OLAP Personalization and Recommendation

Patrick Marcel

OLAP Personalization and Recommendation

Patrick Marcel

Université François Rabelais Tours, Département Informatique, Laboratoire d'Informatique, patrick.marcel@univ-tours.fr

DEFINITION

Personalizing or recommending OLAP queries aims at making the OLAP user experience less disorientating when navigating huge amounts of multidimensional data (also called cubes). Such approaches allow coping with too many or too few query results, or suggesting new queries to pursue the navigation. Personalization allows adding preferences to a query for filtering out irrelevant results or ranking the results to focus on the most relevant first. It also allows turning selection predicates (hard constraints) into preferences (soft constraints) to favor non-empty answers. On the other end, recommendation allows to leverage the cube instance and/or past navigations on it to complement the current query result.

The general problem can be formally defined by: given a sequence of queries $S=<q_1, \ldots, q_c>$ (a session from now on) over an instance I of a cube schema C, a user profile P (consisting of ordered multidimensional objects), a set of past sessions L (a log from now on), generate a set of one or more queries $Q=\{q^p_1, \ldots q^p_n\}$ such that, typically:

- The queries in Q are sub-queries of q_c (personalization), in the classical sense of query inclusion, or none of the queries of Q are sub-queries of the queries of S (recommendation),
- The queries in Q maximize an interestingness score,

In this definition, S represents the current session, with q_c the last query of this session (the current query).

HISTORICAL BACKGROUND

OLAP Personalization and recommendation approaches are distant descendants of cooperative database [11] techniques aiming at enhancing database management systems with a cooperative behavior. Cooperation can be introduced at the different stages of the retrieval process, which is typically iterative. The purpose of the cooperation includes: helping the user to formulate a query corresponding to an objective and acceptable by the database system, dealing with empty answers or too few results, or suggesting additional information and explaining the query result.

This retrieval process perfectly reflects the activity of OLAP users, who interactively analyze multidimensional data, often without exactly knowing what they are looking for. OLAP queries are normally formulated in the form of sequences called OLAP sessions, by using basic operations to transform one OLAP query into another, so that the new query gives a better understanding of the information retrieved so far. The huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating, and OLAP

sessions mainly include extemporary queries that may easily either return huge volumes of data (if their group-by sets are too fine), or little or even no information.

To facilitate this navigation, discovery driven analysis of OLAP cubes [13, 14] was introduced as the definition of two kinds of advanced OLAP operators to guide the user towards interesting regions of the cube by automatically navigating the cube instance. The first kind tries to explain an unexpected significant difference observed in a query result by either looking for more detailed data contributing to the difference [13], or looking for less detailed data confirming an observed tendency. The second kind proposes to the user unexpected data in the cube based on the data she has already observed, by adapting the Maximum Entropy Principle [14].

It was also observed that past navigations, recorded in a (potentially multi-users) query log, could be used for anticipating the next user query. The works presented in [12] propose to pre-fetch cube data by analyzing the OLAP query log and using it to find the query that is the most likely to appear after the current query of the current session. To this end, past queries are grouped by common projections and selections, and a Markov model represents the OLAP sessions.

Finally, giving to the user the possibility to cope with too many or too few query answers emerged in the database community as preference modeling and query personalization [15]. A first type of approaches extends relational query languages with operators to declare preferences (Preference SQL, Skyline), and an operator to compute dominating tuples (Winnow). Another type of approaches expands regular database queries by incorporating elements from a user profile, usually resulting in another query that is a sub-query of the initial one.

SCIENTIFIC FUNDAMENTALS

Personalization and recommendation approaches can be categorized using the following criteria:

- Proactiveness: this first criterion allows distinguishing between query recommendation (suggesting new queries), which is inherently proactive, and query personalization (changing the current query q_c or post processing its results).
- The type of information used to generate Q: the approach may use all or a subset of the set of parameters (the current session S, the instance I, the cube schema C, the profile P, the log L). In particular, we distinguish current-state approaches, exploiting the content and schema of the current query result and database instance, from history-based approaches, exploiting the query log. We call collaborative approaches those approaches leveraging a multiuser log. Notably, queries can be treated either as simple expressions in a formal language, or as the results of the partial or full evaluation of these expressions over the instance I. The full evaluation of an OLAP query is the set of facts (tuples) retrieved by evaluating the query over the instance. The partial evaluation of an OLAP query is defined as the set of references (i.e., positions in a data cube) to be retrieved from the cube to answer the query, which requires only the instances of the dimensions.
- Prescriptiveness: prescriptive approaches use profile elements as hard constraints that are added to a query (typically q_c) while non prescriptive approaches use them as soft ones: tuples that satisfy as much profile criteria as possible are returned even if no tuples satisfies all of them.

Non-proactive approaches are based on the two types of personalization approaches found in the database literature that essentially differ in terms of prescriptiveness. The first approach [5] borrows from the query expansion paradigm, where an OLAP query is transformed into another query by rewriting the former to incorporate elements of the profile, while the second work [9] is inspired by the use of explicit preference constructors for expressing complex preferences directly within the query, à la Preference SQL.

The approach proposed in [5] defines the user profile P as a set of preferences over multidimensional objects and a visualization constraint. The preferences are defined as orders over dimensions, and for each dimension, an order over members (instance of the dimensions at various level of details). These preferences allow defining an order over the set of partially evaluated queries that can be expressed over the instance I. The visualization constraint is defined as an anti-monotone Boolean function over the set of partially evaluated queries. It can for instance be used to indicate the maximum number of references for displaying the query answer. The personalization $Q=\{q^p_1\}$ of query q_c consists of prescriptively expanding q_c with preferences of P, guaranteeing that (i) q^p_1 is included in q_c, (ii) q^p_1 only fetches preferred facts with respect to P, and (iii) q^p_1 respects the visualization constraint. q^p_1 is generated by starting from the set of most preferred references and iteratively adding less preferred references while the visualization constraint is satisfied.

The work of [9] proposes that elements of the profile P are written with each query. It introduces an algebra to annotate OLAP queries with preference expressions, for defining a strict partial order on the instance I. The algebra consists of a set of base constructors on attributes, measures and hierarchies, composed by the Pareto (giving the same importance to two base preferences) or prioritization (giving priority to one of the base preferences) operators. This allows defining preferences on the schema, i.e., on the space of hierarchies, which are used to induce preferences on the space of data, thereby allowing defining preferences over group-by sets (aggregated data). A specific implementation is developed for evaluating preference queries expressed in this language, in order to calculate the personalized query q^p_1, without having to compute all the aggregations. In [2], it is proposed that preference constructors are automatically added to a current query q_c by mining a query log to identify which preferences could fit q_c.

If they also build upon the previous works (especially [12, 13, 14]), proactive approaches are more diverse than non-proactive ones. They range from current-state to collaborative, with a combination thereof; they can be similarity-based, preference-based or stochastic, vary in how they treat sessions and queries, and in how they generate recommendations.

A current-state, preference-based approach is proposed in [10], with a principle similar to that described in [5], the main differences being that the recommendation q^p_1 is usually not a sub-query of q_c, and that q_c is fully evaluated. q^p_1 is derived from q_c using elements extracted from the user profile P that consists of a set of preference predicates, each with a score of interest. The best preferences (according to the interest score) that are consistent with q_c are incorporated to it, resulting in q^p_1.

The work described in [8] is both collaborative and current-state, and uses the operators introduced in [13] to discover in L the (fully evaluated) queries that investigated the same phenomenon as the one shown by q_c. Sessions are associated with a goal, and recommendations

are those queries from former sessions having the same goal as that of the current session. The approach is composed of two steps. In an offline step, a multiuser query log L is processed to detect discovery driven analysis sessions, i.e., sessions investigating (either by rolling up or drilling down) a pair of facts that show a significant difference (like e.g., a drop of sales from one year to the following year). Those pairs are then arranged into a specialization relation based on the cube hierarchies. A goal is created for each most general pair recording the pair and its descendants, together with the queries that contain them. In the second step, at query time, if q_c investigates a pair that is a descendant of a pair discovered in L, then the set Q of queries associated with the corresponding goal is recommended. The main difference with the approach of [13] is that only L and not I is searched for interesting data.

Another two-steps approach is described in [4], where queries are recommended using a probabilistic model of former sessions, inspired by that of [12]. In an offline step, the former queries of L are grouped with a density-based clustering that uses a similarity measure tailored for the syntax of OLAP queries. The Markov model organizes the query clusters into series of states, with a transition score for each pair of clusters. At query time, the current query q_c is matched with the closest state of the Markov model, in the sense of the average similarity between it and each query of the cluster. Then, the most probable state is identified, and the query of this cluster that is the most similar to q_c is recommended.

The work of [6] introduces a generic framework for similarity-based collaborative query recommendations, with a 3-steps approach for generating recommendations. In the first step, the current session S is compared to the sessions of L, to find the ones that are the most similar to S, in the sense of a similarity between sessions. In the second step, candidate recommendations are extracted from the sessions resulting of the first phase. Finally, in the last step, these candidate recommendations are further processed to be presented to the user. [7] instantiates this framework with partially evaluated queries. It introduces an extension of the edit distance to compare sessions, and uses the Hausdorff distance between sets of references to compare partially evaluated queries. These distances enable the definition of a similarity measure for sessions to be used during the first step. In the second step, the last queries of the sessions that are the most similar to S are extracted to form Q. In the last step, these queries are ranked according to how close they are from the current query q_c. Another instance of this framework is described in [1], where only the syntax of queries is considered. The similarity measure between sessions is an extension of the Smith-Waterman alignment algorithm whose goal is to efficiently find the best alignment between subsequences of two given sequences by ignoring their non-matching parts. This extension uses a query similarity measure tailored for the syntax of OLAP queries that compares the 3 parts of queries (the group-by set, the selection predicates set and the measures set) and averages the result of these comparisons. During the first step, log sessions in L are aligned with S and portions of the most similar log sessions are identified as potential futures for S. In the second step, a subsequence of one of these futures is chosen as a base recommendation r, based on its similarity with S and its frequency in L. Finally, in the third step r is adapted to S, by characterizing (i) the differences between S and its aligned counterpart in the log session l from which r is extracted and (ii) the user's behavior during S. These characterizations adapt the technique of [2], and consist of extracting association rules from S and l.

A study of similarity measures tailored for OLAP sessions is provided in [3], where various similarity measures are devised and tested using both subjective (i.e., user) and objective tests.

KEY APPLICATIONS

OLAP personalization and recommendation techniques can be incorporated into any OLAP front-end tool that allows the user to compose and evaluate OLAP queries.

FUTURE DIRECTIONS

Although a number of different personalization and recommendation approaches already exist, a comprehensive comparative study of these approaches is still missing. Objective quality criteria for recommended queries only start to emerge [1] and should be completed, and subjective, user-based tests are still to be conducted. A long-term objective is to define a benchmark allowing assessing the effectiveness of OLAP recommendations, and more generally, how successful OLAP sessions are.

EXPERIMENTAL RESULTS

Approaches are usually evaluated from both the efficiency and effectiveness point of view. Efficiency is crucial in the sense that OLAP sessions are interactive by nature and personalized or recommended queries must be computed on the fly. Efficiency is measured as the time taken to obtain the personalized or recommended queries, varying the characteristics of the information used to obtain them. [7, 8] showed that recommending an OLAP query can be computed efficiently for logs of reasonable sizes. [2, 9] showed that personalization puts no significant overhead in the querying process, and that personalized queries are evaluated faster than non-personalized ones.

Effectiveness is typically measured in terms of reduction of the answer set for personalization approaches [2], or in terms of prediction accuracy for proactive approaches. In this latter case, [1, 7, 8] report effectiveness in terms of precision and recall of the recommendations when recommending for a sub-session of the log while the technique is trained on other parts of this log. More effectiveness quality criteria, including coverage, novelty or foresight of recommendation, are proposed and tested in [1].

DATA SETS

URL to CODE

The I3 project hosts the Java code of the operators described in [13, 14] and used in [8], distributed under the terms of the GPL: http://www.it.iitb.ac.in/~sunita/icube/

CubeLoad is a parametric generator of OLAP workloads written in Java (used in [1]), that can be used to generate a realistic profile-based workload in the form of sessions: http://big.csr.unibo.it/?q=node/371

CROSS REFERENCES

ON-LINE ANALYTICAL PROCESSING
MULTIDIMENSIONAL MODELING

CUBE
DIMENSION
HIERARCHY
MEASURE
PREFERENCE SPECIFICATION LANGUAGES
SKYLINE QUERIES AND PARETO OPTIMALITY
RECOMMENDER SYSTEMS
COLLABORATIVE FILTERING

RECOMMENDED READING

[1] J. Aligon, Similarity based recommendation of OLAP sessions, doctoral dissertation, Université François Rabelais Tours, France, 2013.

[2] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia, Mining preferences from OLAP query logs for proactive personalization, *ADBIS* 2011: 84–97.

[3] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia: Similarity measures for OLAP sessions. *Knowl. Inf. Syst.* 39(2): 463-489 (2014)

[4] M.-A. Aufaure, N. K. Beauger, P. Marcel, S. Rizzi, Y. Vanrompay, Predicting your next OLAP query based on recent analytical sessions, *DaWaK* 2013: 134–145.

[5] L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, D. Laurent: A personalization framework for OLAP queries. *DOLAP* 2005: 9-18

[6] A. Giacometti, P. Marcel, E. Negre: A framework for recommending OLAP queries. *DOLAP* 2008: 73-80

[7] A. Giacometti, P. Marcel, E. Negre, Recommending multidimensional queries, *DaWaK*, 2009: 453–466.

[8] A. Giacometti, P. Marcel, E. Negre, A. Soulet, Query recommendations for OLAP discovery-driven analysis, *IJDWM* 7 (2) (2011) 1–25.

[9] M. Golfarelli, S. Rizzi, P. Biondi, myOLAP: An approach to express and evaluate OLAP preferences, *IEEE TKDE 23* (7) (2011) 1050–1064.

[10] H. Jerbi, F. Ravat, O. Teste, G. Zurfluh, Preference-based recommendations for OLAP analysis, *DaWaK*, 2009: 467–478.

[11] H. Motro. Cooperative Database Systems. In *Encyclopedia of Library and Information Science*, Volume 66 Supplement 29. Marcel Dekker Inc., 2000, pp. 79–97.

[12] C. Sapia, PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems, *DaWaK*, 2000: 224–233.

[13] S. Sarawagi: iDiff: Informative Summarization of Differences in Multidimensional Aggregates. *Data Min. Knowl. Discov*. 5(4): 255-276 (2001)

[14] S. Sarawagi: User-cognizant multidimensional analysis. *VLDB J.* 10(2-3): 224-239 (2001)

[15] K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst*., 36(3):19, 2011.