Insight-based vocalization of OLAP sessions

Matteo Francia[0000−0002−0805−1051], Enrico Gallinucci[0000−0002−0931−4255], Matteo Golfarelli[0000−0002−0437−0725], and Stefano Rizzi[0000−0002−4617−217X]

DISI - University of Bologna, Bologna, Italy
{m.francia,enrico.gallinucci,matteo.golfarelli,stefano.rizzi}@unibo.it

Abstract. Carrying out OLAP analyses in hands-free scenarios requires lean forms of communication between the users and the system, based for instance on natural language. In this paper we introduce VOOL, a framework specifically devised for vocalizing the insights resulting from OLAP sessions. VOOL is self-configurable, extensible, and is aware of the user's intentions expressed by OLAP operators. To avoid overwhelming the user with very long descriptions, we pursue the vocalization of selected insights automatically extracted from query results. These insights are detected by a set of modules, each returning a set of independent insights that characterize data. After describing and formalizing our approach, we evaluate it in terms of efficiency and effectiveness.

Keywords: Vocalization · OLAP · Data warehouse

1 Introduction

The democratization of data access pushes towards the adoption of OLAP (On-Line Analytical Processing) tools, which make data fruition and analysis easier by enabling "point-and-click" queries on the multidimensional cubes stored in a data warehouse. The scenarios requiring hand-free interfaces (e.g., in the field of augmented reality [9] or smart assistants [7]) make this push even more pressing and ask for the introduction of leaner forms of communication between the users and the system, based for instance on natural language. As argued in [28], this setting is not only motivated by the needs of specific user groups, such as visually-impaired ones. More in general, we are assisting to a shift of user-computer communication towards voice interfaces, which are more convenient if users are distracted or unable to access screen and keyboard. While the translation of natural language into actionable OLAP queries has recently been addressed [7], the way to the vocalization of query results has been paved only partially. The goal of this paper is to contribute to bridging this gap.

The description of the many facets shown by the cube resulting from an OLAP query can span from simple insights (e.g., min/max or Top-k) to complex ones (e.g., clusters and outliers); these, in turn, can be representative of very different amounts of facts in the cube. Additionally, according to the OLAP paradigm, data analyses come in the form of sessions, where a query q' can be obtained by applying an OLAP operator to the previous one, q. Hence, differently

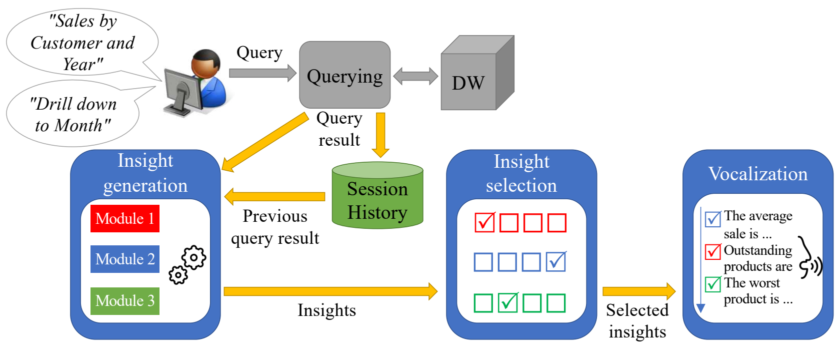

Fig. 1. Functional view of VOOL

from a generic sequence of stand-alone queries, q and q' are strongly related, which enables the detection of insights based on the comparison of the results of consecutive queries. These insights should also be related to the intention expressed by the user when applying an OLAP operator; for instance, drill-down refines the previous result while slice-and-dice shifts the focus to a specific part of the result. Overall, in our vision, the desiderata for a framework for the vocalization of OLAP sessions are the following:

#1 *Intention-awareness*: it must generate vocalizations that describe the comparison of the results of subsequent queries rather than those of a single one; in generating such vocalizations, it must consider the user's intention as expressed by the OLAP operator and aggregation operator applied.
#2 *Extensibility*: it must rely on interfaces that make ad-hoc modules easily pluggable since vocalization is inherently multi-faceted.
#3 *Timeliness*: it must produce vocalizations responsively, avoiding long delays in returning results to the user.
#4 *Conciseness*: it must produce vocalizations that take a limited time not to overwhelm the user.

Following these desiderata, in this paper we present *VOOL*, a framework specifically devised for the VOcalization of OLap sessions. A functional view of VOOL is sketched in Figure 1 (the querying component is grayed out since it is out of the scope of this paper and has been extensively discussed in [7]). Given the result of either a completely-specified query (e.g., "Sales by Customer and Year") or an OLAP operator that refined the previous query (e.g., "Drill down to Month"), the insight generator executes concurrent modules, each returning a set of independent insights that characterize this result. Out of all the insights returned, the insight selector applies an optimization algorithm to return only the most relevant insights given a limited budget (e.g., related to the duration of vocalization); these insights are then sorted into a comprehensive description that is vocalized to the user.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Section 3 provides an overview of VOOL by sketching its functional architecture and the vocalization steps. Section 4 formalizes the necessary background. Section 5 describes the vocalization process, while Section 6 details one of the modules we implemented to support the VOOL framework. Finally, Section 7 evaluates the approach by means of a set of tests, draws the conclusions, and envisions the directions for evolving VOOL.

2 Related work

The first research area that intersects with our contribution is *exploratory data analysis*, a knowledge discovery process in which users explore datasets through sessions that concatenate a sequence of operations. In this context, two interesting research directions are *recommendation* and *insight extraction*. As to recommendation, many studies focus on learning users' preferences and profiling data to give recommendations on the exploration path [27,25]. This is orthogonal to our approach since our goal is not to suggest to the user how to build a session, but rather to return concise insights on the data resulting from a user-defined session. As to insight extraction, OLAP comes with well-known operators to explore multidimensional cubes [22]. Additional operators have been recently classified as [13] *coverage* (returning insights that cover tuples with certain values [12,4]), *information* (returning insights providing information about the distribution of measure values [10]), and *contrast* (returning insights occurring with some values but not the others [24,8]). These operators are complementary to VOOL, since they are potential modules to be plugged into VOOL (as we have already done for [4,10,29,8]). Cinecubes [11], the contribution closest to VOOL, compares the results of a query to results obtained over sibling values or drill-downs to produce insight. With respect to Cinecubes, VOOL allows the description and vocalization of a user-defined OLAP session and also leverages the user's intentions to drive insight extraction.

Another research area closely related to our work is that of *conversational systems*. Natural language interfaces to operational databases enable users to specify complex queries without previous training on formal programming languages (such as SQL) and software [1]. Some examples of approaches that translate natural language into formal SQL/OLAP queries are [17,23,7]. In hand-free scenarios, some emphasis has been given on the one hand to providing effective summarizations of query results, which enables the creation of concise analytic reports [9,6]; on the other hand, some vocalization approaches have been proposed. In [26], the authors translate a database subset into a narrative that synthesizes the contents of the subset following a set of rules and templates. In [5], the authors leverage the provenance of tuples in the query result, detailing not only the results but also their explanations. Finally, a couple of works are placed in the context of multidimensional data and OLAP. In [28], the authors sample the database to evaluate alternative speech fragments; OLAP queries are not fully evaluated and sampling focuses on result aspects that are relevant for voice output. In [21] an end-to-end dialog system is introduced, but the vocalization approach is limited (when too many rows should be returned, only the count of rows is returned).

Overall, in the light of the above-mentioned contributions, it appears that the road to full-fledged conversation-driven OLAP is not paved yet, since end-to-end conversational frameworks are not provided in the domain of analytic sessions over multidimensional data. The closest contribution to VOOL is [28]; however, differently from VOOL, that approach only copes with stand-alone queries, so the vocalization does not take into account the comparison of the sequential query results emerging from OLAP sessions; besides, it is not extensible and does not provide a dynamic interest-based vocalization of the insights.

3 Overview

The interaction with VOOL takes place as follows. (i) A user issues an *initial* query (typically, the first one in a session), whose result is computed; (ii) a set of vocalization insights (i.e., descriptions of insights) are extracted out of the query result; (iii) the most interesting insights are vocalized. Every time the user issues a new query by applying an OLAP operator (obtaining a *refined query*), this process is repeated; the difference is that the insights extracted may describe not only the result of the last query, but also its comparison with the results of the previous query.

Vocalization of initial queries. The result returned by the *Querying* component is sent to the insight generation step, in which a set of modules analyzes the query result to produce different types of insights. Each insight is characterized by a natural language description, an interestingness, a coverage (i.e., the number of tuples covered by the description), and the cost necessary for its vocalization (e.g., the number of words of its natural language description). An example of natural language description for an insight produced by a Top-k module is "The facts with highest Quantity are Beer with 80, Wine with 70, and Cola with 30". Since the number of insights can be arbitrarily high (a module can return any number of insights and there is no limit to the number of modules), the insight selection step determines the insights eligible for vocalization in such a way that the total time necessary for vocalization does not overcome a given time budget and the total interestingness is as high as possible. Finally, the selected insights are vocalized from the most general (i.e., those with high coverage) to the most specific (those with low coverage).

Vocalization of refined queries. The results of the current query and of the previous one are sent to the insight generation step. In this case, both cubes are used to extract the insights entailing the comparison and description of consecutive results. For instance, in the sales domain, after drilling down sales from product category to product, a user might be interested in outstanding products that were previously hidden within average-performing categories. After insight generation, vocalization proceeds as for an initial query.

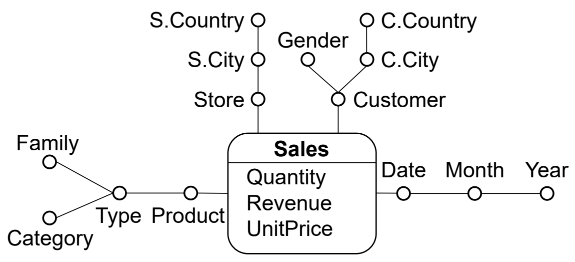

Fig. 2. (Simplified) DFM representation of the Sales cube schema

4 Formal background

A *cube* is the multidimensional representation of a business phenomenon relevant for decision making, and is defined through the following steps.

Definition 1 (Hierarchy and Cube Schema). *A* hierarchy *is a couple* $h = (L, \succeq)$ *where (i)* L *is a set of categorical* levels*, each level* l *being coupled with a* domain *of* members*,* $Dom(l)$*; (ii)* $\succeq$ *is a* roll-up *partial order of* L*. A* cube schema *is a couple* $\mathcal{C} = (H, M)$ *where (i)* H *is a set of hierarchies; (ii)* M *is a set of numerical measures, each coupled with an aggregation operator* $op(m) \in \{\texttt{sum}, \texttt{avg}, \texttt{min}, \texttt{max}\}$.

Example 1. As a working example we will use cube schema Sales $= (H, M)$, whose conceptual representation according to the DFM [14] is shown in Figure 2:

$$H = \{h_{\mathsf{Date}}, h_{\mathsf{Customer}}, h_{\mathsf{Store}}, h_{\mathsf{Product}}\}$$
$$M = \{\mathsf{Quantity}, \mathsf{Revenue}, \mathsf{UnitPrice}\}$$
$$\mathsf{Date} \succeq \mathsf{Month} \succeq \mathsf{Year}, \mathsf{Store} \succeq \mathsf{S.City} \succeq \mathsf{S.Country}, \ldots$$

We have $op(\mathsf{Quantity}) = op(\mathsf{Revenue}) = \texttt{sum}$ and $op(\mathsf{UnitPrice}) = \texttt{min}$. □

Aggregation is the basic mechanism to query cubes, and it is captured by the following definition of group-by. As normally done when working with the multidimensional model, if a hierarchy h does not appear in a group-by it is implicitly assumed that a complete aggregation is done along h.

Definition 2 (Group-by and Coordinate). *Given cube schema* $\mathcal{C} = (H, M)$*, a* group-by *of* $\mathcal{C}$ *is a tuple* G *of levels. A* coordinate *of group-by* G *is a tuple of members, one for each level of* G.

Definition 3 (Base Cube). *Let* G_0 *be the finest group-by. A* base cube *over* $\mathcal{C}$ *is a partial function* C_0 *that maps the coordinates of* G_0 *to a numerical value for each measure* m *in* M.

Each coordinate γ that participates in C_0, with its associated tuple of measure values, is called a *fact* of C_0. The value taken by measure m in the fact corresponding to γ is denoted $\gamma.m$. With a slight abuse of notation, we will also consider a cube as the set of the coordinates corresponding to its facts.

Example 2. Three group-by's of Sales are $G_0 = \langle \mathsf{Date}, \mathsf{Customer}, \mathsf{Store}, \mathsf{Product} \rangle$, $G_1 = \langle \mathsf{Month}, \mathsf{C.City}, \mathsf{Gender} \rangle$, and $G_2 = \langle \mathsf{Year} \rangle$, where $G_0 \succeq_H G_1 \succeq_H G_2$. Coordinates of the three group-by's are, respectively, $\gamma_0 = \langle$2021-04-15, Rossi, BigMart, Beer$\rangle$, $\gamma_1 = \langle$2021-04, Rome, Male$\rangle$, and $\gamma_2 = \langle 2021 \rangle$. □

Definition 4 ((Cube) query). *Given a base cube C_0 over schema $\mathcal{C}$, a* query *over C_0 is a quadruple $q = (C_0, G_q, P_q, M_q)$ where (i) G_q is a group-by of $\mathcal{C}$; (ii) P_q is a (possibly empty) set of selection predicates each expressed over one level of H; (iii) $M_q \subseteq M$.*

Example 3. A query over Sales is the one returning the total quantity sold by product, which can be formalized as $q = (C_0, G_q, P_q, M_q)$ where $G_q = \{\mathsf{Product}\}$, $P_q = \varnothing$ (i.e., no selection predicate is applied), and $M_q = \{\mathsf{Quantity}\}$. □

An *OLAP session* is a sequence of queries; the first query in a session is completely specified, while each of the others is obtained as a refinement by applying an *OLAP operator* to the result of the previous one. A formal definition of the OLAP operators we will consider in this work can be found in [7]; here we just give an intuition:

– *Roll-up* aggregates data (e.g., from $\mathsf{Product}$ to Type).
– *Drill-down* disaggregates data (e.g., from Type to $\mathsf{Product}$).
– *Slice-and-dice* filters data based on a predicate (e.g., $\mathsf{Product}$ = 'Beer').

5 The vocalization process

As already stated, the VOOL framework includes three main stages, namely, *Insight generation*, *Insight selection*, and *Vocalization*; all of these are described in the following subsections.

5.1 Insight generation

At this stage, a set of *modules* (e.g., the Top-k function or a clustering function) are executed to extract *insights* (e.g., the top-3 facts or a pair of clusters) describing query results. An insight consists of a set of *components*, each describing either a single fact (e.g., a fact belonging to the top-3 facts) or a group of facts (e.g., a cluster).

Definition 5 (Module). *Given two cubes C and C' resulting from two consecutive queries in an OLAP session (with $C = \varnothing$ when vocalizing an initial query), a* module *is a function $F(C, C') = S^F$, where S^F is a set of insights.*

The executability of a module is subject to the fulfillment of specific conditions, which may concern the applied OLAP operators, the measures involved in the query result, and the aggregation operator used in the query. Note that, consistently with desideratum #2 (Extensibility), this definition allows the application of any function capable to extract insights from one or two cubes.

$C' = q(\mathsf{Sales}, \{\mathsf{Product}\}, \emptyset, \{\mathsf{Quantity}\})$

Product	Quantity
Beer	80
Wine	70
Cola	30
Bagel	8
Pizza	6
Bread	5

Fig. 3. The cube C' resulting from the (initial) query in Example 3, which represents the Quantity sold by Product

Definition 6 (Insight). *An* insight $s \in F(C, C')$ *is a set of components; each component $v \in s$ describes a set of facts of C', denoted as $Desc(v)$. Insight s is characterized as follows:*

(i) *$NL(s)$ is the natural language description of s.*
(ii) *$int(s)$ is the interestingness of the insight, i.e., its estimated relevance to the decision-making process, defined as*

$$int(s) = \sum_{v \in s} int(v)$$

where $int(v) \in (0, 1]$ is the interestingness of component v.
(iii) *$cov(s) \in (0, 1]$ is the fraction of cube facts covered by the insight, called coverage:*

$$cov(s) = \frac{|\bigcup_{v \in s} v|}{|C'|}$$

(iv) *$cost(s) \in \mathbb{N}$ is the cost related to the vocalization of s, measured as the number of words in $NL(s)$.*

The natural language descriptions of insights, $NL(s)$, are generated from predefined module-specific grammars. The interestingness of insight components, $int(v)$, is also specific of each module; an example of how $int(v)$ is defined for one module will be provided in Section 6. Intuitively, an insight with high coverage is more general, one with small coverage is more specific.

Definition 7 (Insight space). *Let $\mathcal{F}$ be the set of all modules. Given two consecutive cubes C and C' in an OLAP session (possibly with $C = \varnothing$), their* insight space *is the set of the sets of insights produced by all modules:*

$$\mathcal{S} = \{F(C, C'); F \in \mathcal{F}\} = \{\{s_1^F, \ldots, s_n^F\}; F \in \mathcal{F}\}$$

To enable concurrent and efficient insight generation and selection (as shown later), we make two assumptions on insights and modules:

1. Each insight s is *self-contained*, i.e., $NL(s)$ contains all the information necessary for vocalization and is a self-standing sentence. As a consequence, insights can be vocalized independently of each other.

Table 1. Examples of insights describing the query result in Figure 3

Module	Insight	*NL*	*int*	*cov*	*cost*
Statistics	s^{P}	The average Quantity is 33.2	0.0	1.0	5
Top-k	s_1^{T}	The fact with highest Quantity is Beer with 80	0.4	0.2	9
	s_3^{T}	The three facts with highest Quantity are Beer with 80, Wine with 70, and Cola with 30	1.0	0.5	17
Clustering	s_1^{C}	Facts can be grouped into 2 clusters, the largest one has 4 facts and 12 as average Quantity	0.8	0.7	18
	s_2^{C}	Facts can be grouped into 2 clusters, the largest one has 4 facts and 12 as average Quantity, the second one has 2 facts and 75 as average Quantity	1.6	1.0	29
Assess	s_1^{A}	When compared to the previous query, the Quantity of Pizza is 6, tantamount to the average Quantity of Food that is 6.3	1.0	0.2	22

2. The insights generated from the same module F are *incremental*, i.e., they can be arranged into a sequence where the description of one insight extends the previous one by including one more component. In the following we will assume that the resulting inclusion (total) ordering is reflected in the ordering of indices: $S^F = \{s_1^F, \ldots, s_n^F\}$, with $cov(s_{i+1}^F) \geq cov(s_i^F)$, $int(s_{i+1}^F) \geq int(s_i^F)$, and $cost(s_{i+1}^F) \geq cost(s_i^F)$ for $1 \leq i \leq n-1$.

Example 4. Given the query result from Figure 3, examples of insights produced by different modules are shown in Table 1. Note that, from the informative point of view, an insight may be an extension of another insight because it includes additional components (e.g., s_3^{T} extends s_1^{T} with two additional components corresponding to two facts, namely, Wine and Cola, while s_2^{C} extends s_1^{C} with an additional component corresponding to a cluster including two facts). □

5.2 Insight selection

The insight space $\mathcal{S}$ can be very large, so a selection must be done on the insights to be vocalized. The goal of this step is to find the set of insights $\overline{S} \subseteq \mathcal{S}$ such that (i) the total interestingness is maximum and (ii) the total cost does not exceed a given time budget t_{voc} (see desideratum #4, Conciseness). Expressing t_{voc} in seconds makes budget definition intuitive for users. However, the insight cost has been defined as the number of words in its textual description, so as to decouple it from its vocalization (the optimal speech rate may depend on the target audience). Transforming t_{voc} into a maximum number of words is straightforward; for instance, 180 is the average number of words per minute for English speakers/readers [3].

The one formulated above is clearly an optimization problem, with two additional issues to be considered: non-redundancy and right-time response.

Non-redundancy. While, by assumption, different modules produce insights with different semantics, the insights from the same module have overlapping content (since they are built incrementally). As a consequence, given a module F and its output S^F, at most one insight $s_z^F \in S^F$ should be selected. Insight selection can then be formulated as a multiple-choice knapsack problem (MCKP), a generalization of the ordinary knapsack problem. In the MCKP, the set of items ($\mathcal{S}$) is partitioned into classes (the S^F's) and the binary choice of taking or not an item is replaced by the selection of at most one item out of each class [16].

Right-time response. $\mathcal{S}$ is incrementally populated since the modules entail different complexities and execution times and are executed in a *bag-of-task* fashion (see desideratum #3, Timeliness). On the other hand, to preserve the interactivity of OLAP session, vocalization should begin shortly after query execution, without waiting until all the modules have completed their execution. Thus, insight selection is started after a fixed time t_{gen}, and the insights added to $\mathcal{S}$ after t_{gen} are not included in the selection process.

5.3 Vocalization

Vocalization starts with a preamble that describes the query (e.g., "The query result shows the sum of quantity grouped by product"). The preamble is a preliminary description which acts as a context for the subsequent insights. Note that, if the time necessary to vocalize the preamble is greater than t_{gen}, the user will not perceive any pause in the vocalization. After the preamble, the insights in $\overline{S}$ are vocalized. Specifically, they are sorted by descending coverage cov (i.e., from the most general to most specific), then their natural language descriptions NL's are concatenated and vocalized.

6 A closer look at the VOOL modules

The core set of modules currently implemented is summarized below:

- *Statistics* returns general statistics on the overall result (e.g., the average value of the Quantity measure and its skewness).
- *Bottom-k/Top-k* [20], applied to a single measure, returns the worst/best performing facts (e.g., sales with lowest/highest Quantity).
- *Outliers* [19] returns the facts whose measure values deviate from the data distribution (e.g., anomalous sales).
- *Clustering* [18] returns groups of facts that maximize intra-group similarity and minimize inter-group similarity (e.g., facts with similar Quantity).
- *Correlation* returns the degree of Pearson correlation between pairs of measures (e.g., how Quantity and Revenue correlate).
- *Slicing variance*, applied to a single measure, returns the degree of correlation between the values of a measure in the cubes before and after the application of a slice-and-dice operator (e.g., how quantities by product change after applying selection predicate StoreCity='Rome').
- *Aggregation variance*, applied to a single measure, returns the facts with the highest variation in the values of that measure after a roll-up or drill-down operator (similarly to [4]; e.g., after a roll-up from Product to Category, returns the categories showing the highest variation in products' Quantity).

Note that some of these modules are inspired from well-known approaches (e.g., [4,10,29,8]); in some of these cases we just had to devise a textual description of the insight and/or adapt the returned measures of interestingness/relevance. As already mentioned, this set can smoothly be extended with modules that follow the requirements expressed in Section 1.

In the remainder of this section we describe an end-to-end implementation of the Top-k module, which operates on both initial and refined queries. For simplicity, we will drop the superscript denoting the module from the notation of insights. Let $q = (C_0, G_q, P_q, M_q)$ be an initial query, with $M_q = \{m\}$, and C' the resulting cube. The goal of the Top-k module is to describe the three facts in C' having the highest values of m, namely, $\{\gamma_1, \gamma_2, \gamma_3\}$ (we will assume that $\gamma_1.m \geq \gamma_2.m \geq \gamma_3.m \geq \ldots$). Three insights including from one to three components are returned:

$$s_1 = \{\{\gamma_1\}\},\ s_2 = \{\{\gamma_1\}, \{\gamma_2\}\},\ s_3 = \{\{\gamma_1\}, \{\gamma_2\}, \{\gamma_3\}\}$$

These insights are characterized as follows:

$$NL(s_k) = \begin{cases} \text{“The fact with highest } m \text{ is } \gamma_1 \text{ with } \gamma_1.m\text{”}, & \text{if } k = 1; \\ \text{“The two facts with highest } m \text{ are } \gamma_1 \text{ with } \gamma_1.m \text{ and} \\ \gamma_2 \text{ with } \gamma_2.m\text{”}, & \text{if } k = 2; \\ \text{“The three facts with highest } m \text{ are } \gamma_1 \text{ with } \gamma_1.m, \\ \gamma_2 \text{ with } \gamma_2.m,\ \gamma_3 \text{ with } \gamma_3.m\text{”}, & \text{if } k = 3; \end{cases}$$

$$cov(s_k) = \frac{k}{|C'|}$$

As to the interestingness, for each component $v_k = \{\gamma_k\}$ it is

$$int(v_k) = \frac{\gamma_k.m - \gamma_{\overline{k}}.m}{\sum_{i=1}^{\overline{k}} (\gamma_i.m - \gamma_{\overline{k}}.m)}$$

where $\overline{k} > 3$. While the coverage formula is obvious, the interestingness of s_k corresponds to the percentage of m that is retained by the Top-k tuples (e.g., the total Quantity retained by the Top-3 products with respect to the overall units sold by the Top-$\overline{k}$ facts). The reason for limiting the denominator to the Top-$\overline{k}$ facts rather than summing on all the query results is to avoid that a *long tail* of several low values makes $int()$ meaningless. Conversely, by considering only the highest non-top values (in our implementation we set $\overline{k} = 6$) the interestingness function properly expresses how high are the Top-3 as compared to the next ones. Finally, the reason why all measure values are shifted by $\gamma_{\overline{k}}.m$ is to cope with the case of negative values (e.g., if the measure expresses a temperature).

As to refined queries, while NL and cov remain unchanged, the interestingness of a component changes depending on the result of the previous query. Given two consecutive cubes C and C', a fact in C' is considered to be interesting (in the sense of *peculiar*) if its measures deviate significantly from those in the

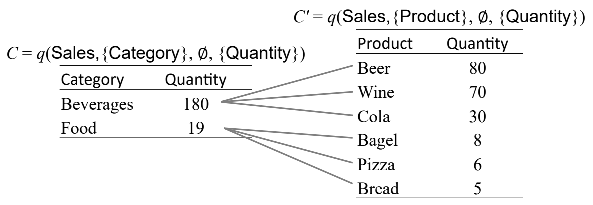

Fig. 4. Example of corresponding facts (gray lines) between the results of two consecutive queries, C and C', the latter being obtained by drilling down the former from Category to Product

corresponding fact(s) of C [8]. This is based on the idea of prior belief [2]: specifically, the interestingness is defined as the difference of belief for corresponding facts in the cubes before and after the application of an OLAP operator. For instance, after drilling down from Category to Product, the more the Quantity of Beer deviates from the Quantity of Beverages, the higher its peculiarity; in other words, a user is less likely to expect a product with outstanding sales coming from a category with middling sales. Measuring interestingness in this way requires to define, for each fact in C', the "corresponding fact(s)" in C. To this end we use, as in [8], a *proxy function* $proxy_C(\gamma)$ (with $\gamma \in C'$) that models a one-to-many (many-to-one) mapping in case of drill-down (roll-up), and a one-to-one mapping in case of slice-and-dice or addition/removal of a measure (see Figure 4 for an example). Intuitively, if the OLAP operator changes the group-by, the corresponding fact(s) of C are determined via the roll-up order; if the operator changes the selection predicate, the corresponding facts of C are one-to-one mapped to the facts of C'; if the operator changes the measure, the corresponding facts are the empty set. For the formal definition of proxy and peculiarity $pec()$, we refer the reader to [8]. Finally, the interestingness of component $v_k = \{\gamma_k\}$ describing the results of a refined query is defined as for initial queries, but weighing measure values on fact peculiarity:

$$int(v_k) = \frac{(\gamma_k.m - \gamma_{\overline{k}}.m) \cdot pec(\gamma_k)}{\sum_{i=1}^{\overline{k}} (\gamma_i.m - \gamma_{\overline{k}}.m) \cdot pec(\gamma_i)}$$

Example 5. As already shown in Table 1, if C' is the cube in Figure 3 resulting from an initial query, examples of insights are

$s_1^{\mathrm{T}} = (NL =$ "The fact with higher Quantity is Beer with 80",
$int = 0.44, cov = 0.20, cost = 9)$

$s_3^{\mathrm{T}} = (NL =$ "The three facts with higher Quantity are Beer with 80, Wine with 70, and Cola with 30",
$int = 0.98, cov = 0.50, cost = 17)$

On the other hand, if C' is the result of a drill-down from Category to Product as in Figure 4, the interestingness changes as follows:

$$s_1^{\mathrm{T}}.int = \frac{(80-5)\cdot 0.21}{64.33} = 0.24$$

$$s_3^{\mathrm{T}}.int = \frac{(80-5)\cdot 0.21 + (70-5)\cdot 0.36 + (30-5)\cdot 1.0}{64.33} = 0.98$$

Intuitively, following the prior belief principle, since Beer is the top product of the top-selling category, Beer is less interesting than Cola (which is the worst-selling beverage). □

7 Evaluation and conclusion

In this paper we have presented VOOL, an approach for vocalizing selected insights out of the results of an OLAP session. To test the approach we have implemented a prototype using Java and Python; the necessary mining models are imported from the Scikit-Learn library and the insights are vocalized through the text-to-speech Google APIs.

To evaluate the efficiency of VOOL, we made some experiments against the Foodmart cube [15] to understand how the performance of each module scales with respect to the cardinality of the query result. To this end we executed 10 OLAP sessions, each involving 3 OLAP steps; different combinations of modules were tested, and all the modules were invoked in at least one session. The tests were run on an Intel(R) Core(TM)i7-6700 CPU@3.40GHz with 8GB RAM. The tests were repeated 10 times and the average results are reported. Figure 5 shows the scalability of each module against query results with increasing cardinalities (up to 10^4). We emphasize that, since our work focuses on the vocalization of OLAP sessions —and not on the generic mining of multidimensional cubes—, 10^4 is large enough to be considered unrealistic for OLAP, since the results analyzed by users are usually constrained by the visualization and interaction metaphors adopted [9]. Noticeably, for query results including 10^4 facts, the computation of all the modules requires less than 1 second. The only exception is Clustering, which requires 7 seconds on average for query results with cardinality 10^4.

To assess the effectiveness of VOOL, we made some preliminary tests with 10 users, mainly master students in data science, with basic or advanced knowledge of business intelligence and data warehousing. The users were briefly introduced to the vocalization problem and to VOOL, then they were assigned three OLAP sessions with different analysis goals (e.g., "As a shop owner, you are analyzing the sum of quantity sold in each product department") and the query results were vocalized. On a scale from 1 (very poor) to 5 (very high), the average results show that both the user experience and the description of query results are deemed to be good (4.2 ± 0.6 and 3.8 ± 0.9, respectively). The insights with highest/lowest appreciation are Aggregation variance and Statistics, respectively (the latter sometimes is too simple to describe the whole result); the users asked to refine some of the proposed modules and suggested extensions with new ones.

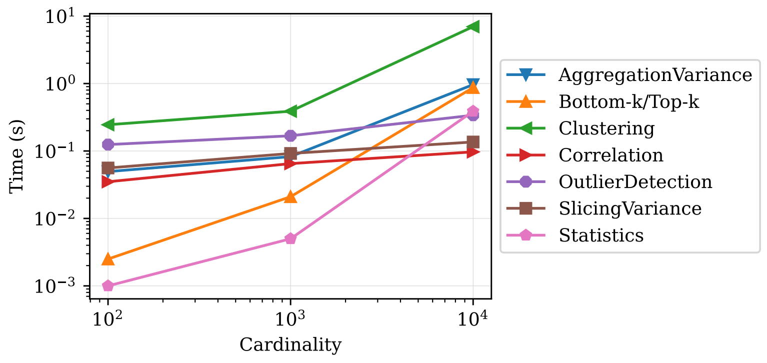

Fig. 5. Performance scalability of the modules

Overall, these preliminary results confirm the effectiveness and efficiency of VOOL. Besides refining and extending the modules, other directions that can be envisioned for future research are: (i) handling the redundancy of insights over single queries (since multiple modules can vocalize the same tuples, the interestingness of an insight should also depend on those previously selected) and sessions (vocalizing the same insight twice or more reduces its interestingness); (ii) introducing a "tell me more" interaction, where users can ask for further details as well as insights retrieved after the time budget; and (iii) conducting additional tests to assess the correlation between the insights vocalized and the users' intentions.

References

1. Affolter, K., Stockinger, K., Bernstein, A.: A comparative survey of recent natural language interfaces for databases. VLDB J. **28**(5), 793–819 (2019)
2. Bie, T.D.: Subjective interestingness in exploratory data mining. In: Proc. IDA. pp. 19–31 (2013)
3. Brysbaert, M.: How many words do we read per minute? A review and meta-analysis of reading rate. Journal of Memory and Language **109**, 104047 (2019)
4. Das, M., Amer-Yahia, S., Das, G., Yu, C.: MRI: meaningful interpretations of collaborative ratings. Proc. VLDB Endow. **4**(11), 1063–1074 (2011)
5. Deutch, D., Frost, N., Gilad, A.: Explaining natural language query results. VLDB J. **29**(1), 485–508 (2020)
6. El, O.B., Milo, T., Somech, A.: Towards autonomous, hands-free data exploration. In: Proc. CIDR (2020)
7. Francia, M., Gallinucci, E., Golfarelli, M.: COOL: a framework for conversational OLAP. Inf. Syst. **104**, 101752 (2022)
8. Francia, M., Golfarelli, M., Marcel, P., Rizzi, S., Vassiliadis, P.: Assess queries for interactive analysis of data cubes. In: Proc. EDBT. pp. 121–132 (2021)
9. Francia, M., Golfarelli, M., Rizzi, S.: A-BI$^+$: A framework for augmented business intelligence. Inf. Syst. **92**, 101520 (2020)

10. Francia, M., Marcel, P., Peralta, V., Rizzi, S.: Enhancing cubes with models to describe multidimensional data. Inf. Syst. Frontiers **24**(1), 31–48 (2022)
11. Gkesoulis, D., Vassiliadis, P., Manousis, P.: CineCubes: Aiding data workers gain insights from OLAP queries. Inf. Syst. **53**, 60–86 (2015)
12. Golab, L., Karloff, H.J., Korn, F., Srivastava, D.: Data auditor: Exploring data quality and semantics using pattern tableaux. Proc. VLDB Endow. **3**(2), 1641–1644 (2010)
13. Golab, L., Srivastava, D.: Exploring data using patterns: A survey and open problems. In: Proc. DOLAP@EDBT/ICDT. pp. 116–120 (2021)
14. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: A conceptual model for data warehouses. Int. J. Cooperative Inf. Syst. **7**(2-3), 215–247 (1998)
15. Hyde, J.: Foodmart. `https://github.com/julianhyde/foodmart-data-mysql`, accessed: 18/01/2021
16. Kellerer, H., Pferschy, U., Pisinger, D.: The multiple-choice knapsack problem. In: Knapsack Problems, pp. 317–347. Springer Berlin Heidelberg (2004)
17. Li, F., Jagadish, H.V.: Understanding natural language queries over relational databases. SIGMOD Record **45**(1), 6–13 (2016)
18. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern recognition **36**(2), 451–461 (2003)
19. Liu, F.T., Ting, K.M., Zhou, Z.: Isolation forest. In: Proc. ICDM. pp. 413–422 (2008)
20. Luo, Z.W., Ling, T.W., Ang, C.H., Lee, S.Y., Cui, B.: Range top/bottom k queries in OLAP sparse data cubes. In: Proc. DEXA. pp. 678–687 (2001)
21. Lyons, G., Tran, V., Binnig, C., Çetintemel, U., Kraska, T.: Making the case for query-by-voice with echoquery. In: Proc. SIGMOD. pp. 2129–2132 (2016)
22. Romero, O., Abelló, A.: On the need of a reference algebra for OLAP. In: Proc. DaWaK. pp. 99–110 (2007)
23. Saha, D., Floratou, A., Sankaranarayanan, K., Minhas, U.F., Mittal, A.R., Özcan, F.: ATHENA: an ontology-driven system for natural language querying over relational data stores. PVLDB **9**(12), 1209–1220 (2016)
24. Sarawagi, S.: Explaining differences in multidimensional aggregates. In: Proc. VLDB. pp. 42–53 (1999)
25. Sarawagi, S.: User-adaptive exploration of multidimensional data. In: Proc. VLDB. pp. 307–316 (2000)
26. Simitsis, A., Koutrika, G., Alexandrakis, Y., Ioannidis, Y.E.: Synthesizing structured text from logical database subsets. In: Proc. EDBT. pp. 428–439 (2008)
27. Song, L., Gan, J., Bao, Z., Ruan, B., Jagadish, H.V., Sellis, T.: Incremental preference adjustment: a graph-theoretical approach. VLDB J. **29**(6), 1475–1500 (2020)
28. Trummer, I., Wang, Y., Mahankali, S.: A holistic approach for query evaluation and result vocalization in voice-based OLAP. In: Proc. SIGMOD. pp. 936–953 (2019)
29. Zgraggen, E., Zhao, Z., Zeleznik, R.C., Kraska, T.: Investigating the effect of the multiple comparisons problem in visual analysis. In: Proc. CHI. p. 479 (2018)