A collaborative filtering approach for recommending OLAP sessions

Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi

A Collaborative Filtering Approach for Recommending OLAP Sessions

Julien Aligon [a], Enrico Gallinucci [b], Matteo Golfarelli [b], Patrick Marcel [a,*], Stefano Rizzi [b]

[a] *Laboratoire d'Informatique – Université François Rabelais Tours, France*

[b] *DISI, University of Bologna, Italy*

Abstract

While OLAP has a key role in supporting effective exploration of multidimensional cubes, the huge number of aggregations and selections that can be operated on data may make the user experience disorientating. To address this issue, in the paper we propose a recommendation approach stemming from collaborative filtering. We claim that the whole sequence of queries belonging to an OLAP session is valuable because it gives the user a compound and synergic view of data; for this reason, our goal is not to recommend single OLAP queries but OLAP sessions. Like other collaborative approaches, ours features three phases: (i) search the log for sessions that bear some similarity with the one currently being issued by the user; (ii) extract the most relevant subsessions; and (iii) adapt the top-ranked subsession to the current user's session. However, it is the first that treats sessions as first-class citizens, using new techniques for comparing sessions, finding meaningful recommendation candidates, and adapting them to the current session. After describing our approach, we discuss the results of a large set of effectiveness and efficiency tests based on different measures of recommendation quality.

Key words: OLAP, Similarity measures, Query recommendation, Personalization

1 Introduction

OLAP is the main paradigm for accessing multidimensional data in data warehouses. To obtain high querying expressiveness despite a small query formulation effort, OLAP provides a set of operations (such as drill-down and slice-and-dice) that transform one multidimensional query into another, so that OLAP queries are normally formulated in the form of sequences called *OLAP sessions* [1,2]. During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, applies one operation to determine a new query that will give her a better understanding of a business phenomenon. The resulting query sequences are strongly related to the user's skills, to the analyzed phenomenon, to the current data, and to the OLAP tool adopted.

While it is universally recognized that OLAP tools have a key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses, it is also commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating. Different approaches were taken in the literature to address this issue; in particular, in the area of personalization, both preference-based (e.g., [3,4]) and recommendation techniques (e.g., [5,6]) were specifically devised for OLAP systems.

* Corresponding author. Address: Campus de Blois, 3 place Jean Jaurés, 41000 Blois, France. Tel.: +332-5455-2155, Fax: +332-5455-2141

Email addresses: julien.aligon@univ-tours.fr (Julien Aligon), enrico.gallinucci2@unibo.it (Enrico Gallinucci), matteo.golfarelli@unibo.it (Matteo Golfarelli), patrick.marcel@univ-tours.fr (Patrick Marcel), stefano.rizzi@unibo.it (Stefano Rizzi).

In this paper we are specifically interested in recommendations. The original claim underlying our approach is that an OLAP session issued by a user is not just a casual path aimed at leading her to a single, valuable query (the one at the end of the session). Indeed, as stated in [7] with reference to clickstream analysis, path data contain information about a user's goals, knowledge, and interests. Undoubtedly, in the case of OLAP interactions, sessions are first-class citizens: the whole sequence of queries belonging to a session is valuable in itself, because (i) it gives the user a compound and synergic view of a phenomenon; (ii) it carries more information than a single query or set of queries by modeling the user's behavior after seeing the result of the former query; and (iii) several sessions may share the same query but still have quite different analyses goals. For this reasons, like done in [8] for recommending sets of pages to users of a Web site, we propose an approach whose goal is not to recommend single OLAP queries, but rather OLAP sessions. Some existing approaches recognize that sessions carry much more information about users' behavior than queries and recommend OLAP queries based on an analysis of past sessions (e.g., [2]); still, like all the other previous work on OLAP recommendation, they are focused on suggesting only single queries to users. In this sense our approach is highly innovative in the OLAP field, and therefore it requires brand new techniques.

Consistently with collaborative filtering approaches, our goal in deciding which sessions to recommend is to reuse knowledge acquired by other users during previous sessions. So let s_{cur} be the current user session for which we have to give a recommendation, and L be the session log. The approach we propose features three phases: *alignment* (search L for sessions that are similar to s_{cur} and optimally align each of them with s_{cur}), *ranking* (extract from the selected sessions the common subsessions and rate them according not only to their degree of similarity with s_{cur}, but also to how frequent they are in L), and *fitting* (adapt the top-ranked subsession r to s_{cur} and recommend the resulting session r'). To assess session similarity we readapt the alignment-based similarity function specifically devised for OLAP sessions in [1]. Note that OLAP users are normally grouped into *profiles* (e.g., CEO, marketing analyst, department manager); in this case, the session log can easily be partitioned by profiles so that each user can get recommendations based on sessions performed by users that share her background and expertise.

The qualifying properties of the recommendations we aim to give are *relevance*: this is obtained by ranking the log subsessions according to their frequencies so as to identify dense area of analysis that could be interesting for users; *foresight*: this is achieved by allowing even a subsession that is “far” from the alignment point with the current session (i.e., intuitively, far in the future) to be recommended; *novelty*: thanks to the fitting phase, the recommendation we give may not be part of the log; and *suitability*: during the fitting phase, the top-ranked subsession found in the log is adapted to the current session.

The paper outline is as follows. After discussing the related literature in Section 2, in Section 3 we introduce a formalization for multidimensional schemata, OLAP queries, and sessions. In Section 4 we present our approach to recommendation and describe its phases, while in Section 5 we discuss the results of experimental tests. Finally, Section 6 draws the conclusions.

2 Related Work

Recommender systems [9] are now well established as an information filtering technology and used in a wide range of domains. They are traditionally categorized as either *content-based* (suggestions are based on the user’s past actions only), *collaborative* (suggestions are based on similarities between users), or hybrid combinations.

A comprehensive survey of collaborative recommender systems evaluation is presented in [10]. Recommender systems are mostly evaluated with accuracy-based measures [11], typically predictive accuracy like MAE, or classification accuracy like precision and recall. Accuracy alone fails to capture the usefulness of recommendations, so other objective metrics related to suitability are being developed. For instance, coverage [10,12] is the degree to which recommendations cover the set of available items and the degree to which recommendations can be generated to all users. Novelty directly measures non-obviousness, often by referring to a fixed list of obvious recommendations, while serendipity measures how far novel recommendations may positively surprise users, for instance by reporting the rate of useful novel recommendations.

Recently recommender systems started to gain interest in the database community, with approaches ranging from content-based [13] to collaborative [14] query recommendation, especially to cope with the problem of interactively analyzing database instances [15,16]. This problem is particularly important in a data warehousing context, where one prominent use of such systems is to analyze the warehouse instance with OLAP queries as a basis for decision support. In a data warehouse context, the peculiarities of the multidimensional schema and queries can be leveraged. In [5], operators are proposed to analyze a query answer by automatically navigating to more detailed or less detailed aggregation levels, in order to detect surprising values to recommend. In [2], the log is represented as two Markov models: one that describes the transitions between query patterns (where a pattern is a simplified query expression) and one that describes transitions between selection predicates. These two models are used to construct the most probable query that follows the current query.

More recently, [4] proposes a content-based recommendation approach that synthesizes a recommendation by enriching the current query answer with elements extracted from a user's profile. [6] introduces a generic framework for query recommendation, where a distance measure between sessions is used to compare log sessions with the current session, and the set of final queries belonging to the closest log sessions are recommended and ranked according to their distance with the final query of the current session. [17] proposes to recommend a graph of former queries, based on the application of the operators of [5] on the query log, to detect surprising values regarding the current query answer. In [18], queries are recommended using a probabilistic model of former sessions, inspired by [2], where a similarity tailored for OLAP queries is used to group queries.

The existing approaches for query recommendation in data warehouses are summarized in Table 1 where, for each approach, we indicate (1) the category of the recommender system: content-based, collaborative filtering, or hybrid; (2) the source of information, i.e., whether the approach is log-driven, answer-driven, or both; (3) whether the approach is session-based and, if so, whether sequential aspects are considered or not; (4) the query model used, i.e., whether the approach leverages query expressions or query answers; (5) the technique used to process the input, i.e., whether the approach is similarity-based, preference-based, or stochastic; (6) the form (query or tuples) of the recommendation; (7) whether the recommendation is taken from a log, from the database instance, or from the current query; (8) the technique used to output the recommendation, i.e., if it is simply selected from the source or if it is synthesized from it; and finally (9) the metrics used for assessing the quality of recommendations: accuracy, coverage, novelty, foresight.

The first lesson learned from this literature review is that sessions are rarely treated as first-class citizens. Sequential aspects are seldom taken into account, and no approach ever considered recommending a sequence of queries. Approaches taking sessions into account only use them as input, to construct a model subsequently used for recommending. In addition, the stochastic approaches that consider sessions use a first order Markov Model, and therefore base their predictions only on the user's current query. Recommendation can be too much prescriptive (only one query) or too little prescriptive (a graph of queries). Besides, recommendations are rarely synthesized queries, but more often queries chosen among past queries stored in some query log or tuples retrieved from the database

Table 1

Query recommendation approaches in data warehouses

Ref.	Cat.	Input				Output			Quality
		Source	Session?	Model	Tech.	Form	Source	Tech.	metrics
[5]	CB	ans.	not seq.	ans.	stoch.	tuples	instance	sel.	acc.
[2]	CF	log	seq.	expr.	stoch.	query	curr.	synth.	-
[6]	H	log	seq.	ans.	sim.	query	log	sel.	acc.
[4]	CB	ans.	no	ans.	pref.	query	curr.	synth.	-
[17]	H	log/ans.	not seq.	ans.	stoch.	query	log	sel.	acc.
[18]	CF	log	seq.	expr.	stoch.	query	log	sel.	acc., cov.
Our approach	CF	log	seq	expr.	sim.	session	log/curr.	synth.	acc., cov., nov., fore.

instance. Many of the techniques proposed rely on query answers, which may result in a poor scalability compared to techniques using only query expressions, as reported in [14] in the case of databases. Finally, none of the former approaches assesses the system quality beyond accuracy and coverage. The approach we propose in this article overcomes these limitations. Sessions are treated as first-class citizen all along the process, query expressions are leveraged with a similarity tailored for OLAP sessions, the recommendation is synthesized from the log and the current session, and the quality of the recommender system is assessed using suitability metrics.

3 Preliminaries

In this section we define the multidimensional model we will use to formalize our approach and introduce a working example.

Definition 3.1 (Multidimensional Schema) *A* (multidimensional) schema *is a couple* $\mathcal{M} = \langle Hier, Meas \rangle$ *where:*

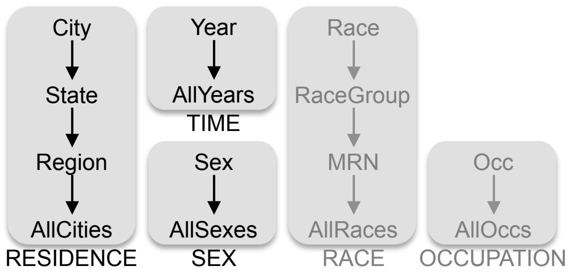


Fig. 1. Roll-up orders for the five hierarchies in the CENSUS schema

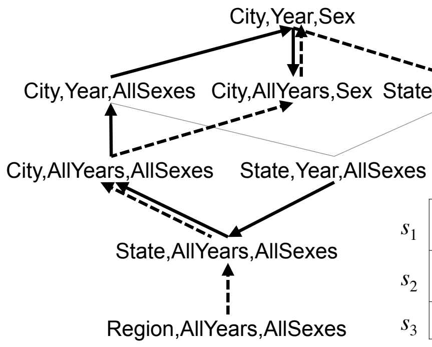
- *Hier is a finite set of* hierarchies*; each hierarchy $h_i \in Hier$ is associated to a set of levels and a* roll-up *partial order $\succeq_{h_i}$ of this set of levels;*
- *Meas is a finite set of* measures*, i.e., numerical attributes.*

Example 3.1 *IPUMS is a public database storing census microdata for social and economic research [19]. Its* CENSUS *multidimensional schema has five hierarchies, namely* RESIDENCE*,* TIME*,* SEX*,* RACE*, and* OCCUPATION*, and several measures. It is* City $\succeq_{\text{RESIDENCE}}$ State *(the complete roll-up orders are shown in Figure 1). While the experimental tests in Section 5 will be carried out on the complete* CENSUS *schema, as a working example we will use a simplified schema featuring only the* RESIDENCE*,* TIME*, and* SEX *hierarchies.*

To characterize the workload we consider a basic form of OLAP query centered on a single schema and characterized by a group-by, that defines a possible way to aggregate data, and a selection expressed by a conjunctive predicate. To be independent of the details related to logical design of multidimensional schemata and to query plans, we express queries using an abstract syntax.

Definition 3.2 (OLAP Query) *A* query fragment *on schema $\mathcal{M} = \langle Hier, Meas \rangle$ is either (i) a level $l \in h_i$ ($h_i \in Hier$), or (ii) a Boolean clause of type $l = value$, or (iii) a measure $m \in Meas$. A* query *on schema $\mathcal{M}$ is a set of fragments q such that:*

(1) q includes at least one level for each hierarchy $h_i \in Hier$; this set of levels defines the group-by *of q;*



	Sel. Predicate	*Measures*	*Group-by*
s_1	State=TN	AvgCostGas, SumIncTot	⟵
s_2	Region=MA	AvgPerWt, AvgCostGas	⟵
s_3	Region=ENC	AvgPerWt	⇠

Fig. 2. A log for our working example

(2) the conjunction of all Boolean clauses in q defines the selection predicate *of q;*

(3) q includes at least one measure; the set of measures $q \cap Meas$ is the one whose values are returned by q.

An OLAP session is an ordered sequence of correlated queries formulated by a user on a schema; each query in a session is typically derived from the previous one by applying an OLAP operator (such as roll-up, drill-down, and slice-and-dice).

Definition 3.3 (OLAP Session) *An* OLAP session *is a sequence $s = \langle q_1, q_2, \ldots \rangle$ of queries on schema $\mathcal{M}$. A* log *L is a set of sessions.*

Given a session s, we will denote with $length(s)$ the number of queries in s, with $s[w]$ ($1 \leq w \leq length(s)$) the w-th query of s, and with $s[v, w]$ ($1 \leq v \leq w \leq length(s)$) the subsession of s spanning from its v-th query to the w-th one. The last query of s, $s[length(s)]$, is briefly denoted with $s[.]$, so $s[v, .]$ is the subsession of s spanning from its v-th query to the end.

Example 3.2 *An example of query on the* CENSUS *schema is $q = \{$State, Year, AllSexes, State $=$ TN, AvgCostGas, SumIncTot$\}$. All examples will be based on a log that consists of 3 sessions, s_1, s_2, and s_3, each including 6 queries. For simplicity, predicates and measures are not changed during each session, while group-by's are changed as shown in Figure 2 with reference to a portion of the multidimensional lattice of the* CENSUS *schema.*

4 Our Approach

Let s_{cur} be the current OLAP session and L be a log of past sessions. As sketched in Figure 3, our approach to compute a recommendation for s_{cur} based on L includes three consecutive phases, described in detail in the following subsections:

(1) *Alignment*, described in Section 4.1, that selects from L a set of sessions that are similar to s_{cur} by finding an optimal alignment between each of them and s_{cur}. For each session l in this set, its *future* is defined as the subsession of l following the query that has been aligned with the last query of s_{cur} (i.e., with the last query currently issued by the user). The set of futures obtained in this way constitutes the set F of *candidate recommendations* for s_{cur}.

(2) *Ranking*, detailed in Section 4.2, that chooses a *base recommendation* r as a subsession of a candidate recommendation in F by considering both its similarity with s_{cur} and its frequency in L.

(3) *Fitting*, described in Section 4.3, that adapts r to s_{cur} by looking for relevant patterns in the query fragments of s_{cur} and r and using them to make changes to the queries in r, so as to deliver a recommendation r'.

Noticeably, the user has a possibility of influencing the recommendation by acting on a parameter called *minimum foresight*, denoted Φ, used in the ranking phase to select the base recommendation. With this threshold, the user indicates how distant from the current query the recommended session should be: higher values result in recommendations being farer from, and less strictly related to, the current session. Automatically adjusting this value based on the history of the user interactions with the system is beyond the scope of this work and is left as future work.

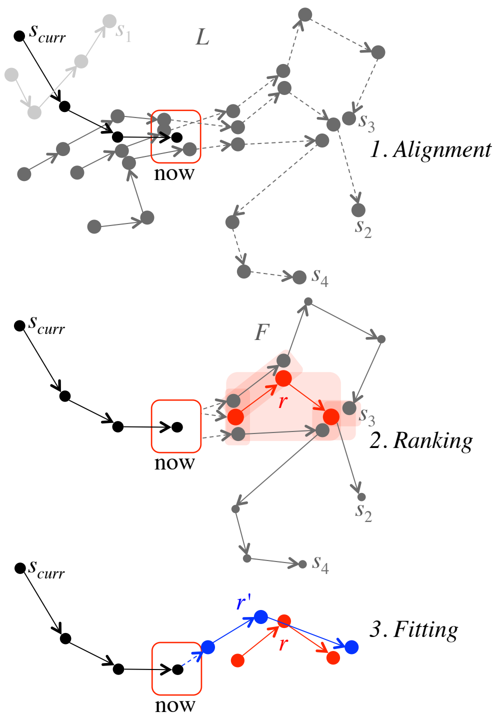


Fig. 3. The three recommendation phases

4.1 Alignment

The goal of this phase is to identify in the log L a set F of candidate recommendations for the current session s_{cur}. To do so, first we try to find an alignment between s_{cur} and each session l in L. The alignment algorithm we adopt is an adaptation of the one that was proposed in [1] to effectively capture the similarities between OLAP sessions; in turn, that algorithm is based on the Smith-Waterman algorithm [20], whose goal is to efficiently find the best alignment between subsequences of two given sequences by ignoring their non-matching parts.

The Smith-Waterman algorithm is a dynamic programming algorithm that computes the optimal alignment between two sequences s and s' based on a score matrix. In position (v, w), this matrix reports a score that expresses how well s and s' match when they are aligned ending in elements $s[v]$ and $s'[w]$; each score is recursively calculated by progressively adding the similarities between all pairs of matching elements in the two sequences (the similarity between two elements can also be negative, to express that they do not match). Intuitively, the algorithm seeks an optimal trade-off between the cost for introducing a gap in the matching subsequences and the cost for including a poorly matching pair of elements. In the extension proposed in [1] for OLAP sessions, sequence elements correspond to queries. A query similarity function, $\sigma_{que} \in [0..1]$, is defined as a combination of three components: one related to group-by's (based on how distant the two group-by's are in the multidimensional lattice), one to selection predicates (based on the distance of the levels on which predicates are formulated), and one to measure sets (their Jaccard index). A similarity threshold is then used to distinguish matches from mismatches. Besides, a time-discounting function is introduced to promote alignments based on recent queries, and a variable gap penalty is used to discourage discontinuous alignments. The output of the alignment algorithm when applied to two sessions s and s' is their best *alignment* a, defined by two matching subsessions $s[v_{start}, v_{end}]$ (denoted $s^{(a)}$) and $s'[w_{start}, w_{end}]$ (denoted $s'^{(a)}$). If an alignment between s and s' is found we say that s and s' are *similar*, denoted $s \sim s'$, and their similarity (computed as explained in [1]), is denoted with $\sigma_{ses}(a) \in]0..1]$. Otherwise it is $s \not\sim s'$.

For our alignment phase we use a variant, denoted SW_{cand}, that promotes alignments between the end of s_{cur} and the beginning of each log session l, so as to favor log sessions capable of providing a "long" candidate recommendation. This is achieved by modifying the two-dimensional logistic sigmoid function originally used as a time-discounting function as defined below and shown in Figure 4:

$$\rho_{cand}(v, w) = 1 - \frac{1 - \rho_{min}}{1 + e^{\frac{-20}{|l|}w + \frac{5}{|s_{cur}|}v + \frac{10}{|s_{cur}|}}} \ ,$$

where v is a position in s_{cur}, w is a position in l, and ρ_{min} is the minimal value desired for ρ_{cand} (i.e., the minimal weight given to query alignments considered as irrelevant).

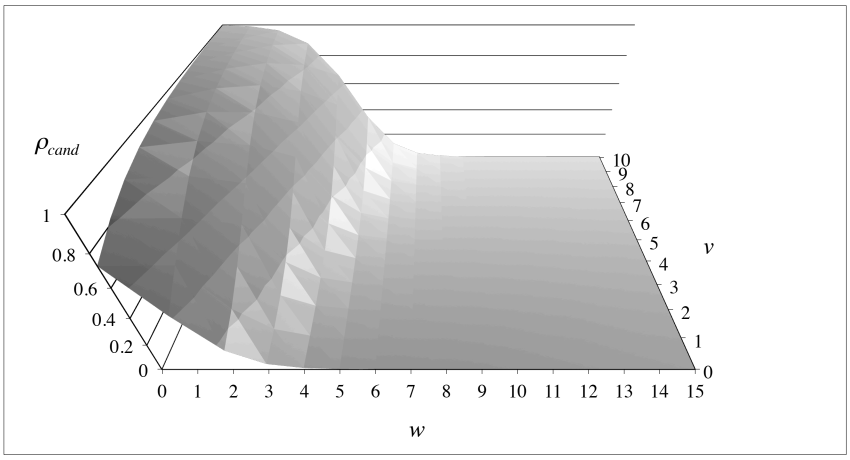


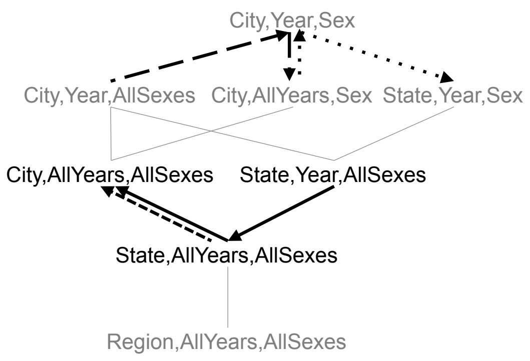
Fig. 4. The sigmoid function used for SW_{cand} ($|s_{cur}| = 10$, $|l| = 15$, $\rho_{min} = 0$)

Algorithm 1 Alignment

```
Input  $s_{cur}$: the current session, $L$: the log
Output  $F$: set of candidate recommendations
1: $F \leftarrow \emptyset$                                              ▷ Initialize $F$
2: for each $l \in L$ do
3:     $a \leftarrow SW_{cand}(s_{cur}, l)$                              ▷ Try to align $l$ with $s_{cur}$
4:     $v \leftarrow$ position in $l$ of $l^{(a)}[.]$
5:     if $l \sim s_{cur}$ and $s_{cur}^{(a)}[.] = s_{cur}[.]$ and $l[v+1,.] \neq \emptyset$ then    ▷ An alignment is found ending in $s_{cur}[.]$
6:         $f \leftarrow l[v+1,.]$                                   ▷ Find the candidate recommendation
7:         $F \leftarrow F \cup \{f\}$
8: return $F$
```

The constants have been experimentally tuned in order to answer to the specific desired behavior: $\frac{-20}{|l|}$ defines the proportion of queries in l whose alignment with the last queries of s_{cur} has to be favored; $\frac{5}{|s_{cur}|}$ defines the proportion of queries in s_{cur} whose alignment with the first queries of s has to be favored; $\frac{10}{|s_{cur}|}$ defines a minimal weight to consider between the first queries of s_{cur} and l.

The pseudocode for the whole alignment process is given in Algorithm 1. For each log session l such that $l \sim s_{cur}$ (line 2), its *future* is determined as the subsession $f = l[v+1, .]$ (line 6) where v is the position of the last query aligned (line 4). If the last query aligned in s_{cur} is $s_{cur}[.]$, i.e., the last query in s_{cur}, and l has a non-empty future f (line 5), then f is added to the set F of candidate recommendations (line 7).



	Sel. Predicate	*Measures*	*Group-by*
s_{cur}	Region=Mountain	AvgPerWt, SumIncTot	⟵
$s_1[1,3]$	State=TN	AvgCostGas, SumIncTot	⟵
$s_2[1,3]$	Region=MA	AvgPerWt, AvgCostGas	⟵
$s_3[2,3]$	Region=ENC	AvgPerWt	←----
$s_1[4,6]$	State=TN	AvgCostGas, SumIncTot	← —
$s_2[4,6]$	Region=MA	AvgPerWt, AvgCostGas	← —
$s_3[4,6]$	Region=ENC	AvgPerWt	←····
r	Region=MA	AvgPerWt, AvgCostGas	← —
r'	Region=Mountain	SumIncTot, AvgPerWt	← —

Fig. 5. A current session, its aligned log subsessions, its candidate recommendations, its base recommendation, and its recommendation

Example 4.1 *With reference to the log of Example 3.2, sketched in Figure 2, let the current session s_{cur} be the one whose group-by's, selection predicate, and measure set are shown in Figure 5. An alignment is found between s_{cur} and each session in the log. In particular, s_{cur} is aligned with log subsessions $s_1[1,3]$ (with similarity 0.15), $s_2[1,3]$ (similarity 0.21), and $s_3[2,3]$ (similarity 0.29); in the last case, similarity is higher (though the matching subsession is shorter) because the component of query similarity related to measure sets is higher. So, in this example the set of candidate recommendations is $F = \{s_1[4,6], s_2[4,6], s_3[4,6]\}$, obtained by considering the futures of the aligned log subsessions.*

4.2 Ranking

The goal of this phase is to examine F to determine the most suitable base recommendation r, which will then be refined in the fitting phase. Consistently with collaborative filtering approaches, we identify the densest areas in F so as to define r as the most relevant subsession in F. More precisely, this phase requires first to compute pairwise alignments between all sessions in F, and to use those alignments to assign a relevance

Algorithm 2 Ranking

```
Input $s_{cur}$: the current session, $F$: set of candidate recommendations, $\Phi$: minimum foresight
Output $r$: base recommendation
Variables: $A_{ij}$: sets of alignments
1: for each $f_i \in F, q \in f_i$ do                                        ▷ Initialize query scores
2:     $q.relevance \leftarrow 0$
3: $maxRelevance \leftarrow 0$
4: for each $f_i \in F$ do
5:     for each $f_j \in F, j > i$ do
6:         $A_{ij} \leftarrow SW_{rank}(f_i, f_j)$                              ▷ Compute pairwise alignments...
7:         for each $a \in A_{ij}$ do                                        ▷ ...and update query scores
8:             for each $q \in f_i^{(a)}$ do
9:                 $q.relevance \leftarrow q.relevance + \sigma_{ses}(a)$
10:            for each $q \in f_j^{(a)}$ do
11:                $q.relevance \leftarrow q.relevance + \sigma_{ses}(a)$
12:    for each $j \neq i, a \in A_{ij}$ do
13:        $avgRelevance \leftarrow \frac{\sum_{q \in f_i^{(a)}} q.relevance}{length(f_i^{(a)})}$        ▷ Compute score for $f_i^{(a)}$
14:        $b \leftarrow SW_{rank}(f_i^{(a)}, s_{cur})$                         ▷ Align $f_i^{(a)}$ with $s_{cur}$ to compute foresight
15:        if $avgRelevance > maxRelevance$ and $(1 - \sigma_{que}(f_i^{(a)}[.], s_{cur}[.])) \cdot (1 - \sigma_{ses}(b)) \geq \Phi$ then
16:            $maxRelevance \leftarrow avgRelevance$
17:            $r \leftarrow f_i^{(a)}$
18: return $r$                                                             ▷ Return the subsession with the highest score
```

score to each query $q \in f_i$, for each $f_i \in F$. Then, a relevance score is computed for each subsession that has been successfully aligned in each f_i by averaging the scores of its queries. Finally, r is chosen as the subsession with the highest relevance among those that meet the minimum foresight constraint Φ set by the user. Note that the ranking methods normally used in collaborative filtering approaches can hardly be adapted to the context of databases [14], and even less in our context because (i) we work with two different levels of aggregation: queries and sessions; and (ii) we compare objects (queries in our case) in terms of similarity and not of identity.

The pseudocode for this phase is sketched in Algorithm 2. The *for* loop starting at line 4 aims at finding, for each candidate recommendation f_i, its subsession yielding the highest relevance score. This is done by first computing the pairwise alignments between f_i and all the other sessions in F making use of a different version of the alignment algorithm. In this version, called SW_{rank}, no time-discounting function is used (as we do not need to favor alignments in particular positions), and every possible alignment is returned; so, differently from SW_{cand}, the result of $SW_{rank}(f_i, f_j)$ is not only the best alignment but a set of alignments A_{ij} between f_i and f_j. For each alignment $a \in A_{ij}$, in lines 7-11 we increase the scores of all aligned queries in f_i and f_j by the similarity $\sigma_{ses}(a)$. Eventually, the queries with the highest scores will be marking the densest areas in F, i.e., those that have been traversed the most by the sessions in F. Then, in lines from 12 to 17, the query scores are used to compute a score for each subsession of f_i corresponding to an alignment found with another session in F. In particular, for subsession $f_i^{(a)}$ aligned in a, its relevance score is computed as the average score of its queries. To check that the constraint on Φ is met, the foresight of $f_i^{(a)}$ is estimated in line 15 as the distance between its last query and the last query of the current session, weighted on the distance between the whole $f_i^{(a)}$ and s_{cur}.

Example 4.2 *With reference to our working example, the subsession in F yielding the highest score (0.32) is $s_2[4,6]$, which becomes the base recommendation r (see Figure 5).*

4.3 Fitting

The goal of this phase is to adapt the base recommendation r to the current session s_{curr}, i.e., to move r closer to s_{curr} while preserving its intrinsic logic. This is achieved by characterizing (i) the differences between s_{cur} and its aligned counterpart in the log session l from which r is extracted and (ii) the user's behavior during s_{cur}. These characterizations adapt the technique of [21], that is itself inspired by *label ranking*, a form of classification that has been shown to be effectively handled by association rules. In [21] we proposed to modify a query using association rules extracted from a query log. Our fitting phase therefore consists of extracting association rules from s_{cur} and l. Two types of rules, called Type-1 rules and Type-2 rules respectively, are extracted and used to transform r as sketched in Algorithm 3. Type-2 rules are those introduced in [21] and have been proved successful in a similar context, while Type-1 rules are novel and ensure that the final recommendation remains focused on the fragments frequently used in the current session. The main difference between the two types is the sessions they are computed from, which impacts the form of the rules.

A Type-1 rule aims at establishing a correlation between a query fragment x (either a measure, a group-by level, or a selection predicate) used in l and a query fragment y of the same type used in s_{cur}, so that r can be transformed by substituting all occurrences of x with y. These rules take the form $x \to y$ and are extracted from couples formed by a query q_i of s_{cur} and q'_j, the query of l aligned with q_i (line 1 of Algorithm 3). For instance, session l may be characterized by a recurrent selection predicate $\mathsf{Year} = 2013$ and be focused on measure $\mathsf{AvgCostGas}$, while session s_{cur} may be characterized by $\mathsf{Year} = 2011$ and measure $\mathsf{AvgPerWt}$; in this case, two rules $(\mathsf{Year} = 2013) \to (\mathsf{Year} = 2011)$ and $\mathsf{AvgCostGas} \to \mathsf{AvgPerWt}$ will be extracted aimed at making the base recommendation r more similar to the current session by focusing r on 2011 and $\mathsf{AvgPerWt}$ rather than on 2013 and $\mathsf{AvgCostGas}$.

Type-2 rules aim at finding query fragments used frequently together in s_{cur} (line 2), and have the form $X \to y$ with X a set of fragments of s_{cur} and y a fragment of s_{cur}. For instance, rule $\{\mathsf{AvgCostGas}, (\mathsf{Year} = 2011)\} \to \mathsf{Region}$ captures the fact that the trends of measure $\mathsf{AvgCostGas}$ for 2011 are mainly analyzed in s_{cur} on a region-by-region basis. If in r the same measure for 2011 is analyzed by State instead, this rule can be used to adapt r by changing query group-by's from State to Region. Noticeably, letting both types of rules potentially work with *all* query fragments ensures that the full range of transformations between OLAP sessions are covered.

Rules of both types are ranked according to the geometric mean of the following figures, which have been experimentally selected: (i) the support of the rule; (ii) the confidence of the rule; (iii) the average position in s_{cur} where the head fragment y appears (to favor recent fragments); (iv) the support of the head fragment y in s_{cur} (to favor frequent fragments). Note that, for Type-1 rules, support is given by $supp(x \rightarrow y) = \frac{|\{(q_i,q'_j) \text{ s.t. } q_i \in s_{cur}, q'_j \in l, x \in q'_j, y \in q_i\}|}{|s_{cur}|}$, while for Type-2 rules it is $supp(X \rightarrow y) = \frac{|\{q_i \text{ s.t. } q_i \in s_{cur}, X \cup \{y\} \subseteq q_i\}|}{|s_{cur}|}$. The confidence of a rule $B \rightarrow H$ (where B and H are sets of fragments) is $conf(B \rightarrow H) = \frac{supp(B \cup H)}{supp(B)}$.

The rules extracted are used to change the fragments Fr shared by all the queries of the base recommendation r (line 3 of Algorithm 3). This ensures that the fitting process respects the intrinsic logic of r, without producing two identical queries in the recommended session. First, Type-1 rules $x \rightarrow y$ are considered (lines 5 to 11), in descending order (line 6), as follows. If fragment x exists in r and not in s_{cur}, then this fragment is replaced by y in r. Every modified fragment is marked so as not to be adapted any more (line 10). Then, Type-2 rules $X \rightarrow y$ are considered (line 12 to 22), in descending order (line 13), only for changing the queries that include the rule body X (line 14), as follows. If the rule head y is a selection predicate or a level and is not already present in the query (line 15), then it replaces the corresponding fragment (i.e., the one belonging to the same hierarchy) of the query (line 17). If y is a measure that is not already present in r (line 19), then it is added to the measure set of the query (line 21). Like for Type-1 rules, once a fragment is modified, it is marked so it is not modified any more (line 21).

Note that, as an effect of the fitting phase, there is a possibility that the recommendation produced, r', includes some queries that are identical to queries that were part of s_{cur} (i.e., queries that the user has already formulated during the current session). Such a recommendation would be completely useless. So, in this case, we go back to the ranking phase and take as a base recommendation the next most relevant subsession.

Algorithm 3 Fitting

```
Input $s_{cur}$: the current session, $r$: the base recommendation, $l$: the log session from which $r$ was extracted
Output $r'$ : the recommended session
1: $T_1 \leftarrow ExtractType1Rules(s_{cur}, l)$                    ▷ Extract rules
2: $T_2 \leftarrow ExtractType2Rules(s_{cur})$
3: $Fr \leftarrow \bigcap_{q \in r} q$                    ▷ Set of shared fragments in $r$
4: $r' \leftarrow r$
5: while $Fr \neq \emptyset$ and $T_1 \neq \emptyset$ do                    ▷ Apply Type 1 rules
6:     $(x \rightarrow y) \leftarrow TopRank(T_1)$
7:     if $x \in Fr$ and $x \notin q \, \forall q \in s_{cur}$ then
8:         for $i = 1$ to $length(r')$ do
9:             $r'[i] \leftarrow r'[i] \setminus \{x\} \cup \{y\}$
10:        $Fr \leftarrow Fr \setminus \{x\}$
11:    $T_1 \leftarrow T_1 \setminus \{x \rightarrow y\}$
12: while $Fr \neq \emptyset$ and $T_2 \neq \emptyset$ do                    ▷ Apply Type-2 rules
13:    $(X \rightarrow y) \leftarrow TopRank(T_2)$
14:    if $X \subseteq Fr$ then
15:        if $y$ is a group-by level or a selection predicate and $\exists z \in Fr$ corresponding to $y$ then
16:            for $i = 1$ to $length(r')$ do
17:                $r'[i] \leftarrow r'[i] \setminus \{z\} \cup \{y\}$
18:            $Fr \leftarrow Fr \setminus \{z\}$
19:        else if $y \notin q \, \forall q \in r'$ then
20:            for $i = 1$ to $length(r')$ do
21:                $r'[i] \leftarrow r'[i] \cup \{y\}$
22:    $T_2 \leftarrow T_2 \setminus \{X \rightarrow y\}$
23: return $r'$
```

Example 4.3 *In our working example, the only applicable rules are the Type-1 rules* AvgCostGas $\rightarrow$ SumIncTot *and* (Region $=$ MA) $\rightarrow$ (Region $=$ Mountain)*, which transform r into r' by changing its measure set and its selection predicate as shown in Figure 5.*

5 Experimental Results

This section describes the results of the experimental tests we performed. After explaining the measures we use to assess the quality of a recommendation, we describe the techniques we used to generate session logs for the experiments. Then, we report and discuss the test results from both the effectiveness and efficiency points of view.

5.1 Assessing the Quality of Recommendations

Consistently with the literature on recommender systems (see Section 2), we assess the accuracy and coverage of our recommender system first, and then use more elaborated measures to assess the suitability of recommendations.

Accuracy is measured by extending the classical precision and recall measures to take session similarity into account. Let L be a log and S be a set of current sessions for which recommendations are to be computed. Given session $s \in S$, let f_s be its actual future (i.e., the sequence of queries the users would have formulated after $s[.]$ if she had not been given any recommendation) and r_s be the future recommended by our system. Recommendation r_s is considered to be *correct* when $r_s \sim f_s$, i.e., when it is similar to the actual future of s.

Let $FS = \{f_s \,;\, s \in S\}$ and $RS = \{r_s \,;\, s \in S\}$. The set of true positives is then defined by

$$TP = \{r_s \in RS \,;\, r_s \sim f_s\}$$

i.e., the set of recommended futures similar to their actual counterparts. The set of false positives is $FP = RS \setminus TP$ and the set of false negatives is $FN = \{f_s \in FS \,;\, f_s \not\sim r_s\}$. Then, *recall* is $\frac{|TP|}{|TP|+|FN|}$ and *precision* is $\frac{|TP|}{|TP|+|FP|} = \frac{|TP|}{|RS|}$.

The global accuracy is measured using the *F-measure* [22]:

$$F = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

while the *coverage* is $\frac{|RS|}{|S|}$.

To assess the suitability of a recommendation r_s for a current session s we adopt two more measures. *Foresight*, measured like in Section 4.2, indicates how "far" from the current query of s the last query of r_s is, weighted by the distance between the two sessions:

$$Foresight(s) = (1 - \sigma_{que}(s[.], r_s[.])) \cdot (1 - \sigma_{ses}(a))$$

(where a is the best alignment between s and r_s, and $\sigma_{ses} = 0$ if no alignment can be found). *Novelty* indicates how distant r_s is from the sessions in the log:

$$Novelty(s) = min_{l \in L}(1 - \sigma_{ses}(a'))$$

(where a' is the best alignment between l and r_s).

5.2 Log Generation

The benchmark we adopted for our tests is based on a set of synthetic logs over the CENSUS schema. The sessions in each log were generated using *CubeLoad*, a Java application specifically aimed at generating realistic workloads [23]. Workload realism is achieved in CubeLoad in different ways:

- OLAP users are normally grouped into profiles with different skills and involved in business analyses with different features. Often, a profile has one or more *segregation predicates*, i.e., it can only view a specific slice of the cube data. Both differentiated profiles and specific segregation attributes are modeled and tunable.
- When a user logs to the OLAP front-end, she is typically shown a page where some predefined "seed" queries are linked. Sometimes seed queries include a *prompt*, meaning that the front-end asks the user to select one value out of the domain of a level. Both seed queries and prompts are managed and tunable.
- After executing one seed query, the user starts applying a sequence of OLAP operations

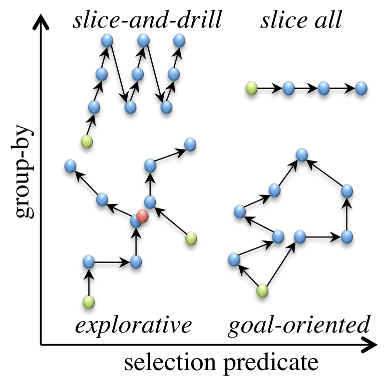


Fig. 6. Session templates (seed queries in green, surprising queries in red) in CubeLoad [23]

that progressively transform a query into another so as to build an analysis session. In CubeLoad, each session starts with a random seed query then it evolves by applying at each step one atomic OLAP operation to the previous query.

- Features such as the number of seed queries available, the maximum size and complexity of reports returned by seed queries (in terms of dimensionality, size, and number of measures), and the average length of sessions may significantly depend on the typical ICT skills and business understanding for the users of each profile —besides on the quality of the OLAP front-end. All these parameters are fully tunable for each single profile.
- Some recurrent types of user analyses are normally found in OLAP workloads. To cope with this, session evolution in CubeLoad is driven by four predefined templates (see Figure 6): *slice-all*, where the value of a selection predicate is iteratively changed, *slice-and-drill*, where sequences of slicing and drill-down operations are executed on hierarchies, *explorative*, where sessions quickly move towards one in a small set of "interesting" queries and then evolve randomly, and *goal-oriented*, where each session slowly converges to one final query chosen from a set of randomly generated ones. These four templates are uniformly distributed in CubeLoad workloads, and are completely independent of the parameters described above (number of profiles, session length, etc.).

5.3 Effectiveness Tests

This section presents the results of the seven groups of tests we conducted to assess the accuracy and coverage of our recommender system, as well as the suitability of the generated recommendations. All tests were conducted on a 64-bits Intel Core i7 quad-core 3.4GHz, with 8GB RAM, running Windows 7 pro SP1.

A set of tests on a log L generated by CubeLoad is executed by iteratively (i) picking one session $s \in L$; (ii) taking subsession $s[1,4]$ as the current session and subsession $f_s = s[5,.]$ as its actual future (except for the third group of tests, where the position of the current query is varied); (iii) finding a recommendation r_s for $s[1,4]$ using the remaining sessions, $L \setminus \{s\}$, as the log sessions. As explained in Section 5.1, r_s is considered to be correct when $r_s \sim f_s$, non correct otherwise.

The aim of the first group of tests we carried was to tune our approach, i.e., to determine a good trade-off between accuracy and coverage. To this end we recall from Section 4.2 that the base recommendation is chosen, among the candidate recommendations, as the one with the highest relevance score; this suggests to check how often selecting a subsession with low relevance leads to a wrong recommendation. So in this tests we gave a recommendation only if the base recommendation had relevance higher than a *minimum relevance* threshold, otherwise we gave no recommendation. The results in Figure 7 show how the accuracy and coverage change when the minimum relevance increases, on a log of 200 sessions. When the minimum relevance is 0 no filtering of base recommendations is made, so coverage is about 90%. Expectedly, precision increases with the minimum relevance, while coverage —and therefore recall— decrease quite rapidly, meaning that base recommendations often have low relevance. However, the curve for precision clearly shows that our approach is well capable of delivering good recommendations even out

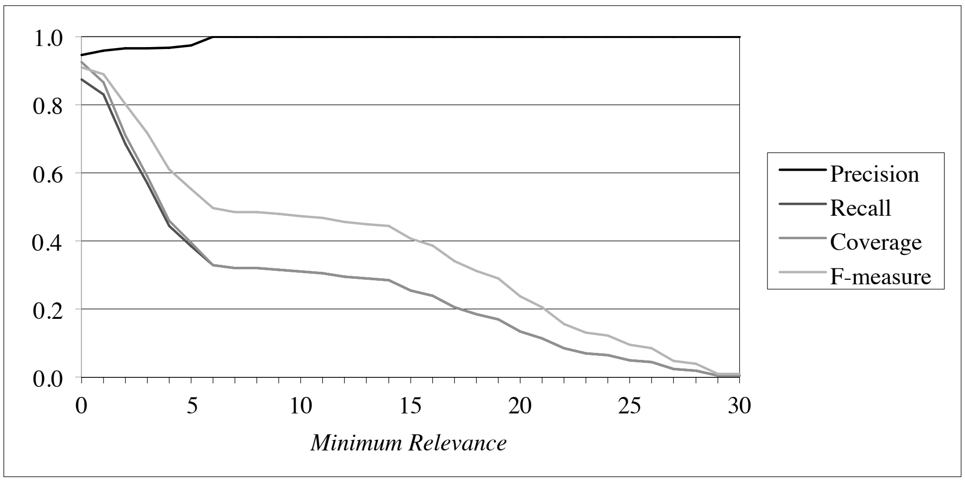


Fig. 7. Accuracy vs. coverage trade-off in function of the base recommendation minimum relevance

of base recommendations with low relevance. These facts are well summarized by the F-measure curve, showing that the best overall performance is achieved when the minimum relevance is 0. Therefore, no minimum relevance threshold was posed for all the following tests.

The focus of the second group of tests was to observe how the approach performs on logs with different characteristics. To this purpose, we tuned CubeLoad parameters to create two series of logs: the first one with different densities, the second one with different session lengths. Note that a *clustered* log is meant as one with dense groups of similar sessions (specifically, each session is similar to about 30 other sessions on average), whereas in a *sparse* log all sessions tend to be dissimilar from each other (each session is similar to about 15 other sessions on average). The minimum foresight Φ was set to 0. As shown in Figure 8 (top left), the coverage increases as sessions get more clustered, while precision is not significantly affected; this behavior is consistent with that of collaborative filtering approaches, where the capability of giving a recommendation depends on the quantity of data that matches the user's query. Also, Figure 8 (top right) shows that it is hard to give good recommendations when log sessions are short; indeed, the shorter the log

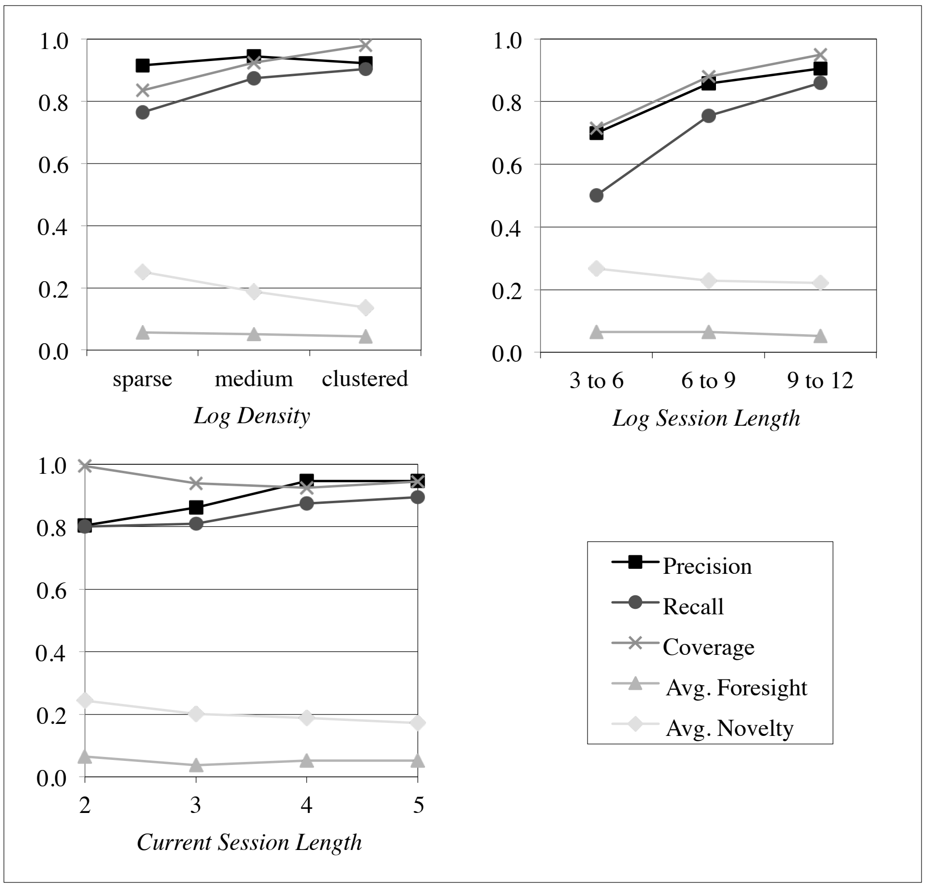


Fig. 8. Effectiveness vs. log and session features

sessions, the less likely for the recommendation to be similar to the (even shorter) actual future —therefore, the lower the precision. As to the average foresight, in these tests it is expectedly low because $\Phi = 0$. Finally, the average novelty is relatively higher but still it does not exceed 0.3, again as a consequence of having set $\Phi = 0$; the reason for this relationship between novelty and Φ will be explained below (see comments to the fourth group of tests).

The core question of the third group of tests was: *which is the best time in the user's OLAP journey to give her a recommendation?* In other words, we analyzed how the recommendation effectiveness changes with the length of the current session on a log with medium density and medium length of sessions (again with $\Phi = 0$). The results in Figure 8 (bottom) show that increasing the length of current sessions has a clear

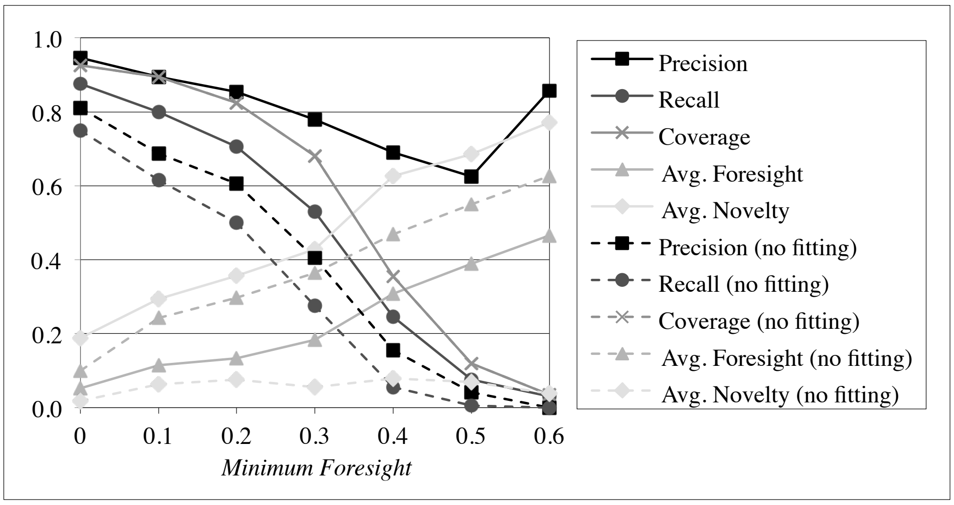


Fig. 9. Effectiveness vs. minimum foresight

positive effect on accuracy: this is compatible with the intuition that a longer past is more explanatory of the user's behavior and intentions, thus leading to the delivery of better recommendations.

The fourth group of tests was aimed at measuring the effectiveness of the parameter Φ set by the user to rule the minimum foresight of the base recommendation. The log used for these tests is again the one with medium density and medium length of sessions, and current sessions have length 4. Figure 9 enables a comparison of the accuracy and suitability measures with the fitting phase of our approach switched on (plain lines) and off (dashed lines); in this regard, some interesting observations can be made:

- The average foresight of the base recommendation (no fitting) is always higher than the minimum foresight Φ, which confirms the correctness of the approach. However, the average foresight of the final recommendation (with fitting) is slightly lower: in fact, fitting inevitably reduces the foresight by applying modifications that move the base recommendation closer to the current session.
- Fitting has a strong impact on the recommendation correctness. Not only precision is improved by fitting when $\Phi = 0$, which indeed is the motivation for the fitting phase,

but the increase in precision caused by fitting with higher values of Φ is remarkable. The reason is that, when Φ is high, the base recommendation tends to be very distant from the current session, so it is probably not similar to the actual future; however, fitting is quite effective in making the base recommendation compatible with the current session.

- The novelty of the base recommendation is always very low; this is expected, as the base recommendation is just a portion of a log session, hence it will always have a strong similarity with the session from which it was extracted. The novelty of the given recommendation is much higher, indicating that the recommendation is actually something new with respect to what we have in the log. Interestingly, the novelty reaches very high values when Φ is also high. This can be explained by considering that the sessions in the log tend to be clustered into some high-density areas; to achieve a high foresight for a current session taken from one of these clusters, base recommendations are mostly chosen from a low-density inter-cluster areas of the log, so fitting transforms them into something very different from the other log sessions.

The fifth group of tests investigates how the recommendation accuracy changes with continued usage of the system. These tests were made on the same three logs used for the second group of tests (a sparse one, a clustered one, and an intermediate one), and the results were averaged. The sessions in each log were then randomly grouped in groups of 20: at the first step, the 20 sessions in the first group were put in the log, and the 20 sessions in the second group were used as current sessions; at the second step, the 20 sessions in the second group were added to the log and the 20 sessions in the third group were used as current sessions; and so on. As a result, the log size is increased by steps of 20 in a way that simulates real usage of the system. Figure 10 shows the results. As expected, recall and coverage increase quickly with the log size; precision is quite stable and above 80% even when the log is very small.

The sixth group of tests is focused on the robustness of the approach in terms of stability

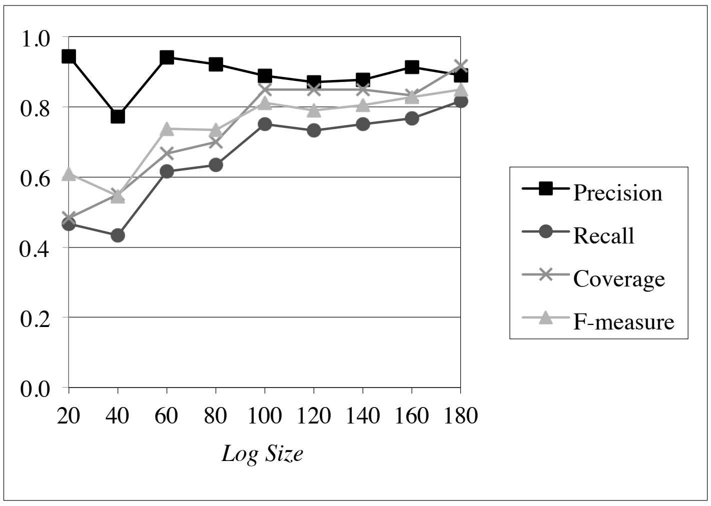


Fig. 10. Effectiveness vs. log size

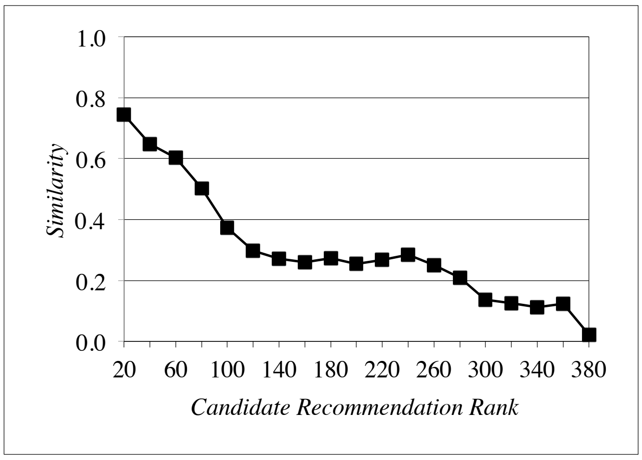


Fig. 11. Inter-session similarity for increasing rank of candidate recommendations

of the recommendation. Figure 11 shows the similarity between the base recommendation and the other candidate recommendations ordered according to their relevance (averaged on groups of 20). The curve smoothly decreases, which means that the most relevant subsessions, as ranked during the ranking phase, are all quite similar to each other. This means that small variations in the current session or in the log features will not change drastically the final recommendation returned.

Finally, we compared our approach with the one proposed in [6] (slightly modified to recommend sessions instead of ordered sets of queries), using a log of 200 sessions with minimum relevance 0 and minimum foresight 0. The approach of [6] is clearly outperformed in terms of accuracy, with a precision of 0.52 (against 0.94) and a recall of 0.52 (against 0.87). This is explained by the fact that this approach always computes a recommendation (coverage 1 against 0.92) that is chosen from the log (novelty 0 against 0.18, foresight 0.18 against 0.05).

5.4 Efficiency Tests

Computing effective recommendations is useless if they are not returned in a time frame compatible with OLAP interactions. Figure 12 shows the execution times for our system considering logs with different features. Execution times are split according to the three phases: alignment, ranking, and fitting. Overall, execution times rarely exceed 50 msec., which is perfectly compatible with a human-computer interaction.

Before analyzing in more details the system behavior, we briefly discuss the computational complexity of the three phases. Algorithm 1 looks for the best Smith-Waterman alignment between the current session and those in the log, thus its computational complexity is $O(|L| \times \overline{v}^2)$ where $\overline{v}$ is the average length of the log sessions [1]. Algorithm 2 ranks the candidate sessions computing an all-against-all Smith-Waterman alignment, thus its computational complexity is $O(|F|^2 \times \overline{v}^2)$ where F is the set of candidate recommendations. Finally, Algorithm 3 applies fitting to the base recommendation; its computational complexity is mainly ruled by Type-2 rules extraction, requiring to extract all rules, even infrequent ones, which has an exponential complexity (see e.g., [24]). The time taken remains acceptable though, since the average number of fragments $\overline{f}$ that are common to all the queries of a base recommendation is low. Type-1 rules extraction is polynomial, due to the nature of the rules extracted. However, Type-1 rules extraction takes as input the set of fragments of both the current session and the log session, whose size can be

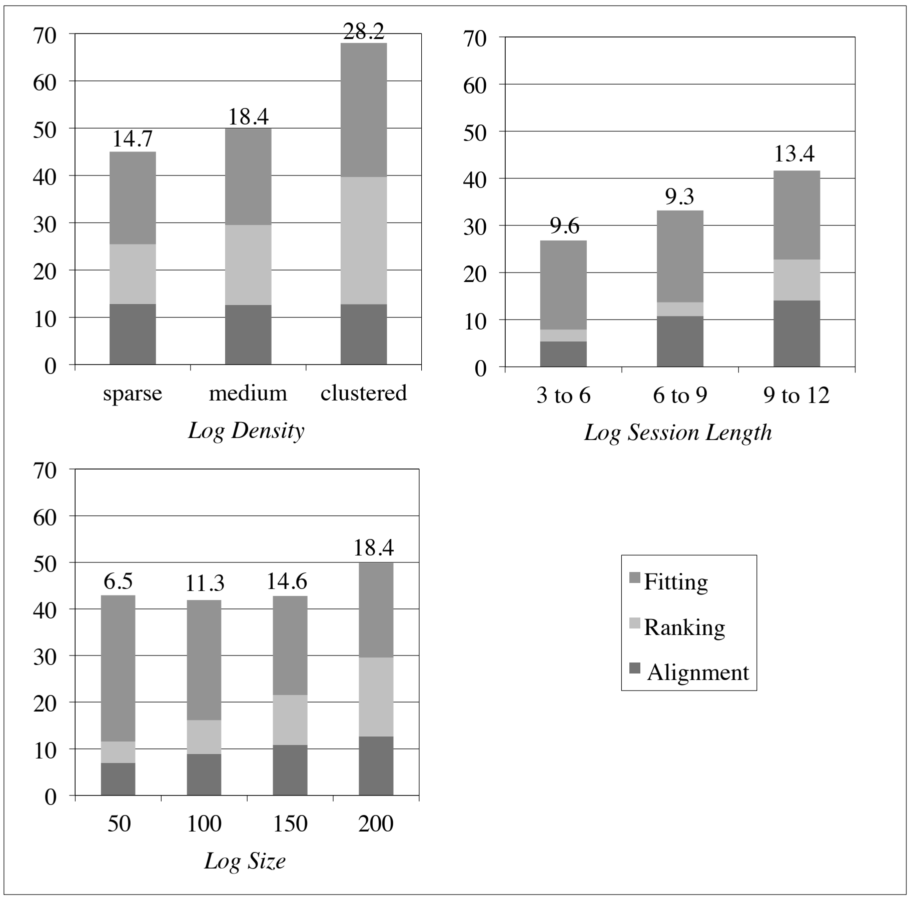


Fig. 12. Efficiency vs. log features (all times in msec.; unless otherwise stated, all logs have size 200; labels report the average number of candidate sessions)

greater than $\overline{f}$, especially in the presence of small logs, where the similarity between the log session and the current session is likely to be low. Clearly, the execution time of each phase depends on the one hand on the cost of the basic operation carried out (i.e., alignment for Algorithms 1 and 2), on the other hand on the number of times such operation is executed. In the light of this premise, the following considerations hold:

- The costs for alignment and ranking increase with the log size $|L|$ and the average session length $|v|$, which jointly determine the number of alignments found with the current session.
- Though fitting works on a single session, its cost is predominant due to the high computational complexity of rule extraction. Unexpectedly, the execution time of fitting

increases as the log size decreases; this is due to the fact that the extraction of Type-1 rules is computationally more expensive when the current session is less similar to the log session, which is more likely in the case of smaller logs.

- As to the two remaining phases, as suggested by the complexity estimates reported above, the predominance of either alignment or ranking depends on the relationship between $|F|^2$ and $|L|$: the cost of ranking tends to become higher than that of alignment for large and clustered logs, that determine several candidate recommendations thus making $|F|^2 > |L|$.

As to the comparison with [6], the tests show that our approach is slightly worse in terms of efficiency (50 msec. against 17.8 msec.) due to the extra costs paid for the fitting phase; however, as discussed in Section 5.3, the higher effectiveness largely compensates for this lower efficiency.

6 Conclusions

In this paper we have proposed a collaborative filtering approach to recommendation of OLAP sessions that reuses knowledge acquired by other users during previous sessions. Like other collaborative approaches, ours follows a three-phases process, but it is the first that treats sessions as first-class citizens, using brand new techniques for comparing sessions, finding meaningful recommendation candidates, and adapting them to the current session. We have extensively evaluated it by discussing its efficiency and its effectiveness from different points of view. Though our approach ensures that the returned recommendations have several desirable properties (such as novelty, relevance, and foresight), we plan to further improve it under different aspects:

- We have observed that large logs and longer log sessions are necessary to obtain a good coverage (see Figures 8 and 10). To cope with this well known cold-start problem, extending our approach with non collaborative recommendations (like, e.g., [5]) is a promising research direction.

- As shown in Figure 11, several candidate recommendations with high relevance can normally be found. Though we have chosen to recommend only the top-1 session, the approach can be easily reworked to recommend a top-k set of sessions. In this case, the approach effectiveness could benefit from query result diversification [25].
- The user should be enabled to easily understand in which direction a recommendation will guide her through multidimensional data. Since we are recommending sessions rather than single queries, and sessions are complex objects, a visualization problem arises. Solving this problem requires to (i) understand the set of features that describe an OLAP session direction at best, and to (ii) find a good visualization metaphor.
- Our approach has been tested with synthetic, yet realistic workloads. However, given real OLAP logs, characterizing user sessions and analyzing sessions and queries to filter out the irrelevant ones (e.g., those showing an erratic behavior due to a trial-and-error user approach), remain open problems, that should be solved to better adapt our approach to different kinds of users. We are currently working to collect real, user-annotated logs, as well as user feedback on our recommender system.

References

[1] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia, Similarity measures for OLAP sessions, KAIS 39 (2) (2014) 463–489.

[2] C. Sapia, PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems, in: Proc. DaWaK, London, UK, 2000, pp. 224–233.

[3] M. Golfarelli, S. Rizzi, P. Biondi, myOLAP: An approach to express and evaluate OLAP

preferences, IEEE TKDE 23 (7) (2011) 1050–1064.

[4] H. Jerbi, F. Ravat, O. Teste, G. Zurfluh, Preference-based recommendations for OLAP analysis, in: Proc. DaWaK, Linz, Austria, 2009, pp. 467–478.

[5] S. Sarawagi, G. Sathe, i^3: Intelligent, interactive investigaton of OLAP data cubes, in: Proc. SIGMOD, Dallas, Texas, 2000, p. 589.

[6] A. Giacometti, P. Marcel, E. Negre, Recommending multidimensional queries, in: Proc. DaWaK, Linz, Austria, 2009, pp. 453–466.

[7] A. Montgomery, S. Li, K. Srinivasan, J. Liechty, Modeling online browsing and path analysis using clickstream data, Marketing Science 23 (4) (2004) 579–595.

[8] S. Gündüz, M. T. Özsu, A web page prediction model based on click-stream tree representation of user behavior, in: Proc. KDD, Washington DC, USA, 2003, pp. 535–540.

[9] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749.

[10] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53.

[11] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, Journal of Machine Learning Research 10 (2009) 2935–2962.

[12] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proc. RecSys, Barcelona, Spain, 2010, pp. 257–260.

[13] M. Drosou, E. Pitoura, YmalDB: exploring relational databases via result-driven recommendations, The VLDB Journal (2013) 1–26.

[14] M. Eirinaki, S. Abraham, N. Polyzotis, N. Shaikh, QueRIE: Collaborative database exploration, IEEE Trans. Knowl. Data Eng. 26 (7) (2014) 1778–1790.

[15] N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, D. Suciu, A case for a collaborative query management system, in: Proc. CIDR, Asilomar, CA, 2009.

[16] M. L. Kersten, S. Idreos, S. Manegold, E. Liarou, The researcher's guide to the data deluge: Querying a scientific database in just a few seconds, PVLDB 4 (12) (2011) 1474–1477.

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

[18] M.-A. Aufaure, N. K. Beauger, P. Marcel, S. Rizzi, Y. Vanrompay, et al., Predicting your next OLAP query based on recent analytical sessions, in: Proc. DaWaK, Prague, Czech Republic, 2013, pp. 134–145.

[19] Minnesota Population Center, Integrated public use microdata series, http://www.ipums.org (2008).

[20] T. Smith, M. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology 147 (1981) 195–197.

[21] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia, Mining preferences from OLAP query logs for proactive personalization, in: Proc. ADBIS, Vienna, Austria, 2011, pp. 84–97.

[22] C. J. van Rijsbergen, Information Retrieval, Butterworth, 1979.

[23] S. Rizzi, E. Gallinucci, CubeLoad: a parametric generator of realistic OLAP workloads, in: Proc. CAiSE, Thessaloniki, Greece, 2014, pp. 610–624.

[24] P. Purdom, D. V. Gucht, D. Groth, Average-case performance of the apriori algorithm, SIAM J. Comput. 33 (5) (2004) 1223–1260.

[25] P. Fraternali, D. Martinenghi, M. Tagliasacchi, Top-k bounded diversification, in: Proc. SIGMOD, New York, NY, USA, 2012, pp. 421–432.