and then it is not possible to calculate these out of date queries. Here, q_c is launched in 2017 and q_{r1} could be really interesting for data related to 2016. In the same way, q_{r3} could be interesting for data related to 2016 and either for another store(s), or cities. To overcome these drawbacks, we intend to define an OLAP context-aware RS allowing the recommendation of some past queries such as q_{r1} and q_{r3} that are out of date and/or non calculable into classical RS. We want to extend classic RS where recommended queries are extended with contextual data. In the example, q_{r1} and

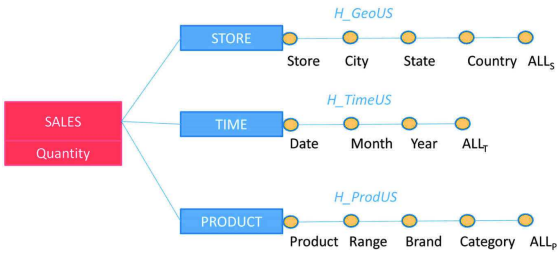

Fig. 1. Example of DW star schema

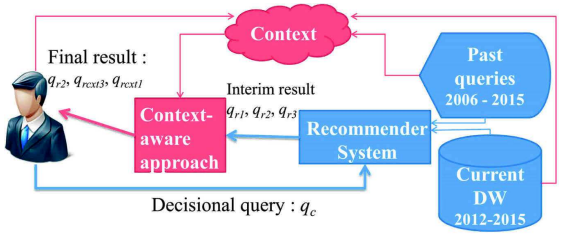

Fig. 2. Our context-aware approach

q_{r3} could be reformulated. The RS may provide a set of contextualized queries $Q_{cxt} = \{(q_{cxt1}, 0.99), (q_{cxt3}, 0.96)\}$ where:

- q_{cxt1} = "Sales of Cola in New York City in 2016".
- q_{cxt3} = "Sales of Fanta in all stores of New York City".

Finally, given a user query q_c, and a past queries set Ω, our approach aims at calculating $Q'_R \cup Q_{cxt} = \{ (q_{cxt1}, 0.99), (q_{cxt3}, 0.96), (q_{r2}, 0.94) \}$, where Q'_R is the set of "usable" recommended queries and Q_{cxt} is a set of out of date queries that are contextualized to be relevant regard to q_c. Figure 2 illustrates the approach to extend the set of recommended queries by reformulating some out of date queries (instead of delete these queries).

Paper Contributions and Organization. In this paper, we propose a context-aware recommender system based on a contextual post-filtering for OLAP queries where we contextualize queries recommended by a classic log-based recommender system. The paper is organized as follows: Sect. 2 presents the related work on OLAP recommender systems and Context-aware recommender systems (CARSs). Section 3 details our indicator of obsolescence and our OLAP queries CARS. Section 4 presents some experiments and results.

2 Related Work

In this section we present some related work about, OLAP recommender systems (RSs) and Context-aware recommender systems (CARSs).

OLAP Recommender Systems. RSs are a particular form of information filtering designed to present information items (e.g., movies, books, ...) that may interest the user. RSs have been studied in many fields, including cognitive science, information retrieval, web and e-commerce. However, some works have focused on recommendations in the field of data warehouses analyzed by OLAP queries and proposed methods and algorithms to assist the user in her/his querying process by proposing relevant queries (items). Among these, some focused on exploiting user profiles and preferences [9], and others focused on the discoveries made during analyses [10] as well as on exploiting logs containing sequences of queries previously run by other users on the same cube [11].

The quality of RSs may be measured by prediction/classification/ranking accuracy, user coverage and recommendation diversity (see [12] for further details). To the best of our knowledge, there does not exist any measure or indicator verifying if the recommendations returned by a RS are out of date.

Context-Aware Recommender Systems. The probably most widely accepted definitions of context is the one of [13] where: "Context is any information that can be used to characterize the situation of an entity". A key accessor to the context in any context-aware system is a well designed model. Some context modeling approaches exist [14,15]. Another complementary approach is to incorporate context. [16] explains that CARSs "generate more relevant recommendations by adapting them to the specific contextual situation of the user". Traditionally, the problem of recommendation can be summarized as the problem of estimating scores for items that have not been seen by a user, i.e. as a prediction problem where the RS predicts the user's ratings for a given item according to a user profile, it is a rating function: $r_{RS} : Users \times Items \rightarrow Ratings$ [16]. With context, the rating function for CARS becomes: $r_{CARS} : Users \times Items \times Contexts \rightarrow Ratings$ [16]. Furthermore, the context can be incorporated in various stages of the recommendation process: (i) pre-filtering, (ii) post-filtering, and (iii) contextual modeling [16]. Finally, [17] concludes that post-filtering seems to be the more efficient.

To the best of our knowledge, there exists no CARS in OLAP field. So, in this paper, our goal is to propose a CARS based on a contextual post-filtering for OLAP queries.

3 OLAP Queries Context-Aware Recommender System

3.1 Definitions

A N-dimensional *Cube* C has for schema $C = \langle D_1, ..., D_N, F \rangle$ where: (i) For $i \in [1, N]$, D_i is a dimension, (ii) F is a fact containing the set of measures.

For a *dimension* D (of the cube C), having for schema an ordered list of m attributes $\{L_j\}$ ($\forall j \in [1, m]$, where m is the depth of the dimension). L_1 is the lowest granularity attribute. Each attribute L_j of the hierarchy is the child of a single parent present at the level of granularity immediately higher L_{j+1} of the hierarchy. In the following, $ADOM(D) = \bigcup_{j=1}^{m} adom(L_j)$, where $adom(L_j)$ is the set of existing values of L_j on C and $ADOM(D)$ is the set of all existing attribute values of dimension D (on C).

Given a cube $C = \langle D_1, ..., D_N, F \rangle$, a *multidimensional query* q_M on C can be represented as $q_M = \langle q_1, ...q_N \rangle$ where $\forall i \in [1, N]$, q_i is a relational query (to obtain the values of the attributes of each dimension D_i). Thus, $q_M = q_1(D_1) \times ... \times q_N(D_N) = \times_{i=1}^{N} q_i(D_i)$.

3.2 ObsoIndic

As displayed in the previous section, there does not exist any indicator verifying if recommended OLAP queries returned by a RS are out of data of use and/or missing data. First of all, we give the definitions of some notations.

– Q_R is the set of all possible recommended queries
– Q'_R is the set of computable recommended queries ($Q'_R \subset Q_R$)

So, we define a new indicator for OLAP queries log-based RS, *ObsoIndic* that measures the number of recommended queries that are obsolete of use (i.e. "old"/irrelevant) or obsolete of data (i.e. non-computable) in relation to the total number of recommended queries such that:

$$ObsoIndic = \frac{|Q_R| - |Q'_R|}{|Q_R|}$$

where $ObsoIndic \in [0; 1]$ and the lower the value, the higher the non-obsolescence of the recommendations.

For example, according to the motivating example (Sect. 1), the log-based RS returns three recommendations (Q_R): q_{r_1}, q_{r_2} and q_{r_3} where q_{r_1} and q_{r_3} are out of date. Our indicator is: $ObsoIndic = \frac{3-1}{3} \approx 0.666$: more than 66% of the recommended queries are irrelevant or non-computable. So this RS needs to enhance its process by integrating our context-aware approach.

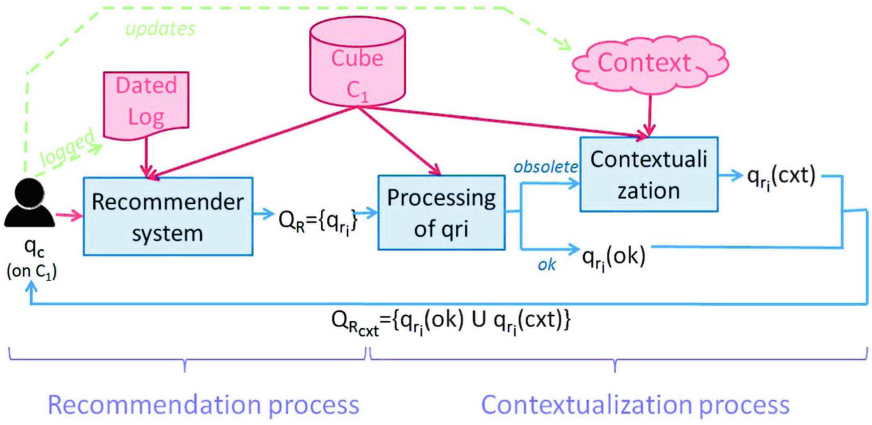

Fig. 3. Our proposition. *Red arrows are inputs and Blue arrows are execution* (Color figure online)

3.3 Global Process

Unlike classic approaches, ours aims at recommending queries that did not exist. Starting from log-based recommended queries, our system exploits additional information to obtain context-aware recommendations (not existing queries). In fact, "old"/irrelevant queries will be updated and non-computable queries will become computable. Our approach mixes decision making process, recommendation process and contextual data to help a decision-maker. We extend a classic RS where returned queries are improved with contextual data. Then, the obtained context-aware recommendations are returned to the decision-maker. Figure 3 displays this additional component. During the recommendation process, the OLAP query, q_c, launched by the decision-maker (i) is logged, (ii) updates the

context and (iii) is the input of the RS. Then, the log-based RS returns a set of recommended queries $Q_R = \{q_{r_i}\}$. Our proposition starts from this point: the contextualization process. Each query q_{r_i} of Q_R is processed. Then:

1. the query q_{r_i} can be processed and is not out of date, in which case, this query is added to the final output $Q_{R_{cxt}}$, or
2. the query q_{r_i} is out of date, in which case, this query does require some updates according to a given context and the new context-aware query q_{cxt_i} is added to the final output $Q_{R_{cxt}}$.

Finally, this set $Q_{R_{cxt}} = \{q_{r_i}\} \cup \{q_{cxt_i}\}$ of context-aware recommendations is returned to the decision-maker.

Specifically, given a current query q_c, and Ω the set of all past queries (query log), the log-based RS proposes a set of recommended queries $Q_R \subseteq \Omega$ that can follow q_c in an analysis session. The set Q_R is composed of queries q_{r_i} ($\forall i \in [1; |\Omega|]$) dated t_i (see Recommendation process part of Fig. 3). Ω can contain relatively old or even obsolete queries. In our approach, to overcome these two cases of obsolescence (because of using only the contents of a log Ω), we want $Q_{R_{cxt}} \subseteq \Delta$ (where Δ represents all the queries that can be launched on the cube C_1 ($\Omega \subset \Delta$)). So when the queries q_{r_i} recommended for q_c are not out of date, we do not act. By cons, when queries q_{r_i} are considered out of date, we want to contextualize them. What we mean by "context-aware queries" is, intuitively, to put the queries "in the style of today". More formally, we have to modify each obsolete query to make them relevant and appropriate to the specific context of the current query q_c (see Contextualization process part of Fig. 3). In more details, during the contextualization process, there are two steps: (i) Processing the q_{r_i} and (ii) Contextualizing the obsolete q_{r_i}.

3.4 Defining/Modeling Context

The context is a set of contextual information [18]. To take fully account of the context of OLAP analysis, we represent it using 5 categories of elements (according to [19]): Time, Individuality, Relations (between users), Activity, Material.

Context modeling is still complicated given the nature of the data and/or contextual information: the model must be able to manage various data sources, their quality and lifetime heterogeneity and their imperfect nature [20]. According to [21], only the ontological model allows a good partial validation of the data and a good formalization of the model. Thus, we use the general ontology of [19] regrouping the five categories.

3.5 Contextualization Process

The contextualization process can be modeled as an algorithm where the inputs are the set Q_R of recommended queries returned by the classic log-based RS (during the recommendation process), the log of queries Ω, the cube on which queries are launched, some contextual data/information and the obsolescence threshold. Concretely, for each recommended query q_{r_i} of Q_R:

– when q_{r_i} is computable and obsolete of use ($isObsoleteU$), the $RecCxtOld$ function updates q_{r_i} into $q_{r_i}(cxt)$ which is added to the final set $Q_{R_{cxt}}$;
– when q_{r_i} is computable and not obsolete of use, q_{r_i} does not require any update and is added to the final set $Q_{R_{cxt}}$;
– when q_{r_i} is not computable, the $RecCxtData$ function updates q_{r_i} into $q_{r_i}(cxt)$ which is added to the final set $Q_{R_{cxt}}$.

Finally, the set $Q_{R_{cxt}}$ of context-aware recommendations is returned by our following Algorithm which complexity is $O(|Q_R| \times t \times k \times |\Omega|^3)$ where $|Q_R|$ is the number of queries in Q_R, t is time to calculate the more complicated query, k is the number of clusters and $|\Omega|$ is the size of the query log Ω.

Algorithm. Context-aware recommended queries: CarsOlap

Require: C_1 the cube on which queries are launched
Q_R : the set of recommended queries returned by the log-based recommender system
$Context$: contextual data/information (ontology)
Ω : the log of queries
$ObsoTh$: the obsolescence threshold
Ensure: $Q_{R_{cxt}}$: the set of context-aware recommended queries
$Q_{R_{cxt}} = \emptyset$
for each q_{r_i} $(\forall i \in |Q_R|)$ **do**
if $q_{r_i} \subset C_1$ (computable on C_1) **then**
if $isObsoleteU(q_{r_i}, \Omega, ObsoTh)$ **then**
Obsolescence of use/irrelevance:
$Q_{R_{cxt}} \leftarrow Q_{R_{cxt}} \cup RecCxtOld(q_{r_i}, Context, \Omega, ObsoTh, C_1)$
else
$Q_{R_{cxt}} \leftarrow Q_{R_{cxt}} \cup q_{r_i}$
end if
else
Obsolescence of data (non-computable):
$Q_{R_{cxt}} \leftarrow Q_{R_{cxt}} \cup RecCxtData(q_{r_i}, Context, C_1)$
end if
end for
Deleting duplications
return $Q_{R_{cxt}}$

The boolean $isObsoleteU(q, \Omega, ObsoTh)$ function returns $true$ when the query q has not been launched in Ω since a certain time ($ObsoTh$ is a threshold, Sect. 4.2 shows how to define its value).

The $RecCxtOld(q, Context, \Omega, ObsoTh, Cube)$ function updates the query q which is obsolete of use with the contextual data $Context$. If the query q contains a selection on some temporal levels, the corresponding values are replaced/upgraded with the TIME category of the $Context$, by keeping the time span between the time the query q was launched and time selections. If the query q does not contains temporal selections, the k-medoid clustering algorithm is used to partition the more recent queries of the log Ω (according to the threshold $ObsoTh$). Each cluster cl_j is represented by a specific query : q^j_{med}. Then, the query q is replaced by the more similar q^j_{med}. Only if the obtained context-aware query is computable on $Cube$ and can be displayed[1] (according to the MATERIAL category of the $Context$), it is returned by the system.

[1] Note that the obtained context-aware query may be update by rolling-up, level by level, until the query can be displayed (according to the MATERIAL category) .

The $RecCxtData(q, Context, Cube)$ function updates the query q which is non-computable on $Cube$, with the contextual data $Context$. If it is a schema problem, i.e. some levels do not exist or are not accessible, and the corresponding dimension exists, then, the query q rolls-up to the 'ALL' level of this dimension. If it is an other schema problem, i.e. some dimensions do not exist or are not accessible, then this dimension is removed from q and some constraints are added through the INDIVIDUALITY category of the $Context$. If it is a data problem, i.e. some values of levels do not exist or are not accessible, and the corresponding level exists, then, all the existing values of the level are displayed and some constraints are adding through the INDIVIDUALITY or TIME categories. Only if the obtained context-aware query is computable on $Cube$ and can be displayed (see footnote 1) (according to the MATERIAL category), it is returned by the system.

4 Experiments

We present the results of the experiments we have conducted to assess the capabilities of our framework. We used synthetic data produced with our own data generator [11]. Both our prototype and our generator are developed in Java using JRE 1.6.0-27 with Postgres 9.1.10 and Mondrian 3.3.0.14703. All tests are conducted with a Core i5-2520M (2.5 Ghz $\times$ 4) with 8 GB of RAM using Linux Ubuntu 12.04.

4.1 Data Set

The process of synthetic log generation is detailed in [11]. Our experiments are conducted with the log-based RS prototype: RecoOLAP [22].

4.2 Results

Obsolescence Threshold. Obsolescence and freshness were studied in the case of OLAP queries [23]. Here, we attempt to experimentally determine the threshold value of obsolescence of a query log. We make 10 successive 10-fold

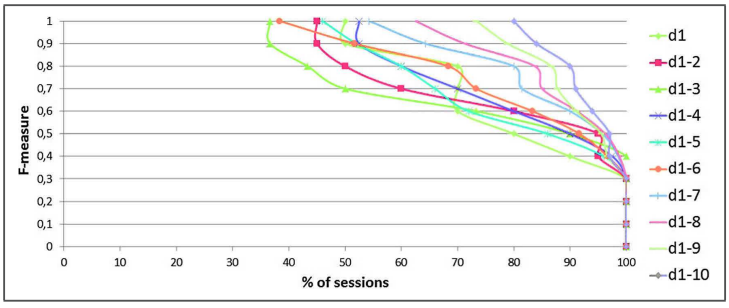

Fig. 4. F-Measure of the recommendations for the 10 validations

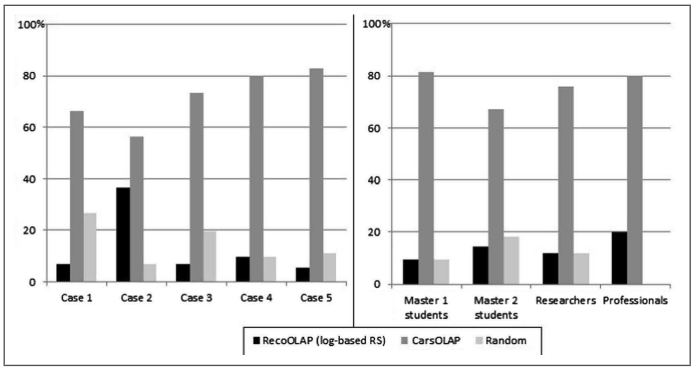

Fig. 5. User feedback analysis (% of pertinent recommendations, left: for each of the 5 cases and right: for different status of responders).

cross validation. First, the generated set of sessions is partitioned, according to the seniority of the sessions in the log, in 10 equally sized subsets, i.e. 10 deciles. We then make a 10-fold cross validation with the 10 deciles. For each such session s_c of size n, we use the sequence of the first $n-1$ queries as the current session, and we compute the recommendations for the n-th query. The generated log contains 610 queries (100 sessions).

Figure 4 shows the inverse cumulative frequency distribution of the recorded F-measure for the 10 validations (d1-10 is the log with 100 sessions, d1-9 contains the 9 more recent deciles, 90 sessions, ... and d1 contains the more recent decile, 10 sessions). This experiment allows us to tune our system in order to choose for the obsolescence threshold the value that achieves best F-measure. First, note that these good results can be explained by the density of the log generated. Second, as many machine learning systems, the more logs are large, the better the quality. So it is normal that $d1-10$ obtains the best results. In the case of $d1$, the log is small but provides a relatively good quality (over 70% of sessions have a F-measure ≥ 0.8). From $d1-2$ to $d1-5$, less than 60% of sessions reach a F-measure ≥ 0.8 (which is not satisfactory). Starting from $d1-6$, results are correct (and close to those of $d1$), we can conclude that we must keep at least the 6 first deciles. Finally, the obsolescence threshold can be defined as $\frac{6}{10}$, e.g. if the log sessions spread over 10 years, sessions/queries over 6 years are obsolete.

User Feedback. User feedback is weakly used to evaluate RSs, due to the difficulty of setting up a protocol and/or the one to involve users (who find the task time-consuming), but a user feedback evaluation allows to position the user at the heart of the evaluation knowing that the vocation of the RS is to satisfy the user. In order to test the quality of the context-aware recommendations, we established a protocol of tests conducted with 72 real users[2] on the FoodMart dataset[3]. Starting from a session of queries (where goal is explained), users are

[2] Students, researchers, professionals from different French universities and enterprises - Master 1 students: 21%, Master 2 st.: 61%, researchers: 15% and professionals: 3%.

[3] http://mondrian.pentaho.org.

offered 3 recommendations (note that users do not know which system returned which recommendation) from (i) the log-based RS (RecoOLAP), (ii) our proposition, *CarsOLAP*, (iii) taken at random from the log. Users should choose which recommendation seems the most relevant for each of the 5 cases[4]. Figure 5 shows that whatever the case and whatever the responder's status, CarsOLAP's recommendations are always the most pertinent (more than 50% even more than 80%). Notice that professionals never chose a recommendation obtained by random.

According to this feedback analysis, it seems that recommendations obtained with our proposition, *CarsOLAP*, a CARS based on a contextual post-filtering, are more relevant than recommendations obtained with a classic log-based RS.

5 Conclusion

In this paper, we exposed the limits of classic log-based RSs when recommending OLAP queries. Indeed, recommended queries can be irrelevant or non-computable. In order to overcome these limitations, we propose (i) an indicator of obsolescence and (ii) to enhance these systems (especially when our indicator value is poor), a process for contextualizing recommended queries and develop a CARS. In OLAP area, we model the concept of Context as an ontology including five categories of elements: Individuality, Activity, Time, Relations and Material. Our contextual post-filtering approach couples a classic recommendation process and a contextualization process where recommended queries returned by classic RS are contextualized to obtain context-aware recommended queries. This allows the system to recommend more relevant and context-aware recommendations.

As future work, we intend to perform experiments on real datasets and scale them up. Some experiments will be perform on more or less obsolete datasets to compare results and, obtain user feedbacks to compare the recommendation quality. Our CARS should be applicable both to detailed as well as aggregated data. Our system must be able to choose whether the query runs or not on detailed data or on aggregates. In the same way, our system should remove irrelevant aggregates and/or create new ones. Finally, other approaches can be proposed to modeling and integrating context into CARSs, taking into account for example, the multidimensional nature of context. Furthermore, we hope that RSs will become more than context-aware RSs and perhaps even context-driven RSs.

References

1. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
2. Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., Zhuge, Y.: The stanford data warehousing project. IEEE Data Eng. Bull. **18**(2), 41–48 (1995)

[4] The questionnaire (in French) is available online: http://www.lamsade.dauphine.fr/~negre/questionnaire.html.

3. Gray, J., et al.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. DMKD **1**(1), 29–53 (1997)
4. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: SIGMOD 2011, New York. ACM, pp. 853–864 (2011)
5. Bobadilla, J., Ortega, F., Hernando, A., GutiéRrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
6. Koutrika, G., Bercovitz, B., Garcia-Molina, H.: Flexrecs: expressing and combining flexible recommendations. In: SIGMOD 2009, New York, pp. 745–758. ACM (2009)
7. Bimonte, S., Negre, E.: Evaluation of user satisfaction with OLAP recommender systems: an application to recoolap on a agricultural energetic consumption datawarehouse. IJBIS **21**(1), 117–136 (2016)
8. Ravat, F., Teste, O.: Personalization and OLAP databases. In: New Trends in Data Warehousing and Data Analysis, pp. 1–22 (2009)
9. Golfarelli, M., Rizzi, S., Biondi, P.: myOLAP: an approach to express and evaluate OLAP preferences. IEEE Trans. Knowl. Data Eng. **23**(7), 1050–1064 (2011)
10. Sarawagi, S.: User-adaptive exploration of multidimensional data. In: VLDB, pp. 307–316(2000)
11. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query recommendations for OLAP discovery-driven analysis. IJDWM **7**(2), 1–25 (2011)
12. Negre, E.: Information and Recommender Systems. Advances in Information Systems Set. WILEY-ISTE (2015)
13. Dey, A.K.: Understanding and using context. Pers. Ubiquitous Comput. **5**(1), 4–7 (2001)
14. Strang, T., Popien, C.L.: A context modeling survey. In: UbiComp, Nottingham, pp. 31–41, September 2004
15. Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. **36**(4), 19–26 (2007)
16. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_7
17. Panniello, U., Gorgoglione, M.: Context-aware recommender systems: a comparison of three approaches. In: CEUR Workshop Proceedings, vol. 771 (2011)
18. Zimmermann, A., Lorenz, A., Oppermann, R.: An operational definition of context. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 558–571. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74255-5_42
19. Negre, E.: Pertinent context information for OLAP applications. In: KMIKS 2017 (2017)
20. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: models and approach. Pervasive Mob. Comput. **2**(1), 37–64 (2006)
21. Soualah Alila, F.: CAMLearn: a context-aware mobile learning recommender system. Application to M-Learning Domain. Ph.D. thesis, Dijon, France (2015)
22. Giacometti, A., Marcel, P., Negre, E.: Recommending multidimensional queries. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 453–466. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03730-6_36
23. Röhm, U., Böhm, K., Schek, H.J., Schuldt, H.: Fas: A freshness-sensitive coordination middleware for a cluster of olap components. In: VLDB, pp. 754–765 (2002)