Automatic Assessment of Interactive OLAP Explorations.

Mahfoud Djedaini, Krista Drushku, Nicolas Labroche, Patrick Marcel, Veronika Peralta, Willeme Verdeaux

Version of Record: https://www.sciencedirect.com/science/article/pii/S0306437918301005
Manuscript_011d55ed7de7160e40702925dd349755

Automatic Assessment of Interactive OLAP Explorations

Mahfoud Djedaini[a], Krista Drushku[b,a], Nicolas Labroche[a], Patrick Marcel[a,*], Verónika Peralta[a], Willeme Verdeaux[a]

[a] *University of Tours, France*
[b] *SAP Research, France*

Abstract

Interactive Database Exploration (IDE) is the process of exploring a database by means of a sequence of queries aiming at answering an often imprecise user information need. In this paper, we are interested in the following problem: how to automatically assess the quality of such an exploration. We study this problem under the following angles. First, we formulate the hypothesis that the quality of the exploration can be measured by evaluating the improvement of the skill of writing queries that contribute to the exploration. Second, we restrict to a particular use case of database exploration, namely OLAP explorations of data cubes. Third, we propose to use simple query features to model its contribution to an exploration. The first hypothesis allows to use the Knowledge Tracing, a popular model for skill acquisition, to measure the evolution of the ability to write contributive queries. The restriction to OLAP exploration allows to take advantage of well known OLAP primitives and schema. Finally, using query features allows to apply a supervised learning approach to model query contribution. We show on both real and artificial explorations that automatic assessment of OLAP explorations is feasible and is consistent with the user's and expert's viewpoints.

1 Introduction

Interactive Data Exploration support addresses the development of techniques that allow users to interactively explore their data and help them to better gain insights. Many approaches have recently been developed to support IDE, as illustrated by a recent survey of the topic [1]. Typically, an exploration includes several queries where the result of each query triggers the formulation of

*Corresponding author
Email addresses: Mahfoud.Djedaini@univ-tours.fr (Mahfoud Djedaini), Krista.Drushku@sap.com (Krista Drushku), Nicolas.Labroche@univ-tours.fr (Nicolas Labroche), Patrick.Marcel@univ-tours.fr (Patrick Marcel), Veronika.Peralta@univ-tours.fr (Verónika Peralta), Willeme.Verdeaux@etu.univ-tours.fr (Willeme Verdeaux)

the next one. OLAP analysis of data cubes is a particular case of IDE, that takes advantage of simple primitives like drill-down or slice-and-dice for the navigation. Given the exploratory nature of OLAP analysis of multidimensional data (see e.g., [2, 3]), many exploration techniques have been specifically developed in the context of interactive OLAP exploration of data cubes. However, while there exists many benchmarks recognized by the database community as relevant for evaluation and comparison of performance of database systems, such as the benchmarks from TPC organization, there is yet no commonly agreed upon method for evaluating to what extent interactive explorations conducted with such systems are indeed successful.

In this paper, we propose an approach for automatically evaluating interactive OLAP explorations. This evaluation is based on the concept of query contribution, which represents the degree to which a query contributes to the success of an exploration, in terms of user experience. Intuitively, a query is contributive if it is related to an underlying information need, if it refines or generalizes previous queries, if it allows to investigate related data perspectives, if it returns new data not previously analyzed or allows to highlight unexpected data, briefly, if in some way it allows to increase user knowledge about the studied phenomenon. We remark that there is currently no formal and commonly agreed definition of query contribution, and that writing contributive queries can be seen as a form of procedural knowledge. Procedural knowledge is the knowledge about how to do something. Different from declarative knowledge, that is often verbalized, application of procedural knowledge may not be easily explained [4]. However, models exist to automatically evaluate procedural knowledge acquisition [5].

These observations motivate two important choices we made in our approach. First, we represent the procedural knowledge related to writing contributive queries as a supervised machine learning problem that automatically learns a model of query contribution. More precisely, we formalize the problem as a binary classification task. We start from a set of explorations over a data warehouse, where each query composing an exploration is both described with a set of features and analyzed by an expert. Each feature corresponds to a score and the expert analysis corresponds to a binary label. Using the expert label as the target variable, we use a classifier to build a model of query contribution as a linear combination of the query features. This model allows to not only identify whether a query contributes or not to an exploration, but also to estimate the degree of its contribution and to understand the relative importance of each feature. Our second choice is to give an overall score to the exploration that corresponds to the probability that the skill of writing contributive queries is mastered by the analyst. We use a classical model of skill acquisition, called Bayesian Knowledge Tracing [5], that estimates the probability that a skill is mastered from a collection of opportunities to use the skill. In our context, each query corresponds to an opportunity to contribute to the exploration. We note that a similar principle has been used recently to score sequences of book reviews [6].

Our approach builds upon our previous work that aimed at characterizing

focus in OLAP explorations [7]. In addition to using a model of skill acquisition for assessing explorations, we develop a robust method to produce the model of query contribution, adding extra features to the feature set defined in [7], studying feature correlation, and balancing datasets. We evaluate our approach on three different datasets, two real ones and an artificial one.

Many practical benefits of the proposed assessment technique can be envisioned. As it puts the user and their skills in the center of the data analysis activity, it can be seen as an important driver in the design of systems supporting Interactive Data Exploration [1], as well as the corner stone of the development of benchmarks for such systems [8, 9].

The outline of the paper is the following. Next section presents the formal background of this work, including the Bayesian Knowledge Tracing, OLAP explorations and linear SVM classification. Section 3 defines the query features used to model the contribution of a query to an exploration and Section 4 describes the procedure for learning this model. Section 5 details the application of Knowledge Tracing in our context to assess explorations quality. Section 6 presents the tests done to validate our approach. Section 7 discusses related work. Finally, Section 8 concludes and draws future work.

2 Background

This section presents the concepts underlying our approach. We start by presenting the Knowledge Tracing model for skill acquisition. We formalize the notion of OLAP explorations. Finally, we describe the classification scheme used.

2.1 Bayesian Knowledge Tracing

As mentioned in the introduction, procedural knowledge is the knowledge about how to do something, which application may not be easily explained [4]. Many models exist to evaluate procedural knowledge acquisition. One of the most popular and successful models is Bayesian Knowledge Tracing [5]. An individual's grasp of the procedural knowledge is expressed as a binary variable, L, expressing whether the corresponding skill has been mastered or not. The knowledge of an individual cannot be directly observed, but can be induced by the individual answering a series of questions (or opportunities to exercise the skill) to guess the probability distribution of knowledge mastering. Measuring the skill mastery is noted $P(L_i)$, which corresponds to the probability that the skill L is mastered after answering i questions. Observation variables, X_i, are also binary: the answer to the question is either correct and wrong.

Specifically, the Knowledge Tracing model has four parameters, namely, two learning parameters, $P(L_0)$ and $P(T)$, and two performance parameters, $P(G)$ and $P(S)$. $P(L_0)$ is the probability that the skill has been mastered before answering the questions. $P(T)$ is the knowledge transformation probability: the probability that the skill will be learned at each opportunity to use the skill (i.e., the transition from not mastered to mastered). $P(G)$ is the probability

of guessing: in the case of knowledge not mastered, the probability that the individual can still answer correctly. $P(S)$ is the probability to slip, i.e., to fail while the skill is already mastered. The model uses these parameters to calculate the learning probability after each question to monitor individual's knowledge status and predict their future learning probability of knowledge acquisition using a Bayesian Network.

The probability that the skill L at opportunity n is mastered, $P(L_n)$, is the probability the skill is learned at step $n-1$, or not learned at step $n-1$ but learned at this step n:

$$P(L_n|X_n = x_n) = P(L_{n-1}|X_n = x_n) + (1 - P(L_{n-1}|X_n = x_n)) \times P(T) \quad (1)$$

where:

$$P(L_{n-1}|X_n = 1) = \frac{P(L_{n-1})(1 - P(S))}{P(L_{n-1})(1 - P(S)) + (1 - P(L_{n-1}))P(G)} \quad (2)$$

$$P(L_{n-1}|X_n = 0) = \frac{P(L_{n-1})P(S)}{P(L_{n-1})P(S) + (1 - P(L_{n-1}))(1 - P(G))} \quad (3)$$

with $X_n = 1$ (resp. 0) means problem n has been solved (resp. not solved).

Due to its predictive accuracy, Corbett and Anderson's Bayesian Knowledge Tracing is one of the most popular models. However, several challenges, including identifiability, local minimum, degenerate parameters and computational costs during fitting, still exist. Hawkins et al. proposed a fitting method avoiding these problems while achieving a similar predictive accuracy, and evaluated it against one of the most popular fitting methods: Expectation-Maximization (EM) [10]. In this extension, the parameters are fitted by estimating the most likely opportunity at which each individual learned the skill. Learner's performance is thus annotated with an estimate of when the skill is learned, assuming that a known state can never be followed by an unknown state. This annotation is used to construct knowledge sequences, that when compared with the actual performance sequence allows to empirically derivate the model's four parameters.

As aforementioned, traditionally, the performance of an individual is presented in binary value, correct or wrong, which does not account for all the cases of skill learning situation. Wang et al. proposed to extend the Knowledge Tracing model by replacing the discrete binary performance node with continuous partial credit node [11]. In this extension, it is assumed that Guess and Slip follow two Gaussian distributions, that are described respectively by their means and standard deviations. Prediction of the performance node also follows a Gaussian distribution, in which the mean value is used for the prediction. Noticeably, the standard deviation contains the information of how good the prediction is. Experiments with this extension show that by relaxing the assumption of binary correctness, the predictions of an individual's performance can be improved.

These two improvements of the Knowledge Tracing model (in the fitting method and the use of partial credits) were used successfully in sequencing educational content to students [12]. We conclude this subsection by noting that other models exist for predicting a learner's skill. Specifically, Performance Factor Analysis [13] uses standard logistic regression with the student performance as dependent variable. Interestingly, it is shown in [14] that Knowledge Tracing can achieve comparable predictive accuracy as Performance Factor analysis. Finally, Deep Knowledge Tracing [15] uses Recurrent Neural Networks to model student learning, with the advantage of not having to set explicit probabilities for slip and guess. However these models need very large datasets to learn the latent state from sequences, and most importantly, the encoding of the input vectors depends on an upper bound on the number of exercises which does not directly fit our context.

2.2 OLAP explorations

In order to keep the formalism simple, we consider cubes under a ROLAP perspective, described by a star schema [16]. For convenience, we consider that a dimension consists of a unique hierarchy, and we consider simple hierarchies without branches, i.e., consisting of chains of levels.

Let $\mathcal{A}$ be a set of attributes called levels, and for $L \in \mathcal{A}$, a member is an element of $Dom(L)$. Given two levels L_j and L_k, we define $Rollup(L_j) = L_k$ and $Drilldown(L_k) = L_j$ if there exists a functional dependency $L_j \rightarrow L_k$. A hierarchy h_i is a set $Lev(h_i) = \{L_0, \ldots, L_d\}$ of levels together with a *roll-up* total order $\succeq_{h_i}$ of $Lev(h_i)$, which is such that, for any L_j and L_k in $Lev(h_i)$, $L_k \succeq_{h_i} L_j$ if $Rollup(L_j) = L_k$. For each hierarchy h_i, the bottom level L_0 of the order determines the finest aggregation level for the hierarchy. Conversely, the top level L_d has a single possible value and determines the coarsest aggregation level.

Definition [Multidimensional Schema] A multidimensional schema (or, briefly, a schema) is a triple $\mathcal{S} = \langle \mathcal{A}, \mathcal{H}, \mathcal{M} \rangle$ where:

- $\mathcal{A}$ is a finite set of levels, whose domains are assumed pairwise disjoint,
- $\mathcal{H} = \{h_1, \ldots, h_n\}$ is a finite set of hierarchies, such that $\{Lev(h_1), \ldots, Lev(h_n)\}$ defines a partition of $\mathcal{A}$;
- $\mathcal{M}$ is a finite set of measure attributes, each $M \in \mathcal{M}$ being defined on a numerical domain $Dom(M)$.

A group-by set includes one level for each hierarchy, and defines a possible way to aggregate data. A reference (or coordinate) of a group-by set is a point in the n-dimensional space defined by the levels in that group-by set.

Definition [Group-by Set, reference, cells] Given a schema $\mathcal{S} = \langle \mathcal{A}, \mathcal{H}, \mathcal{M} \rangle$, $\mathcal{H} = \{h_1, \ldots, h_n\}$, let $Dom(\mathcal{H}) = Lev(h_1) \times \ldots \times Lev(h_n)$; each $G \in Dom(\mathcal{H})$ is called a group-by set. Let $G = \langle a_{k_1}, \ldots, a_{k_n} \rangle$ and $Dom(G) = Dom(a_{k_1}) \times \ldots \times Dom(a_{k_n})$; each $g \in Dom(G)$ is called a reference (or a coordinate) of G. A cell is a tuple $c = \langle g, M, m \rangle$ where g is a reference over $\mathcal{S}$, $M \in \mathcal{M}$ is a measure and

$m \in dom(M)$ is the value of the measure. An empty cell is a cell whose value is equal to $Null$.

A cube instance $\mathcal{I}$ over a schema $\mathcal{S}$ is defined as a set of non empty cells over $\mathcal{S}$.

The queries considered in this paper are multidimensional queries modeled as a collection of fragments extracted from the query expression, as in [17].

Definition [OLAP query] A *query* over schema $\mathcal{S} = \langle \mathcal{A}, \mathcal{H}, \mathcal{M} \rangle$ is a triple $q = \langle G, P, M \rangle$ where:

1. $G \in Dom(H)$ is the query group-by set;
2. $P = \{p_1, \ldots, p_k\}$ is a set of Boolean predicates, at most one for each hierarchy, whose conjunction defines the *selection predicate* for q; they are of the form $l = v$, or $l \in V$, with l a level, v a value, V a set of values.
3. $M \subseteq \mathcal{M}$ is the measure set whose values are returned by q.

In practice, it is useful to handle only the detailed levels of a group by set (i.e. excluding the ones that are top levels in some hierarchies). In addition, in predicates of the form $l \in V$, it is useful to handle the individual filters of the form $l = v$ for all $v \in V$. We define the functions $Levels$ and $Filters$ for simplifying both cases, as follows: Given a group by set $G = \langle a_{k_1}, \ldots, a_{k_n} \rangle$ of a query over a schema $\mathcal{S} = \langle \mathcal{A}, \mathcal{H}, \mathcal{M} \rangle$, let $Levels(G) = \{L_j \in \{a_{k_1}, \ldots, a_{k_n}\} | \exists L_k \in \mathcal{A}, L_k \succeq_{h_i} L_j\}$. Given a set of predicates P of a query over a schema $\mathcal{S}$, let $Filters(P) = \{"l = v" \in P\} \cup \{"l = v" | "l \in V" \in P, v \in V\}$.

Given a cube instance $\mathcal{I}$, the answer to a query q, denoted $answer(q)$, is the set of non empty cells of $\mathcal{I}$ whose coordinates are defined by the query group by set, the selection predicates and the measures. In a given query result, we distinguish between base cells and aggregated cells. Base cells are tuples $\langle \langle l_1, \ldots, l_n \rangle, m, v \rangle$ where the l_i are taken in $Dom(L_i^0)$ for all i (L_i^0 being the bottom level of the ith hierarchy), m is a measure and and v is a value. Aggregated cells are tuples $\langle \langle l_1, \ldots, l_n \rangle, m, v \rangle$ where there exists one l_i not in $Dom(L_i^0)$, m is a measure and v is a value. In what follows, we note $accessArea(q)$ the set of base cells used by a query q to produce $answer(q)$.

Finally, we define explorations to be sequences of OLAP queries.

Definition [Exploration] Let $\mathcal{I}$ be a cube instance over a schema $\mathcal{S}$. An exploration s over $\mathcal{I}$ is a triple $\langle e, bts, ats \rangle$, where $e = \langle q_1, \ldots, q_p \rangle$ is a sequence of p OLAP queries over $\mathcal{S}$, bts is a function that gives for a query the timestamp before its execution and ats a function that gives for a query the timestamp after its execution. We note $q \in e$ if a query q appears in the exploration e.

2.3 Linear SVM classification

In this work, we formalize the problem of modeling the contribution of a query to an exploration as a supervised classification task. Assuming that queries are described with a set of features f_i, our objective is to learn a linear combination of the features that separates contributing queries from non contributing ones. Restricting to linear combination allows for the model to remain interpretable, while adopting a supervised approach results in a model indirectly embedding human expertise.

The expected contribution model is, for a query q:

$$contrib(q) = w_0 + \sum_{i=1}^{n} w_i f_i(q) \tag{4}$$

where n is the number of features, w_0 is the bias, w_i is the weight of feature i and $f_i(q)$ is the value for feature i for query q. This formula represents the estimated contribution of query q.

An SVM classifier learns the weights $w_0 \dots w_n$ by considering two sets of data (in our case, the set of contributive queries and the set of non-contributive queries) as n-dimensional vectors, that should be separated with a (n-1)-dimensional hyperplane. The hyperplane is constructed so that the distance between the hyperplane and the nearest point from either group is maximized. In our case, the resulting hyperplane is our model $contrib(q)$, which can be used to classify any query q based on its features values.

Both $contrib(q)$ and each weight w_i can be interpreted according to two aspects that we call *polarity* and *intensity*. The polarity of $contrib(q)$ is represented by its sign and indicates the class of q, i.e., a query that contributes (resp. does not contribute) to the exploration. The intensity of $contrib(q)$ is represented by its absolute value, and indicates the intensity to which q contributes or not to the exploration. The polarity of a weight w_i is represented by its sign and indicates the impact of f_i on $contrib(q)$. A positive (resp. negative) value of w_i means that an increase of f_i will increase (resp. decrease) $contrib(q)$. The intensity of w_i is represented by its absolute value and indicates the sensitivity of $contrib(q)$ regarding f_i.

3 Descriptive model of OLAP queries

In this section, we define the set of features used in our model of query contribution. We represent an OLAP query as a set of simple descriptors computed from the query and its context, and covering a maximum of aspects of a query. For the sake of presentation, we categorize these descriptors as follows: i) intrinsic descriptors, i.e., only related to the query itself, ii) relative descriptors, i.e., also related to the query's predecessor in the exploration, and iii) contextual descriptors, i.e., related to the whole exploration, providing more context to the descriptor. Table 1 presents an overview of descriptors.

For all of the definitions given in this section, let $q_k = \langle G_k, P_k, M_k \rangle$ be the query occurring at position k in exploration e over instance $\mathcal{I}$ of schema $\mathcal{S}$, and let $\mathcal{Q}$ (resp. $\mathcal{E}$) be the set of all possible queries (resp. explorations) over $\mathcal{S}$. All the queries we considered are supposed to be well formed, and so we do not deal with query errors. Also, for the sake of readability, we do not include the schema $\mathcal{S}$ and instance $\mathcal{I}$ as descriptors inputs, while actually they are implicitly used in some descriptors.

Intrinsic descriptors	
NoM	Number of measures, that appear in the query text
NoL	Number of levels, in the group-by set
NoF	Number of filters (filtering predicates)
LDepth	Level depth (depth of each group-by set level in its hierarchy)
FDepth	Filter depth (depth of levels appearing in filters)
NoC	Number of cells, in query answer
QoI	Quantity of information, contained in query answer
ExecTime	Execution time
Relative descriptors	
NCM	Number of common measures, with previous query
NCL	Number of common levels, with previous query
NCF	Number of common filters, with previous query
RED	Relative edit distance (effort to express a query starting from the previous one)
RI	Relative identity (whether the query is identical to the previous one)
RR	Relative recall (recall of cells in query answer w.r.t. previous answer)
RP	Relative precision (precision of cells in query answer w.r.t. previous answer)
IsRefine	Is refinement (whether the query is a refinement of the previous one)
IsRelax	Is relaxation (whether the query is a relaxation of the previous one)
Contextual descriptors	
CpQ	Clicks per query (number of successor queries that differ in at most 1 operation)
ClickDepth	Click depth (length of the sequence of successor queries that differ in at most 1 operation)
IVA	Increase in view area (number of cells in query answer that were not seen previously)
NoQ	Number of queries (absolute position of the query)
QRP	Query relative position
ElapsedTime	Elapsed time (from the starting of the exploration)
QF	Query frequency (number of queries per unit of time)
ConsTime	Consideration time (time spent in analyzing query answer)

Table 1: Query descriptors

3.1 Intrinsic descriptors

Intrinsic descriptors are those that can be computed only considering the query q_k, independently of the exploration e and other queries in e. In other words, these descriptors will give the same score to q_k, independently of e.

Number of Measures -. $NoM(q_k)$ represents the number of measures that explicitly appears in the text of q_k. If no measure has been explicitly written by the user, the default behavior of OLAP systems is to use a default measure. In that case, $NoM(q_k)$ is equal to 1.

$$NoM(q_k) = \begin{cases} card(M_k) & \text{if } M_k \neq \emptyset \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

Number of Levels -. $NoL(q_k)$ represents the number of levels (excluding top levels) in the group-by set G_k. By definition, G_k contains at most one level per hierarchy.

$$NoL(q_k) = card(Levels(G_k)) \tag{6}$$

Number of Filters -. $NoF(q_k)$ represents the number of filtering predicates in P_k.

$$NoF(q_k) = card(Filters(P_k)) \tag{7}$$

Level Depth -. $LDepth(q_k)$ measures the granularity of q_k in terms of the depth of each level in its hierarchy. It can be seen as the number of drills down necessaries for obtaining G_k from the most aggregated group-by set. Let $depth(l_i)$ be the depth of level l_i in the hierarchy h_i to which it belongs (ranging from 0 if l_i is the top level of h_i to $card(Lev(h_i)) - 1$ if l_i is the bottom level of h_i):

$$LDepth(q_k) = \sum_{l_i \in Levels(G_k)} depth(l_i) \tag{8}$$

Filter Depth -. $FDepth(q_k)$ measures the filtering granularity of q_k, i.e. the depth of levels appearing in filters.

$$FDepth(q_k) = \sum_{"l_i = v_i" \in Filters(P_k)} depth(l_i) \tag{9}$$

Number of Cells -. $NoC(q_k)$ represents the number of cells in $answer(q_k)$.

$$NoC(q_k) = card(answer(q_k)) \tag{10}$$

Quantity of Information -. $QoI(q_k)$ measures the quantity of information contained in $answer(q_k)$. We measure it with an entropy inspired metric, computed as follows:

$$QoI(q_k) = 1 - (interest(answer(q_k))) \tag{11}$$

where $interest(C)$ measures the interestingness degree of a set of cells C as a simple normalized entropy, defined by:

$$interest(C) = \frac{(-\sum_{i=1}^{m} p(i) \log(p(i)))}{\log(m)} \tag{12}$$

with $|C| = m$, $C(i)$ is the i^{th} value of the set C and $p(i) = \frac{C(i)}{\sum_{i=1}^{m} C(i)}$

Execution Time -. $ExecTime(q_k)$ is a measure of the computation time for q_k. *ExecTime* assumes all the queries are executed on the same system, with the same technical performances.

$$ExecTime(q_k) = ats(q_k) - bts(q_k) \tag{13}$$

where $ats(q_k)$ and $bts(q_k)$ are respectively the timestamps after and before q_k execution.

3.2 Relative descriptors

Relative descriptors are those that are computed comparing the query q_k to the previous query in the exploration e. Let $q_{k-1} = \langle G_{k-1}, P_{k-1}, M_{k-1} \rangle$ be the previous query, being undefined for the first query of e (i.e. q_1). Each descriptor provides a default score for this limit case.

Number of Common Measures -. $NCM(q_k, q_{k-1})$ counts the number of common measures of q_k relatively to q_{k-1}. The default value is 0 for q_1.

$$NCM(q_k, q_{k-1}) = \begin{cases} card(M_k \cap M_{k-1}) & \text{if } k > 1 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Number of Common Levels -. $NCL(q_k, q_{k-1})$ counts the number of common levels of q_k relatively to q_{k-1} (excluding top levels). The default value is 0 for q_1.

$$NCL(q_k, q_{k-1}) = \begin{cases} card(Levels(G_k) \cap Levels(G_{k-1})) & \text{if } k > 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Number of Common Filters -. $NCF(q_k, q_{k-1})$ counts the number common filters of q_k relatively to q_{k-1}. The default value is 0 for q_1.

$$NCF(q_k, q_{k-1}) = \begin{cases} card(Filters(P_k) \cap Filters(P_{k-1})) & \text{if } k > 1 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Relative Edit Distance -. $RED(q_k, q_{k-1})$ represents the edition effort, for a user, to express the current query starting from the previous one. It is strongly related to OLAP primitives operations, and computed as the minimum number of atomic operations between queries, by considering the operations of adding/removing a measure, drilling up/down, and adding/removing a filter. The considered cost for each observed difference (adding/removing) is the same.

$$\begin{aligned} &RED(q_k, q_{k-1}) = card(M_k - M_{k-1}) + card(M_{k-1} - M_k) \\ &+ nbDrills(G_k, G_{k-1}) + card(Filters(P_k) - Filters(P_{k-1})) \\ &+ card(Filters(P_{k-1}) - Filters(P_k)) \end{aligned} \tag{17}$$

where $nbDrills(G_k, G_{k-1}))$ denotes the number of changes in query aggregation level (drills). $RED(q_k, q_{k-1}) = 0$ if $k = 1$.

Relative Identity -. $RI(q_k, q_{k-1})$ checks whether q_k is identical to q_{k-1}, i.e. if both queries have the very same measures, group-by set and predicates.

$$RI(q_k, q_{k-1}) = \begin{cases} 1 & \text{if } q_k = q_{k-1} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Relative Recall -. $RR(q_k, q_{k-1})$ is computed as a classical recall of cells in $answer(q_k)$ w.r.t. cells in $answer(q_{k-1})$. We give a default neutral score of 0.5 for q_1

$$RR(q_k, q_{k-1}) = \begin{cases} \frac{answer(q_k) \cap answer(q_{k-1})}{answer(q_{k-1})} & \text{if } k > 1 \\ 0.5 & \text{otherwise} \end{cases} \tag{19}$$

Relative Precision -. $RP(q_k, q_{k-1})$ is computed as a classical precision of cells in $answer(q_k)$ w.r.t. cells in $answer(q_{k-1})$. We give a default neutral score of 0.5 for q_1

$$RP(q_k, q_{k-1}) = \begin{cases} \frac{answer(q_k) \cap answer(q_{k-1})}{answer(q_k)} & \text{if } k > 1 \\ 0.5 & \text{otherwise} \end{cases} \tag{20}$$

Is Refinement -. $IsRefine(q_k, q_{k-1})$ checks whether q_k is a refinement of q_{k-1}. Intuitively, a refinement happens when the access area of q_k is included in the access area of q_{k-1}. When a user first opens a system for starting an exploration, the default query is the one that aggregates the whole cube. Therefore, for q_1, we give a default score of 1.

$$IsRefine(q_k, q_{k-1}) \begin{cases} 1 & \text{if } k = 1 \\ 1 & \text{if } (accessArea(q_k) \subseteq accessArea(q_{k-1})) \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Is Relaxation -. $IsRelax(q_k, q_{k-1})$ checks whether q_k is a relaxation of query q_{k-1}. This happens when the access area of q_k includes the access area of q_{k-1}. For the same reason as above, for q_1, we give a default score of 0.

$$IsRelax(q_k, q_{k-1}) = \begin{cases} 0 & \text{if } k = 1 \\ 1 & \text{if } (accessArea(q_k) \supseteq accessArea(q_{k-1})) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

3.3 Contextual descriptors

Contextual descriptors are exploration dependent and make sense only in the context of an exploration. The same query q_k occurring in different explorations may be given different scores for descriptors in this category.

Clicks Per Query -. CpQ is a classical measure used in web search to evaluate search engines through their Search Engine Results Pages (SERP). Given a SERP, CPQ represents the number of links in this page that have been clicked by the user. We adapt it by considering a click as obtaining a new query that differs in one operation from the current query. This model allows to capture typical user behaviors in front of OLAP systems. Formally, we measure $CpQ(q_k, e)$ by counting the number of queries occurring after q_k in the exploration e that are at a distance of at most 1 operation from q_k (measured in terms of relative edit distance - RED, as defined in Subsection 3.2).

$$CpQ(q_k, e) = card(\{q_p \in e \mid p > k \wedge RED(q_k, q_p) \leq 1\}) \tag{23}$$

Click Depth -. In web search, *ClickDepth* evaluates the number of pages visited by successively following hyper links, starting from the SERP and ending when the user breaks the chain by, for example, submitting new keywords. We adapt *ClickDepth* by calculating, the length of the query sequence starting from q_k such as each query is at a distance of 1 from its predecessor.

$$ClickDepth(q_k, e) = length(S_{kp}) \tag{24}$$

where S_{kp} is the longest subsequence of queries in exploration e starting at query q_k and ending at query q_p inclusive, such that $\forall q_i, q_{i+1} \in S_{kp}, RED(q_i, q_{i+1}) \leq 1$.

Increase in View Area -. $IVA(q_k, e)$ characterizes the increase in terms of new cells in $answer(q_k)$ compared to all the cells seen during the previous queries of the exploration.

$$IVA(q_k, e) = \frac{card(answer(q_k) \setminus \bigcup_{i \in [1,k-1]} answer(q_i))}{card(\bigcup_{i \in [1,k]} answer(q_i))} \tag{25}$$

Number of Queries -. $NoQ(q_k, e)$ counts the absolute position of query q_k in e.

$$NoQ(q_k, e) = k \tag{26}$$

Query Relative Position -. $QRP(q_k, e)$ measures the position of query q_k relatively to the exploration e. It is computed as the rank of the query in the exploration, normalized by the size of the exploration:

$$QRP(q_k, e) = \frac{k}{length(e)} \tag{27}$$

Elapsed Time -. $ElapsedTime(q_k, e)$ measures the absolute time elapsed from the starting of e to the time right after the execution of q_k.

$$ElapsedTime(q_k, e) = ats(q_k) - bts(q_1) \tag{28}$$

where $ats(q)$ and $bts(q)$ are respectively the timestamps right after and right before the execution of q.

Query Frequency -. $QF(q_k, e)$ measures the number of queries submitted per unit of time.

$$QF(q_k, e) = NoQ(q_k)/ElapsedTime(q_k) \tag{29}$$

Consideration Time -. $ConsTime(q_k, e)$ measures how long a user has spent in analyzing $answer(q_k)$. We compute it as the time period between the end of the query execution and the submission of the next query. We fixed $ConsTime$ to a neutral value (the average of all the previous considerations times) for the last query of the exploration.

$$ConsTime(q_k, e) = \begin{cases} lts(q_{k+1}) - ets(q_k) & \text{if } k < length(e) \\ avg(\{lts(q_{k+1}) - ets(q_k) \mid & \\ k \in [1, length(e) - 1]\}) & \text{otherwise} \end{cases} \tag{30}$$

4 Model of query contribution

This section presents the procedure for learning the model of query contribution. The 25 features described in Section 3 provide a quantitative description of the queries of an OLAP exploration. Our model is constructed using a supervised classification approach based on this description. To keep the model understandable, we chose a model representing the contribution of a query as a linear combination of the features describing the query.

4.1 Normalization and selection of the set of features

We first remark that these features have highly different magnitudes. For instance, *QRP* (Query Relative Position) scores are normalized in $[0, 1]$, while *ConsTime* (Consideration Time) scores have no theoretical limit. Such differences have an undesired impact on the weights that the classifier learns, since for instance a feature with higher values may be penalized by the classifier with a low weight. As a result, we would loose the interpretability of features weights, which is not desirable. In order to harmonize the feature scores, we assume that these scores are normally distributed, and we use the *z-score*, which is the signed number of standard deviations by which the value of an observation exceeds the mean value of what is being measured. Formally, let X be a set of elements, μ its mean, σ its standard deviation, and x an element of X. The z-score of x, denoted $z(x)$, is computed as:

$$z(x) = \frac{x - \mu}{\sigma} \tag{31}$$

Each feature score is then replaced with its z-score.

To validate our choice of features, we study the pairwise correlation of the features and perform feature selection. For the latter, we use an automatic recursive feature elimination procedure, with a 10-fold cross validation over the entire dataset, and accuracy as a target. The least important features, in terms of the weights learned, are progressively pruned and the accuracy is computed for all resulting feature-sets. The optimal number of features corresponds to the set achieving the best accuracy.

4.2 Learning the classifier

To train and evaluate our classifier, we asked experts to independently label explorations, by deciding for each query if it contributes (label C) or not (label N) to its exploration. For the sake of simplifying the experts task and to be less sensitive to experts subjectivity, we restrict the annotation task to binary labels. In order to obtain reliable labels, when labels are given by 2 experts, we computed the agreement between experts. We then include in the training dataset only the queries on which the experts agreed.

In case of unbalanced dataset (one class of queries being significantly more present in the dataset than the other), we compared several methods on the basis of their respective accuracy and F1-measure, over a 10-fold cross-validation: either random undersampling of majority class, or oversampling of minority class. In the last case, several heuristics have been tested: random oversampling, 3 variants of SMOTE (with different approaches to sample borderline points between classes) or ADASYN [18]. We finally favor oversampling to ensure the model is learned over a balanced set of classes while not pruning too much the size of our datasets.

To learn our model, we trained a linear SVM classifier on the dataset of queries described by their label and the set of select features, standardized using a z-score. When testing over a single dataset, the model is learned on 80% of the corpus, using 10-fold cross validation to choose its best hyperparameter. We kept the remaining 20% as a test set for model evaluation. When cross-testing with two datasets, the model is learned using 10-fold cross validation over the entire first dataset and then tested over the entire second one.

5 Assessing the overall quality of an exploration

We recall the primary hypothesis underlying our approach: the quality of an exploration can be assessed by measuring to what extent the skill of writing queries that contribute to the exploration is mastered by the user.

The model presented in the previous section allows to give a score $contrib(q)$ to each query q of an exploration e. This score can be interpreted as the contribution of the query q to its exploration. Each exploration e can then be seen as a sequence of scores $contrib(q)$ for each q in e. In other words, each step of an exploration can be treated as an opportunity to exercise the skill of writing a contributive query. Therefore, a Knowledge Tracing (KT) model can be used, based on these contribution scores, to predict the skill of writing contributive queries for a specific user. To do so, as our $contrib(q)$ scores are real valued, we propose to use the extension of the KT model to continuous partial credits [11]. Other, more simple, modeling choices could have been made, such as setting a binary KT model based on the contribution scores polarity alone, but as recalled in Section 2, the continuous KT extension has been proven to evaluate skills more precisely than the binary KT.

We recall from Section 2 that in this extension, $P(G)$ and $P(S)$ are assumed to follow a Gaussian distribution, and as such, these two quantities are represented by a mean value and a standard deviation. As a consequence, and as

opposed to the binary KT, the prediction $P(L_n)$ also follows a Gaussian distribution, whose mean is used as the value of the prediction and whose standard deviation expresses the confidence attached to this prediction.

To learn the 6 parameters of the continuous KT, we extend the approach proposed by Hawkins et al. [10] so that it outputs estimates of $P(G)$ and $P(S)$ described by a mean and a standard deviation. Then, based on these 6 parameters, the estimation of each skill acquisition $P(L_n)$ is performed by running 100 tests with each time randomly generated values for $P(G)$ and $P(S)$ following their respective distribution. From these 100 $P(L_n)$ estimates, we compute a mean and a standard deviation following the normal hypothesis. In the end, the mean $P(L_n)$ is the overall score of the exploration and the standard deviation is the confidence in this prediction.

It is important to note that we apply the KT on each exploration independently, even if the KT parameters are learned from a representative set of explorations.

6 Experiments

6.1 Experimental setup

The experiments have been conducted on three datasets, two with real data and real explorations, and one with artificial data and artificial explorations.

*Real explorations on **open data**.* The first dataset consists of navigation traces collected in the context of a French project on energy vulnerability. To constitute our corpus, we used three cubes instances built from open data, concerning expenses in mobility and heating. In the main cube, called *MobPro*, facts represent people trips between home and workplace, and dimensions allows to characterize a trip according to various characteristics of the worker (e.g. age, gender, level of studies), home (e.g. location, family size), job (e.g. location, branch), transport mode, traveled distance, energy used, etc. The cube is organized as a star schema with 19 dimensions, 68 (non-top) levels, and 24 measures. 37,149 trips are recorded in the fact table. The other cubes are organized in a similar way.

To obtain explorations, we logged the OLAP sessions of 8 volunteer students of a Master's degree in Business Intelligence, answering some fuzzy information needs defined by their lecturer. Students were asked to find some interesting behaviors and to perform the exploration. However, students were not aware that skills acquisition tests would be performed on the basis of their anonymized queries, not to perturbate their behavior and bias our experiment. During their task, students investigated some relations among data, for example, *"Which profiles of workers, having low revenues, expend the most in mobility"* and tested some popular hypothesis like *"Executives make longer home-work trips than people with other professions"*. To explore the cube, the students used Saiku[1],

[1] http://meteorite.bi/products/saiku

a web application that allows to navigate OLAP databases in a user-friendly manner, and generates MDX code from graphical manipulations. The students were quite familiar with this OLAP tool, but not necessarily with the data in the cube.

From this experiment, we could gather 39 explorations from the system logs. In total, these explorations represent 1608 queries, with an average of 41 queries per exploration. A particularity of some third party OLAP tools, like Saiku, is that their user interface submits a new query for each user action (including intermediate drag-and-drops), resulting in very long explorations in the log. Nevertheless, there were some extremely short explorations (7 explorations counting less than 10 queries), which mainly correspond to incomplete studies.

A subset of 1114 queries were labeled independently by two experts (interns working on OLAP exploration under the supervision of a lecturer), using a labeling tool specifically designed for that purpose. Both experts agreed at $67,59\%$ on the 1114 queries, which represents a total of 753 queries. On the agreed queries, 649 have been labeled C (contributive) and 104 N (non contributive). In order to avoid any side effect due to the classes distribution, we balanced our dataset using oversampling, as explained in Section 4.

Real explorations on ***enterprise data****.* The second dataset consists in navigation traces of 14 volunteers of SAP in the context of a previous study on discovering user intents [19]. Analysts covered a range of skills in data exploration, classed, based on their position in the company, in two expertise groups: beginners and expert users. We set 10 business needs (named Q_1 to Q_{10}), each corresponding to a specific user interest. Users were asked to analyze some of the 7 available data sources to answer each of the 10 business needs, using a SAP prototype that supports keyword-based BI queries[2]. The business needs were grouped in different business cases like: *"For each European country, detect which genres of films did not reach the expected sales"* or *"In which Income Group would you classify a candidate country with a GDP of $6 billion?"*.

As users enter keywords, the BI system suggests, on the fly, further tokens to complete the current ones, letting the user choose among them, as in web search engines. The underlying idea is that a suggestion completes the original BI question in order to obtain a well-formed query over a database. Then, conversely to Saiku tool, SAP prototype only evaluates final queries, after all keywords were entered and a formal query was selected. In this context, average exploration length is around 5 queries. Another difference on this dataset, is the absence of timestamps in the log. We re-executed the queries in order to compute execution times, but we have no information about consideration time, which had to be excluded from the model. Finally note that elapsed time is only computed in the basis of execution time. All these technical differences, in addition to working with different types of users and different information needs,

[2]Patent Reference: 14/856,984 : BI Query and Answering using full text search and keyword semantics

provide a good opportunity for testing our approach in different configurations. To this aim, our experiments also discuss the ability of our model to adapt to such configurations.

In total, this dataset contains 103 user explorations, accounting for 525 queries. Table 2 describes, for each business need, its difficulty, estimated by an expert (in terms of time, number of queries and exploited sources expected in its resolving), the number of explorations devised for solving it, the number of queries and the number of queries perceived as relevant by users in their own activity. For the sake of confidentiality, the 525 queries were labeled by only one expert, who is one of the authors of this paper. Interestingly, 52% of queries were labeled C (contributive), which results in no need for balancing in this case.

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}
Difficulty	low	med	med	med	low	high	low	low	med	high
Nb. interactions	19	11	10	10	10	8	9	9	9	8
Nb. queries	84	65	58	41	50	43	58	51	26	49
Nb. relevant queries	34	26	30	16	26	10	27	24	24	9
Queries / interaction	4.4	5.9	5.8	4.1	5.0	5.4	6.4	5.7	2.9	6.1
Relevant queries / interaction	1.8	2.4	3.0	1.6	2.6	1.25	3.0	2.7	2.7	1.1

Table 2: Analysis of business needs

Artificial *explorations.* The third dataset, with artificial data, comes from the Star Schema Benchmark [20], and was used with artificial explorations. The Star Schema Benchmark (SSB) is a variation of TPC-H, a popular benchmark from the Transaction Processing Performance Council (TPC). SSB cube consists of a relational database under the form of a star schema, with one fact table and 4 dimension tables. We used the SSB generator to generate data with a scale factor of 1, corresponding roughly to 1 Gb of data. We used the PDGF data generator [21] to generate a realistic instance with skew data.

Instead of using the rather limited SSB workload, we generated artificial explorations using CubeLoad [22], a tool for generating realistic explorations over star schemas. CubeLoad takes as input a cube schema and creates the desired number of explorations according to templates modeling various user exploration patterns. Templates available in Cubeload simulate: (*Goal Oriented*) users with limited OLAP skills pursuing a specific analysis goal, (*Slice And Drill* and *Slice All*) more advanced users navigating with a sequence of

slice and/or drill operations, (*Exploratory*) users tracking unexpected results with exploratory sessions. We generated a collection of 100 explorations, each one containing between 30 and 70 dimensional queries. Noticeably, explorations were generated before the generation of the cube instance, which enabled us to take advantage of the most frequent selection predicates in the explorations to produce data skew in the most queried zones of the cube.

Environment. All experiments were run on a 64 bits Windows 8.1 Operating System, featuring a Intel(R) Xeon(R) CPU E3-1241 v3 @3.50GHZ and 16GB of RAM. Cubes are stored using the MonetDB database for SSB and SQL Server for MobPro. Our prototype is written in Java 8 and Python 3, with Scikit-learn and Imbalanced-learn [23] packages.

6.2 Feature scores computation

Our first experiment consists in computing the feature scores for all queries in both open and enterprise datasets. In Tables 3 and 4, we report for each descriptor its range, average value and standard deviation. In Table 3 we also report the average time needed for the computation.

A first observation is that computation remains in interactive time, in the sense that it does not exceed few hundred milliseconds. In average, the computation of all the feature scores for a query is 152 milliseconds, which is low, especially given that the average consideration time for a query result is $29,371$ milliseconds.

A second observation is the ranges of values that show important variations from one feature to another, which justifies the standardization done before training the SVM.

Concerning the comparison among datasets, there are some major differences in the range of feature values. First, students use more filters (features NoF, $FDepth$ et NCF) than analysts. This can be explained by the size of the cubes (bigger in the open dataset) and the nature of informations needs (student analyzed specific phenomena in portions of the cubes while analyst analyzed the overall data). Second, execution times are considerably higher in the open dataset because of the underlying BI system, which impacts features $ExecTime$, QF and $ElapsedTime$. Finally, Saiku interface causes explorations to be longer (feature NoQ) as previously discussed.

6.3 Feature correlation analysis

Our second test aimed at validating the choice of features. We analyzed the pairwise correlation of feature scores, using Pearson's correlation coefficient, over each corpus of queries independently. We report the result in the heat maps of Figure 1 that represents for each pair of features the correlation between them with a specific color code. As expected, correlation is maximal on the diagonal which represents the correlation of a feature with itself. Correlation values range between -1 (maximal negative correlation) and 1 (maximal positive correlation), with 0 indicating no correlation. In the heat map, the more intense the yellow

Feature	Min	Max	Avg	StdDev	CT (ms)	stdev(CT)
Intrinsic descriptors						
NoM	1	6	1.472	0.748	0.002	0.007
NoL	0	6	2.237	1.034	0.002	0.007
NoF	0	48	1.968	4.720	0.004	0.010
LDepth	0	12	3.536	1.972	3.849	2.978
FDepth	0	185	4.352	14.484	0.003	0.008
NoC	0	12438	156.756	662.932	33.814	57.829
QoI	0	1	0.221	0.200	34.056	58.765
ExecTime(ms)	0	82.061	0.465	3.391	0.001	0.004
Relative descriptors						
RED	0	47	1.479	3.170	2.045	2.135
NCM	0	5	1.326	0.808	0	0
NCL	0	6	1.981	1.113	1.868	1.947
NCF	0	48	1.637	4.125	2.085	2.001
RI	0	1	0.406	0.491	0.003	0.008
RR	0	1	0.573	0.469	68.574	117.447
RP	0	1	0.582	0.467	71.877	129.544
IsRefine	0	1	0.207	0.406	0.023	0.024
IsRelax	0	1	0.118	0.323	0.0217	0.033
Contextual descriptors						
CpQ	0	22	1.945	3.067	0.570	0.836
ClickDepth	0	23	3.219	4.144	9.179	9.765
IVA	0	1	0.005	0.042	152.792	181.939
NoQ	1	165	37.218	33.025	0.748	0
QRP	0.006	1	0.510	0.289	0.748	0
QF	0.005	1.970	0.076	0.153	0.001	0.005
ElapsedTime(ms)	0	6975.457	1207.484	1523.829	0.001	0.005
ConsTime(ms)	0	889178	29371	65622	0	0

Table 3: Range, average, standard deviation and computation time for the 25 features on the open dataset

color, the greater the positive correlation, the darker the blue color, the greater the negative correlation. Medium tones indicate correlation close to 0. As an example, in the symmetric correlation matrix used to plot the heat map for the open dataset, the highest value (diagonal excluded) is 0.95 while the lowest value is -0.54. Out of the 300 coefficients, only 10 values are lower than -0.3 and only 18 values are greater than 0.6.

A first general observation is that the vast majority of coefficients show no significant correlation. Interestingly, similar trends of correlation happen in both datasets. We detail the most significant correlations. Unsurprisingly, NoQ (number of Queries) is correlated with both ElpasedTime and QRP (query relative position). The features related to some specific fragments of a query are correlated, which was expected. For instance, the highest correlation is between NoF (number of filers) and FDepth (filter depth), and NoL (number of levels) is correlated with LDepth (level depth). Similarly, we also observed correlations

Feature	Min	Max	Avg	StdDev
Intrinsic descriptors				
NoM	1	5	1.170	0.441
NoL	0	3	1.023	0.645
NoF	0	3	0.720	0.712
LDepth	0	1	0.170	0.224
FDepth	0	5	0.146	0.343
NoC	0	7790	194.507	665.205
QoI	0	0.641	0.132	0.139
ExecTime(ms)	0.083	0.264	0.101	0.015
Relative descriptors				
RED	0	7	1.381	1.671
NCM	0	3	0.770	0.571
NCL	0	3	0.550	0.671
NCF	0	7	0.476	0.807
RI	0	1	0.220	0.402
RR	0	1	0.432	0.490
RP	0	1	0.341	0.414
IsRefine	0	1	0.164	0.370
IsRelax	0	1	0.084	0.277
Contextual descriptors				
CpQ	0	7	0.622	1.145
ClickDepth	0	13	0.851	1.550
IVA	0	1	0.041	0.149
NoQ	0	23	4.305	3.613
QRP	0.043	1	0.599	0.292
QF	1	12	8.199	3.633
ElapsedTime(ms)	0.083	2.655	0.442	0.384

Table 4: Range, average and standard deviation for the 24 features on the enterprise dataset

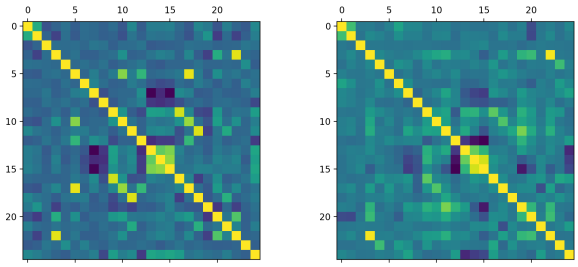
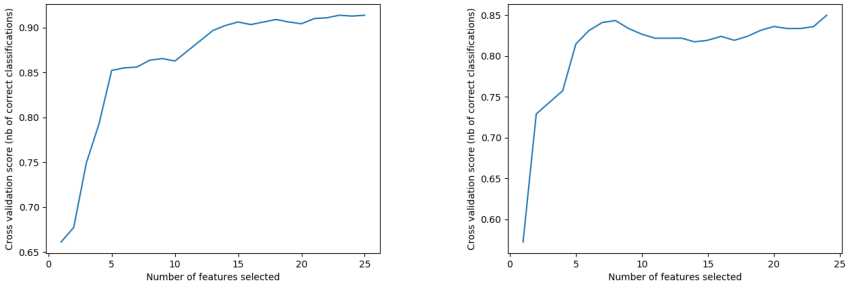


Figure 1: Feature correlation heat maps: (i) open dataset, (ii) enterprise dataset



open dataset enterprise dataset

Figure 2: Recursive selection of features for open (left) and enterprise (right) datasets. For each number of features the plot indicates the accuracy of the linear SVM classifier.

between relative descriptors and the intrinsic descriptors related to a specific query fragment. Finally, medium correlations can be observed between relative recall (resp., relative precision) and relative identity, which again was expected.

In general, we believe that this level of correlation is acceptable. Nevertheless, we experimented with a feature selection method, as explained in Section 4 and whose results are reported in Figure 2.

From Figure 2-left it can be seen that selecting and removing features from the 25 original features decreases the accuracy of the classifier in the case of the open dataset. This trend is clearly less pronounced for the enterprise dataset in Figure 2-right, that shows a decrease, but has a second high accuracy score for 8 features. This conducts to observe some variability for the enterprise dataset in the number of features to select. As a consequence and as in both datasets, the model with the whole set of features provides a high accuracy, we decide to keep all the features in the model in the experiments hereafter.

	Accuracy	Precision	Recall	Training set size
Random Over Sampler	0.883 ± 0.005	0.967 ± 0.004	0.897 ± 0.005	1034 ± 0
SMOTE regular	0.887 ± 0.006	0.972 ± 0.007	0.896 ± 0.005	1034 ± 0
SMOTE borderline1	0.887 ± 0.005	0.968 ± 0.006	0.901 ± 0.004	1034 ± 0
SMOTE borderline2	0.879 ± 0.008	0.973 ± 0.004	0.887 ± 0.009	1033.6 ± 0.49
SMOTE svm	0.879 ± 0.006	0.963 ± 0.005	0.897 ± 0.008	1033.6 ± 0.49
ADASYN	0.890 ± 0.006	0.976 ± 0.008	0.896 ± 0.003	1057 ± 0
Random Under Sampler	0.836 ± 0.003	0.980 ± 0.003	0.829 ± 0.004	170 ± 0

Table 5: Comparative results of different sampling strategies for the open dataset. ADASYN performs better and is used throughout the paper for the open dataset.

6.4 Contribution model for the open dataset

In the following test, we use the subset of queries that were labeled similarly by the two experts, as explained in Subsection 6.1. This represents a corpus of 753 queries. We used 10-fold validation, with 80% of the corpus for training and 20% for test. The training set was balanced by oversampling, as explained in Section 4. Table 5 presents the different strategies that we compared to balance the 2 classes of our dataset, either by over-sampling the minority class or under-sampling the majority class. It can be seen that all results are very close, but ADASYN provides the best results along with SMOTE regular. However, ADASYN has the advantage of providing a slightly larger dataset (1057 tuples) for our experiments after resampling. We decide to use ADASYN as an oversampling strategy for the open dataset.

The model we obtained is presented in Table 6, which shows, for each query feature, its importance in the definition of contribution to an exploration. The model of contribution was evaluated on the test set; results are reported in Table 7. It achieved an accuracy of 0.88. We note that this is close to the accuracy obtained during the training phase.

A first general observation is that the importance of a feature is fairly related to the features dimensions and/or categories. No dimension or category should be ignored, in the sense that all of them include metrics having relatively high weights. A general trend is that the features taken from relative descriptors have a high impact on the query contribution. Moreover, most of them have a positive impact, which means that the contribution of a query is mostly determined by the consistency of the moves around that query. Features from intrinsic and contextual categories are quite balanced, each one containing almost half of the features with positive weights.

The model fits some intuitions one can have of a contributive query. It can be assumed that analysts pay more attention to contributive queries, and

Intrinsic features		**Relative features**		**Contextual features**	
Feature	Weight	Feature	Weight	Feature	Weight
Bias $w_0 = 0.076$					
NoM	0.294	NCM	-0.167	CpQ	0.043
NoL	0.289	NCL	0.243	ClickDepth	0.247
NoF	0.406	NCF	0.241	IVA	0.032
LDepth	-0.133	RED	-0.025	NoQ	-0.057
FDepth	-0.431	RI	0.568	QRP	-0.099
NoC	-0.117	RR	0.108	QF	0.006
QoI	-0.021	RP	0.184	ElapsedTime	-0.211
ExecTime	-0.235	IsRefine	0.421	ConsTime	0.142
		IsRelax	0.174		

Table 6: Model of query contribution on the open dataset

dataset	**accuracy**	**precision**	**recall**
open	0.880	0.960	0.902

Table 7: Accuracy, precision and recall of the open data model

investigate them more than non contributing ones. A typical behavior is to ask for the same query but with different points of view, by pivoting it in another cross-table, or to explore the direct neighborhood of the query in order to check or explain a result, by using roll-up or drill-down operations. This behavior is represented in the model by the positiveness and high intensity given to features like *ConsTime* (consideration time) that represents the time taken to inspect the query result, or *RI* (relative identity), *IsRefine* (is refinement), *NCL* (number of common levels), *NCF* (number of common levels) and *ClickDepth*, that are relative features indicating the proximity of two successive queries in terms of their expression.

The model also acknowledges that queries with a high contribution are more complex and more selective than other queries. This characteristic is recognized by the model by giving a high positive intensity to features *NoM* (number of measures), *NoF* (number of filters) or *NoL* (number of levels). On the contrary, queries which are too much detailed contain a very high number of cells and are not of high utility to the analyst. Analysts usually do not pay too much attention to these queries, and quickly write another one in order to reduce the size of the query answer. The model identifies this well by weighting negatively features like *NoC* (number of cells), *FDepth* (filter depth), *LDepth* (level depth) and *ExecTime* (execution time).

Surprisingly, while *ConsTime* (consideration time) is relatively important in query contribution, features *QoI* (quantity of information) and *IVA* (increase in view area) have a weak intensity, despite their relatively complex nature. This could mean that even more complex features are needed to better account for

the user's inspection of query results. Interestingly, *ElapsedTime* is weighted negatively, and *NoQ* (number of queries) is weighted weakly. This means that contributive queries are not necessarily located in specific places in the exploration, but tend to be less frequent when the duration of the exploration is high.

6.5 Contribution model for the enterprise dataset

The next experiment aims at investigating whether the contribution model is sensible to the application context. To this end, we repeat the previous experiment for the enterprise dataset (corpus of 525 queries), building a second contribution model.

Many of the differences between the datasets were introduced in Subsection 6.1: different types of users (students vs. analysts); different types of information needs (fuzzy information needs to be extended by students vs. predefined information needs); different characteristics of underlying data (large cubes with tens of dimensions and measures vs. small cubes with specific data); different query tools (Saiku vs. SAP prototype).

Another important difference is that queries in each dataset were labeled by different experts. This is a major issue as the concept of query contribution (as the notion of quality itself) is fuzzy and highly dependent on the evaluator. For example, the experts labeling the open dataset indicated that queries providing the same measures, levels and filters (i.e. identical in our query model) but presenting results in a different way (swapping columns-rows, different charts) were highly contributive, because they showed the analysis of a specific phenomenon around these data. On the other hand, the expert labeling the enterprise dataset preferred queries providing new information and in relation to the user need. Repetitive queries were judged as non contributive.

The model we obtained is presented in Table 8, which shows, for each query feature, its importance in the definition of contribution to an exploration. The model of contribution was evaluated on the test set; results, reported in Table 9, are similar to the ones obtained for the open dataset.

A first general observation is that the importance of the features is substantially different compared to the model obtained on the open dataset. Features that have the highest weights in one model, have negative polarity in the other (ex. *RI* (relative identity) and *IsRefine*), and features that have the lowest weights in one model have positive or insignificant weight in the other (ex. *FDepth* (filter depth), *RI* (relative identity), *RR* (relative recall), *RP* (relative precision), *QRP* (query relative position)). In addition, 15 out of 24 common features have opposite polarity.

The model captures the differences on the underlying datasets. For example, open data cubes being very large, many contributive queries are quite aggregated and have many filters for focusing in a specific portion of a cube. This explains the substantial weight of *NoF* (number of filters) and the quite high weights of *NoL* (number of levels) and *NoM* (number of measures). On the other hand, enterprise data sources being simpler, both in the number of dimensions and levels, and users tend to analyze the entire dataset (less filters),

Intrinsic features		**Relative features**		**Contextual features**	
Feature	Weight	Feature	Weight	Feature	Weight
Bias $w_0 = 0.364$					
NoM	0.034	NCM	0.151	CpQ	-0.015
NoL	0.604	NCL	-0.009	ClickDepth	0.042
NoF	0.001	NCF	-0.070	IVA	-0.086
LDepth	0.569	RED	-0.025	NoQ	0.027
FDepth	-0.109	RI	-0.694	QRP	-0.890
NoC	0.000	RR	-0.621	QF	-0.038
QoI	1.130	RP	-0.399	ElapsedTime	0.133
ExecTime	0.004	IsRefine	0.118		
		IsRelax	-0.141		

Table 8: Model of query contribution on the enterprise dataset

dataset	**accuracy**	**precision**	**recall**
enterprise	0.800	0.817	0.831

Table 9: Accuracy, precision and recall of the enterprise data model

but at specific data granularity (ex. media shops, countries, years) as required in their information needs. In this context, *NoL* (number of levels) and *LDepth* (level depth) had more substantial weights than *NoF* (number of filters) and *FDepth* (filter depth).

The model also captures labeling differences. For example, on the open dataset, the substantial weight of *RI* (relative identity) and in general of almost all relative features, reflect the expert's taste of contributive queries being similar to previous ones. On the contrary, on the enterprise dataset, the very high weight of *QoI* (quantity of information) as well as negative weights for relative features, translate expert's opinion of contributive queries providing new information instead of repeated one.

As a conclusion of this experiment, our learning approach allows the learning of a definition of query contribution (difficult to be verbalized) that captures the characteristics of the underlying dataset and respects experts' judgment.

6.6 Cross-evaluation of contribution models on real datasets

A common objective of machine learning oriented techniques is to be generalizable to any kind of context. The tests reported in previous two sections indicate that, when properly trained on queries related the objective at hand, our method produces good and explainable results. The following test aims at pushing one step further the experiment and at investigating if a model learned for a data set can predict on a completely different data set. It is expected that the results are not as good as previous experiments because of all the differences between datasets and annotation listed in Section 6.5.

Tests	Learning	Testing	Accuracy	Precision	Recall
1	Open	Enterprise	0.408	0.345	0.064
2	Enterprise	Open	0.860	0.862	0.998

Table 10: Results of cross-evaluation between the two real datasets (Open and Enterprise)

Our experimental protocol is as follows. Two tests are conducted: the first one consists of training the model on the open dataset and testing on the enterprise dataset, and the second one consists of performing the other way around. In each test, 100% of the tuples of the learning dataset are used to train the contribution model, this model being tested on 100% of tuples from the second test dataset.

Table 10 presents for each test configuration the classification results observed. Although, the results seem very contrasted, with good results observed in the second test, this experiment shows that our approach is not able, in this context and because of the inherent limits presented before, to learn a single contribution model that can be applied on all our datasets. Indeed, the model learned on the open dataset fails to capture the nature of contributive queries in the enterprise dataset as traduced by a very low recall score. In the second test, the results seem more promising but are biased due to the unbalance of the classes in the open dataset. Indeed, the model learned on the well-balanced enterprise datasets only predicts the majority class. This majority class is the one that we want to predict, hence the good results.

As a concluding remark, this experiment shows that our method may be ready to produce generalizable models as soon as we are able to annotate datasets with some consistency among human experts. However, a second generalization test is described in this paper based on synthetic queries as explained in the next section.

6.7 Validation on artificial explorations

Besides validating the model performance in terms of accuracy and interpretability, we also evaluated the generalization of our model on artificial explorations. In this case, it is not possible to train a new model, neither is it possible to compute traditional accuracy of our model since we have no ground truth about the contribution of each query. However, the choice of CubeLoad as a generative tool for the explorations makes it possible to evaluate qualitatively our model of contribution against its predefined navigation templates. Indeed, these templates directly relate to the level of expertise of the simulated user and thus correlates with the proportion of possible contributive queries.

For example, following the definition of these templates, we could expect *Goal Oriented* explorations to have a high number of contributive queries, as they model users who know what they are looking for. Similarly, *Exploratory* explorations are expected to contain contributive queries because of the diversity in the queries and the increase in information novelty.

Templates	Exploratory	Goal oriented	Slice & Drill	Slice All
# of explorations	23	23	27	27
avg contribution	1.704	0.422	0.650	-0.508
stdev contribution	2.286	1.125	1.158	1.078
min contribution	-1.241	-2.836	-2.085	-3.441
max contribution	8.397	3.756	3.139	3.142
% of contributive queries	87.5	64.9	66.3	24.4

Table 11: Average contribution of queries for each CubeLoad template

Table 11 presents, per exploration template, different statistics on the contribution of the queries forming the explorations. The second line corresponds to the average of the contribution of the requests as predicted by the model learned on the real dataset. The next three lines respectively represent the standard deviation, the minimum and the maximum of contribution a query can reach within the pattern. First, it can be observed that queries from the Slice All template are mostly non contributive queries (25% of them are contributive) with a negative average contribution score of -0.508. This can be explained by the relative simplicity of this pattern which consists solely in making slices on a single level of the cube during the entire exploration. As a consequence, the exploration is purely horizontal and unidirectional. The negative contribution score can therefore be explained by the fact that the queries in these explorations have a very low score for most of the features of the relative category that have positive weights (like NCF, RI, RR, RP), while a high score is achieved for features like NCM or RED that have a negative weight.

Second, it is worth noticing that the Goal Oriented and Slice and Drill templates produce relatively equally contributive explorations on average as denoted by their strong proximity between mean, min and max contribution. That can be explained by the fact that the Slice and Drill pattern can be seen as a special case of the Goal Oriented pattern. Indeed, Goal Oriented simulates a user navigating from a random query to a particular target query. An efficient way to reach this target query is often to use slice and drill sequences. As a consequence of these more complex exploration schemes, the average contribution of each of these templates is much important than the previous Slice All template. The explanation for the better score is the inverse as for the Slice All template: for these templates, a high score is expected for features whose positive weight accounts for an exploratory behavior, like NoM, NoL, NoF, NCL, NCF, RI, RR, RP, IsRefine, IsRelax and ClickDepth.

Finally, the Exploratory template provides by far the most contributive queries with an average of 1.704 and a maximum contribution above 8 which is more than two times the second most contributive template. The high standard deviation value observed is typical of this behavior that goes from very focused queries in some areas of the cube that contribute a lot to answering the problem, to low-focus phases where the user searches for new ways of explorations that

may be very different from what was previously analyzing. This Exploratory behavior favors a high number of distinct filters, measures and levels visited as well as the novelty with pronounced differences between consecutive queries hence increasing many of the features that are highly and positively related to the contribution score. This explains in turn that our model considers that 87.5% of the queries of this template are contributive.

6.8 Model vs. User expertise

In general, an analyst with high level of expertise performs explorations of better quality, with more contributing queries and also queries with higher contribution. The objective of this experiment is to verify that our model is able to confirm this intuition.

To this aim, we conducted this experiment on the open dataset where each exploration has been tagged by professors with A, B or C labels, depending on analyst's skills. Label A corresponds to good explorations, clearly following an information need, investigating it and containing coherent queries. Students producing such explorations are considered to have analysis skills. Contrarily, label C denotes those of the students that produced poor explorations, with less contributive queries, typically switching topics, with no clear information need. Label B corresponds to students that are learning analysis skills, but still produce middle-quality explorations.

A specific protocol has been settled: using the learned model, the contribution score of each query of these explorations has been computed. In order to relate the labels of the explorations to the contribution scores of each query, we have defined an aggregated score for each exploration as the average of its queries' contribution scores. After matching exploration contribution scores and exploration labels, we verify that our model predictions are correct. Results are presented in Table 12.

Analyst skills	A	B	C
Ratio of contributive queries	59.17	53.22	0.25
avg contribution	0.172	0.028	-0.548
stdev contribution	1.042	1.091	0.857

Table 12: Query contribution vs. analyst's skill.

From this table, it is easy to identify that classes A and C are clearly distinguished by their average scores of contribution. Explorations conducted by users who have been labeled as skilled are more contributive than others in average. Regarding classes A and B, we can see that they are correctly ordered when considering the ratio of contributing queries and their average contribution. However, as their scores are close to each other and standard deviation is large, the difference may not be significant. As a consequence, we conducted two complementary experiments in order to figure out how the contributive queries of each class, A and B, are distributed along the explorations.

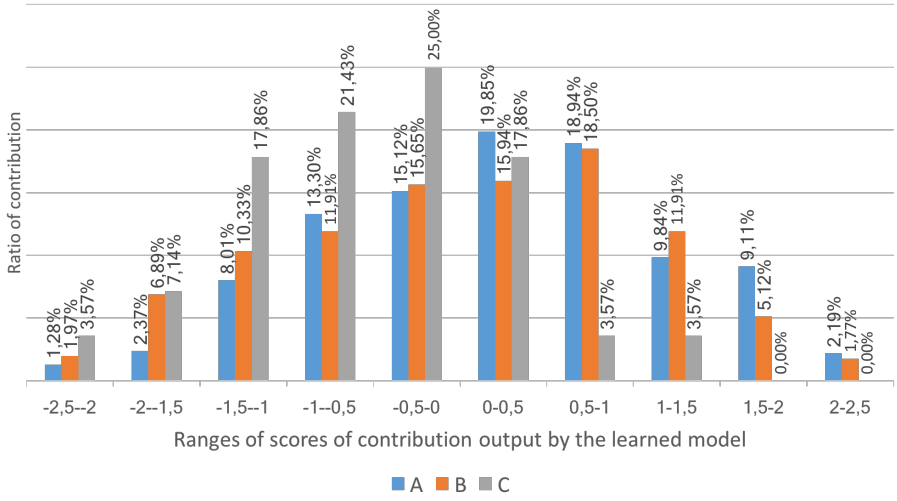


Figure 3: Distribution of ratio of $contrib(q)$ per expertise level

The first complementary experiment aims at verifying that the best analysts produce queries that contribute more. Figure 3 presents, for each level of expertise, the distribution of the ratio of query scores (i.e. $contrib(q)$). For example, in this chart, we can see that 25% of the queries produced by analysts in C have a contribution between -0.5 and 0. More generally, we can see that users in C produce a majority of non contributive queries. The small amount of contributive queries they produce still have a weak contribution compared to analyst in A and B. In general, analysts in A produce less queries of low contribution, and more queries with high contribution, compared to analysts in B. For example, analysts in A produce almost 10% of their queries with a contribution between 1.5 and 2, whereas analysts in B produce only 5% of their queries with such contribution.

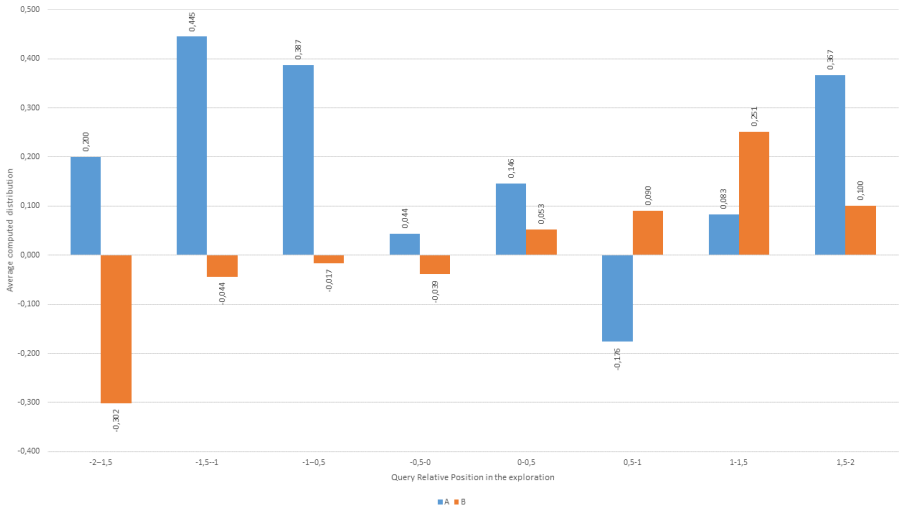


Figure 4: Distribution of ratio of $contrib(q)$ per QRP

The second complementary experiment we conducted aims at observing how the queries contribution distributes along the exploration for analysts of categories A and B. Figure 4 represents the distribution of the average computed contribution of the queries w.r.t. the query relative position (QRP) in the exploration. For obtaining this figure, we split each exploration in 8 parts and compute the average contribution of queries in each part. From this chart, we can read that analysts in B produce queries with an average contribution of 0.302 in the first parts of their explorations. More generally, analysts in B are supposed to have an intermediate level of expertise, and this is corroborated by the fact that the first half of their explorations has a negative contribution in average. However, the average contribution of their queries increases with the query position and starts to be positive approximately at the second half of their explorations. They generally even reach pretty good contribution scores at the end of their explorations. On the contrary, analysts in A master the skill of writing contributing queries from the beginning of their explorations. This is traduced by positive average contribution scores all along their explorations, with some locally low contributions that may correspond to low-focus phases of exploratory navigation behaviors.

6.9 Assessing OLAP explorations

The objective of this last experiment is to validate our primary hypothesis: assuming that expert users are more likely to develop better explorations, we claim that the quality of an exploration can be measured by evaluating the skill of writing contributive queries. To this aim, we relate the scores provided by our continuous Knowledge Tracing (KT hereafter) prediction model to the labels given by the experts on the explorations on the open and enterprise datasets (both sets of explorations were labeled as explained in the previous subsection).

In order to learn the KT model, we started by learning KT parameters, as explained in Section 5. It can be seen from Table 13 that the initial probability of writing a contributive query $P(L_0)$ is very low, which can be directly related to the fact that explorations have been performed by master students who knew very little about the data beforehand. Interestingly, and expectedly, Table 14 shows a higher $P(L_0)$ for enterprise users. However, in both cases users have a sound theoretical background on OLAP exploration which in turn explains the relatively good probability $P(T)$, i.e., the probability to acquire the skill at each step of the exploration. Finally, the exploratory nature of OLAP analysis, combined with limited knowledge of the dataset, translates in the explorations by a lot of trials and errors that increased significantly the average probability and standard deviation of $P(G)$ and $P(S)$.

P(L0)	0.085
P(T)	0.243
mean(P(G))	0.320
variation(P(G))	0.307
mean(P((S))	0.323
variation(P(S))	0.273

Table 13: Main parameters of continuous KT as learned on the open dataset.

P(L0)	0.238
P(T)	0.360
mean(P(G))	0.331
variation(P(G))	0.297
mean(P((S))	0.330
variation(P(S))	0.278

Table 14: Main parameters of continuous KT as learned on the enterprise dataset.

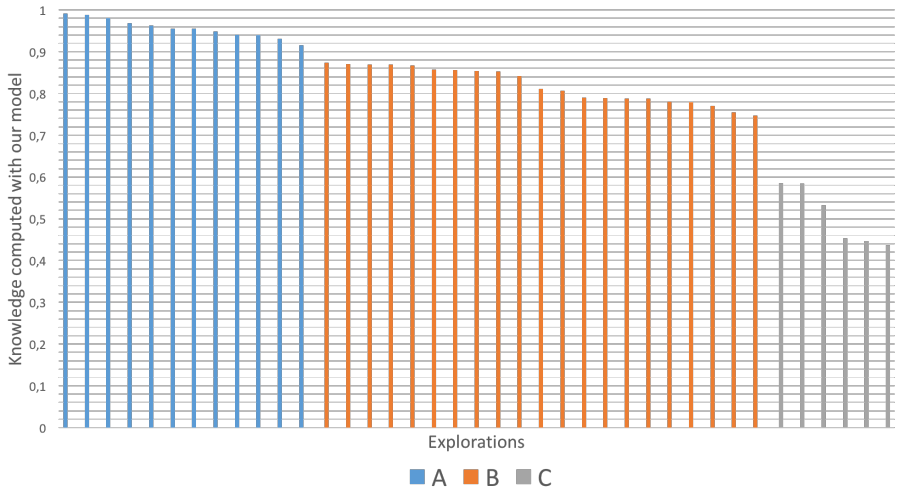


Figure 5: KT prediction of analysts skills vs. experts evaluation for the open dataset

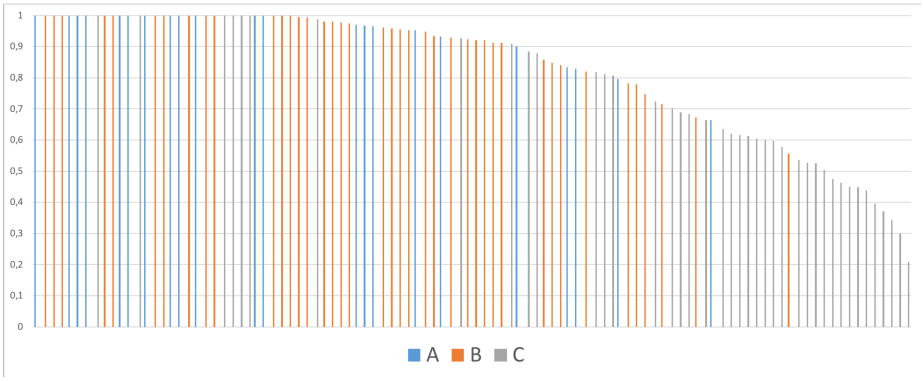


Figure 6: KT prediction of analysts skills vs. experts evaluation for the enterprise dataset

Figures 5 and 6 represent on the horizontal axis all the explorations considered for the experiment (39 for open dataset and 103 for enterprise dataset), and on the vertical axis the prediction of the level of expertise of the user provided by our KT model. The colors represent the assessments made by human experts. For the open dataset, it appears clearly that our model provides consistent evaluations, giving users a rating that corresponds to the assessment made by the expert. The distinction between competent analysts (A and B) and non-competent analysts (C) is clearly marked, in contrast to the distinction between A and B which is less pronounced. This can be explained by the fact that it is difficult for an expert to distinguish between a good analyst and a very good one. However, the distinction between a good and an unqualified analyst

is intuitively much easier. The result is more nuanced on the enterprise dataset, even though we retrieve a quite good distinction between competent and non-competent analysts (C-labeled explorations tend to cluster on the right of the chart, which corresponds to low scores). Interestingly, if only one third of the explorations obtain a score greater than 0.9 for the open dataset, it is slightly more than 50% for the enterprise dataset, which reflects the average expertise of enterprise users. In other words, the KT model was able to correctly retrieve that enterprise users were rather skilled compared to students who explored the Mobpro cube.

Finally, we evaluate our continuous KT based on a traditional RMSE score. RMSE has been shown to be the strongest performance indicator for binary KT with significantly higher correlation than Log Likelihood and Area Under Curve [24]. In our case, this RMSE score is computed as the difference between the expected contribution of each query in an exploration and the value predicted by the continuous KT for each of these queries. Our KT model obtains a RMSE score of 0.291 with a standard deviation of 0.181 for the open dataset, and 0.238 with a standard deviation of 0.270 for the enterprise dataset. Consistently with the literature on KT, we consider that this score are rather good and we conclude that our model is effective at assessing if the skill "writing a contributive query" is acquired.

7 Related Works

7.1 Interactive database exploration

Supporting Interactive Data Exploration (IDE) attracts a lot of attention these days [1]. In their survey, Idreos et al. adopt a top-down viewpoint and classify the existing approaches in three main categories: user interaction, middleware and database layer. Techniques range from visual optimization (like query result reduction [25]), automatic exploration (like query recommendation [26]), assisted query formulation (like data space segmentation [27]), data prefetching (like result diversification [28]) and query approximation [29]. One of the conclusion of this survey is that data navigation systems should be able to optimize exploration tasks, with user profile and navigation histories as corner stones of this optimization. We believe that our work contributes in this direction.

OLAP exploration of data warehouses is a particular use case of database exploration that enables to work with simplifying assumptions. Precisely, the multidimensional star schema or the regularities of multidimensional queries (as presented in Section 2) make possible the definition of simple, intuitive features like level depth, relative edit distance, etc. (see Section 3).

Many approaches have been specifically developed to support OLAP exploration. For example, the PROMISE pre-fetching approach [30] predicts a query based on a Markov Model constructed out of the server's log. Sarawagi [2] proposes an advanced OLAP operator based on entropy maximization to detect surprising cells of a cube, based on the user's current exploration. Collaborative recommendation techniques have been used to recommend OLAP

explorations based on the user's histories and a user's current navigation [31]. Some approaches also give an important place to the presentation of the data, like Cinecubes [32] that builds a user-friendly sequence of explanation queries, to analyze a given query result. We remark that, if these approaches share the same goal of reducing the tediousness of the exploration, they vary a lot in the technique adopted. Noticeably, no commonly agreed upon framework exist to position theses approaches from the user's standpoint.

7.2 Assessing database exploration

The database community enjoys a variety of popular benchmarks to assess and compare the performance of database systems. The TPC consortium[3] proposes benchmarks that include metrics covering time, performance, price, availability or energy consumption. However, while TPC acknowledges the importance of the explorative nature of decision support queries (see e.g., the OLAP interactive queries in the TPC-DS benchmark), none of the existing TPC metrics are appropriate for measuring database exploration support. In other words, the existing benchmarks adopt a system-centric viewpoint, measuring the efficiency of data retrieval, and are not appropriate to measure exploration efficacy under a user-centric angle.

Recently, Eichmann et al. discuss the need for new, user-centric benchmarks [9]. They propose some tracks to investigate the building of such a benchmark. Considering that IDE's main objective is to gain insights about the data, they propose the use of *number of insights per minute* as a primary metric for evaluating systems. They raise the challenges in defining such a metric, like defining user-specific insights and measuring the complexity of an insight. This work is complementary to ours in the sense that insight extraction can be incorporated as one additional feature in our framework.

In a previous work, we propose a framework for benchmarking exploratory OLAP support systems [8]. We showed that such a benchmark can be implemented using state-of-the-art techniques for data and user traces generation, and for metrics definition. We have validated the benchmark by proving that it correctly ranked a set of exploration strategies for which the behavior is well known. The focus of this present work is different in that it treats user explorations, and not systems, as a first class citizen. Additionally, in the present work we use the KT as a systematic way of assessing explorations.

In [7], we propose an approach to detect focus in OLAP explorations. As in the present work, we used supervised learning and feature based description of OLAP queries. This present work can be seen as a generalization of [7], with a more detailed model for query contribution and the use of the KT to score explorations.

[3] See http://www.tpc.org/ for details

7.3 Analyzing web search sessions

Interestingly, other domains, like Information Retrieval and more particularly Web Search, have a longer history of understanding the successfulness of an exploration and processing traces for getting more insight about users.

Analyzing user sessions has been studied for many years in web search. Recent works aim at characterizing the difficulty of search tasks and detecting variations in sessions. For instance, in [33], Athukorala et al. proposed a method for distinguishing between exploratory phases and lookup phases in the context of Information Retrieval. Their idea consists in experimentally discovering features that can be used for this distinction. They submitted several tasks to participants: some of them require exploration while other require more simple lookups. Based on objective measurements, they could identify that query length, completion time, and maximum scroll depth (in the browser), are the most distinctive indicators for distinguishing between exploratory/lookup tasks.

A recent trend in web search is to analyze web search sessions by means of machine learning, and more particularly with classifiers. The goal of authors in [34] is to discover new intent and obtain content relevant to users long-term interests. They develop a classifier to determine whether two search queries address the same information need. This is formalized as an agglomerative clustering problem for which a similarity measure is learned over a set of descriptive features (the stemmed query words, top 10 web results for the queries, the stemmed words in the titles of clicked URL, etc.). In [35], Odijk et al. worked on characterizing user struggling during web searches. They propose a method for distinguishing between users exploring search results from those struggling for satisfying a given information need. The methodology they use is very similar to the one we adopt. They use a bunch of keyword features, and using a large real set of explorations, they trained a machine learning algorithm for learning how to differentiate between the two aforementioned types of explorations.

Measuring the quality of exploration has attracted a lot of attention in the field of Exploratory Search [36]. Exploratory search can be defined as a search paradigm centered on users and the evolution of their knowledge. This paradigm aims at a better support for information understanding by moving beyond the traditional query-browse-refine paradigm. The basic model of exploration distinguishes two main phases. In a first phase, called *exploratory browsing*, users are likely to explore the space, as well as better defining and understanding their problem. At this stage, the problem is being limited, labeled, and a framework for the answer is defined. Over time, the problem becomes more clearly defined, and the user starts to conduct more targeted searches. In this second phase, called *focused phase*, users (re)formulate query statements, examine search results, extract and synthesize relevant information. Being particularly driven by the quality of users experience, it has been proposed that Exploratory Search systems should be evaluated based on the 5 following categories of metrics: i)*Engagement and Enjoyment* measures the "degree to which users are engaged and are experiencing positive emotions". It includes "the amount of interaction

required during exploration", the "extent to which the user is focused on the task". ii) *Task Success* assesses "whether the user reaches a particular target" and finds a "sufficient amount of information and details" along the way. iii) *Information Novelty* measures the "amount of new information encountered". iv) *Task Time* measures the "time spent to reach a state of task completeness". v) *Learning and Cognition* measures the "attainment of learning outcomes", "the amount of the topic space covered" and "the number of insights acquired".

Our work is largely inspired by the Exploratory Search framework. In particular, the features of OLAP queries were defined based on the 5 categories proposed for the evaluation of exploratory search systems.

8 Conclusion

In this work, we have proposed an approach to automatically assess the quality of interactive OLAP explorations of data cubes. Our approach is based on a model of query contribution built using supervised learning and the assessment of the exploration relates to the user's skill of writing queries that contribute to the exploration. The tests conducted on both real and artificial explorations showed the validity of our approach.

Our future works will first investigate the refinement of our model of query contribution, in particular by including complex features like Sarawagi's advanced OLAP operators [2]. Another short term goal is to relax the assumption of multidimensional schema and query language, and target SQL explorations over less normalized databases. We identified SQLShare [37]) as a promising dataset to work with. Finally, a challenging direction will be to switch to unsupervised learning in the approach, to avoid the need for manual labeling and the strong dependency on human expert annotation variability.

References

[1] S. Idreos, O. Papaemmanouil, S. Chaudhuri, Overview of data exploration techniques, in: SIGMOD, 2015, pp. 277–281.

[2] S. Sarawagi, User-adaptive exploration of multidimensional data, in: VLDB, 2000, pp. 307–316.

[3] N. Kamat, P. Jayachandran, K. Tunga, A. Nandi, Distributed and interactive cube exploration, in: ICDE, 2014, pp. 472–483.

[4] K. M. Cauley, Studying knowledge acquisition: Distinctions among procedural, conceptual and logical knowledge, in: 67th Annual Meeting of the American Educational Research Association, 1986.

[5] A. T. Corbett, J. R. Anderson, Knowledge tracing: Modelling the acquisition of procedural knowledge, UMUAI 4 (4) (1995) 253–278.

[6] M. Megasari, P. Wicaksono, C. Y. Li, C. Chaussade, S. Cheng, N. Labroche, P. Marcel, V. Peralta, Can models learned from a dataset reflect acquisition of procedural knowledge? an experiment with automatic measurement of online review quality, in: 19th International Workshop On Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP), 2018.

[7] M. Djedaini, N. Labroche, P. Marcel, V. Peralta, Detecting user focus in OLAP analyses, in: Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings, 2017, pp. 105–119. doi:10.1007/978-3-319-66917-5_8.
URL https://doi.org/10.1007/978-3-319-66917-5_8

[8] M. Djedaini, P. Furtado, N. Labroche, P. Marcel, V. Peralta, Benchmarking exploratory olap, in: TPCTC, 2016.

[9] P. Eichmann, E. Zgraggen, Z. Zhao, C. Binnig, T. Kraska, Towards a benchmark for interactive data exploration., IEEE Data Eng. Bull. 39 (4) (2016) 50–61.

[10] W. J. Hawkins, N. T. Heffernan, R. S. J. de Baker, Learning bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities, in: ITS, 2014, pp. 150–155.

[11] Y. Wang, N. T. Heffernan, Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes, in: AIED, 2013, pp. 181–188.

[12] Y. B. David, A. Segal, Y. K. Gal, Sequencing educational content in classrooms using bayesian knowledge tracing, in: LAK, 2016, pp. 354–363.

[13] P. I. Pavlik, H. Cen, K. R. Koedinger, Performance factors analysis - A new alternative to knowledge tracing, in: AIED, 2009, pp. 531–538.

[14] Y. Gong, J. E. Beck, N. T. Heffernan, Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures, in: ITS, 2010, pp. 35–44.

[15] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, J. Sohl-Dickstein, Deep knowledge tracing, in: NIPS, 2015, pp. 505–513.

[16] R. Kimball, The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses, John Wiley, 1996.

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

[18] G. E. A. P. A. Batista, R. C. Prati, M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explorations 6 (1) (2004) 20–29. doi:10.1145/1007730.1007735.
URL http://doi.acm.org/10.1145/1007730.1007735

[19] K. Drushku, J. Aligon, N. Labroche, P. Marcel, V. Peralta, B. Dumant, User interests clustering in business intelligence interactions, in: Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings, 2017, pp. 144–158. doi:10.1007/978-3-319-59536-8_10.
URL https://doi.org/10.1007/978-3-319-59536-8_10

[20] P. E. O'Neil, E. J. O'Neil, X. Chen, S. Revilak, The star schema benchmark and augmented fact table indexing., in: TPCTC, 2009, pp. 237–252.

[21] T. Rabl, M. Poess, H. Jacobsen, P. E. O'Neil, E. J. O'Neil, Variations of the star schema benchmark to test the effects of data skew on query performance, in: ICPE'13, 2013, pp. 361–372.

[22] S. Rizzi, E. Gallinucci, Cubeload: A parametric generator of realistic OLAP workloads, in: CAiSE 2014, 2014, pp. 610–624.

[23] G. Lemaître, F. Nogueira, C. K. Aridas, Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning, Journal of Machine Learning Research 18 (17) (2017) 1–5.
URL http://jmlr.org/papers/v18/16-365.html

[24] R. Pelánek, Metrics for evaluation of student models, in: EDM, 2015, p. 19.

[25] L. Battle, M. Stonebraker, R. Chang, Dynamic reduction of query result sets for interactive visualizaton, in: Int'l Conference on Big Data, 2013, pp. 1–8.

[26] M. Drosou, E. Pitoura, Ymaldb: exploring relational databases via result-driven recommendations, VLDB J. 22 (6) (2013) 849–874.

[27] T. Sellam, M. L. Kersten, Meet charles, big data query advisor, in: CIDR, 2013.

[28] H. A. Khan, M. A. Sharaf, A. Albarrak, Divide: efficient diversification for interactive data exploration, in: SSDBM, 2014, pp. 15:1–15:12.

[29] J. M. Hellerstein, P. J. Haas, H. J. Wang, Online aggregation, in: SIGMOD, 1997, pp. 171–182.

[30] C. Sapia, Promise: Predicting query behavior to enable predictive caching strategies for olap systems, in: DaWaK, 2000, pp. 224–233.

[31] J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, S. Rizzi, A collaborative filtering approach for recommending OLAP sessions, DSS 69 (2015) 20–30.

[32] D. Gkesoulis, P. Vassiliadis, P. Manousis, Cinecubes: Aiding data workers gain insights from OLAP queries, IS 53 (2015) 60–86.

[33] K. Athukorala, D. Glowacka, G. Jacucci, A. Oulasvirta, J. Vreeken, Is exploratory search different? A comparison of information search behavior for exploratory and lookup tasks, JASIST 67 (11) (2016) 2635–2651.

[34] R. Guha, V. Gupta, V. Raghunathan, R. Srikant, User modeling for a personal assistant, in: WSDM, 2015, pp. 275–284.

[35] D. Odijk, R. W. White, A. H. Awadallah, S. T. Dumais, Struggling and success in web search, in: CIKM, 2015, pp. 1551–1560.

[36] R. W. White, R. A. Roth, Exploratory Search: Beyond the Query-Response Paradigm, Morgan & Claypool Publishers, 2009.

[37] S. Jain, D. Moritz, D. Halperin, B. Howe, E. Lazowska, Sqlshare: Results from a multi-year sql-as-a-service experiment, in: SIGMOD, 2016, pp. 281–293.