Augmented Business Intelligence

Matteo Francia
DISI - University of Bologna
m.francia@unibo.it

Matteo Golfarelli
DISI - University of Bologna
matteo.golfarelli@unibo.it

Stefano Rizzi
DISI - University of Bologna
stefano.rizzi@unibo.it

ABSTRACT

Augmented reality allows users to superimpose digital information (typically, of operational type) upon real world entities. The synergy of analytical frameworks and augmented reality opens the door to a new wave of situated OLAP, in which users within a physical environment are provided with immersive analyses of local contextual data. In this paper we propose an approach that, based on the sensed augmented context (provided by wearable and smart devices), proposes a set of relevant analytical queries to the user. This is done by relying on a mapping between the entities that can be recognized by the devices and the elements of the enterprise data, and also taking into account the queries preferred by users during previous interactions that occurred in similar contexts. A set of experimental tests evaluates the proposed approach in terms of efficiency and effectiveness.

1 INTRODUCTION

With the disruptive advances in pervasive computing and industry 4.0, business intelligence is shifting its focus to the integration of (internal) enterprise and (external) contextual data. In this context, the synergy of analytical frameworks and augmented reality opens the door to *situated OLAP*, in which users within a physical environment are provided with immersive analyses of local contextual data. Indeed, Augmented Reality (AR), a variation of virtual reality, allows users to superimpose digital information upon real world entities [1], thus determining an augmented environment. Nowadays digital data returned to users are typically *operative*, meaning that they either provide information on the current state of visualized objects (e.g., the temperature of a machine) or on the current operations to be carried out (e.g., the instruction to use the machinery). Conversely, limited attention has been devoted to provide analytical reports that can be useful to let the user compare the current behavior of the visualized objects with their historical behavior.

This new goal opens relevant research challenges and revamps many issues related to business intelligence and recommendation systems [2]. Indeed, when working with high-dimensional contextual data (the multi-dimensional nature of the context is well understood [3, 4]), identifying insightful queries and visualizations is not trivial [5]: How is the perceived context mapped to enterprise data? Which data is salient with respect to the user analysis? How can data be retrieved in real time? How do users interact with the retrieved information?

To the best of our knowledge, none of the context-aware recommender systems proposed in literature addresses the above questions with reference to situated analytics in general, and to augmented reality in particular. In this paper we envision and formalize a foundation for *Augmented Business Intelligence* (A-BI), a framework empowering AR users with analytical information under visualization and time constraints. The analytical information returned comes in the form of reports obtained by running OLAP queries on the enterprise multidimensional cubes. The quantity of data returned must be limited in size and focused on the context to meet performance constraints and be easily interpretable by the user; furthermore, the intrinsic dynamics of AR applications asks for right-time (reasonably a few seconds) responsiveness of A-BI.

As shown in Figure 1, given a user situated in an environment and equipped with a smart device capable of perceiving relevant elements (i.e., the context) of such environment by means of sensors, A-BI provides real-time generation and visualization of multidimensional analyses out of contextual features. The context is modeled as a set of recognizable environment elements (e.g., a package) plus a set of user/environmental information (e.g., the user role and the room temperature); we assume the pattern recognition capabilities necessary to recognize them are provided by the AR smart device (specifically, through the *Context generation* component in Figure 1).

A-BI keeps track of user feedbacks on queries, and adopts collaborative filtering to provide a more focused experience. With reference to Figure 1, this task is carried out by the *Context interpretation* component by relying on the query log. Although A-BI supports the possibility to learn meaningful queries from the log, its capability of returning the right information primarily comes from some a-priori knowledge provided by domain experts. This choice is not simply a solution to the well-known cold start problem (i.e., the problem of providing significant recommendations when user feedbacks are still very few); it is rather a design choice aimed at enabling the system to give a useful answer in complex context scenarios, where learning from the log would require too many examples. The a-priori knowledge is modeled through a set of weighted mappings between the potentially recognizable environment entities (stored in the *dictionary*) and the cube multidimensional ones. Instead of proposing the most relevant query only (called *maximal query*), A-BI proposes a set of alternative queries to the user; all of them are related to the current context but they are different enough to offer to the user different flavors of the same information. This phase is implemented by the *Diversification* component.

A-BI can be applied to different application domains ranging from healthcare to factories; for this reason, the main modeling choices underlying our approach (e.g., how to define the relevance of an object in a context) have been formalized in a domain-independent fashion, while domain-dependent examples are provided in the context of AR in a factory where a smart device is coupled with a sensor data warehouse [6]. To better understand the user/system interaction suppose that Bob, a machinery inspector, is situated in a factory to investigate a malfunctioning packaging machine that —possibly due to overheating— produces dented packages. Environmental data is gathered and integrated from wearable and smart devices into a single context representation. So, when looking at the machine, the AR helmet Bob is wearing recognizes the machinery and, knowing the role of Bob, suggests the set of analytical queries that are more relevant according both to a-priori knowledge and to previous feedbacks given by users in similar contexts. Relevant figures

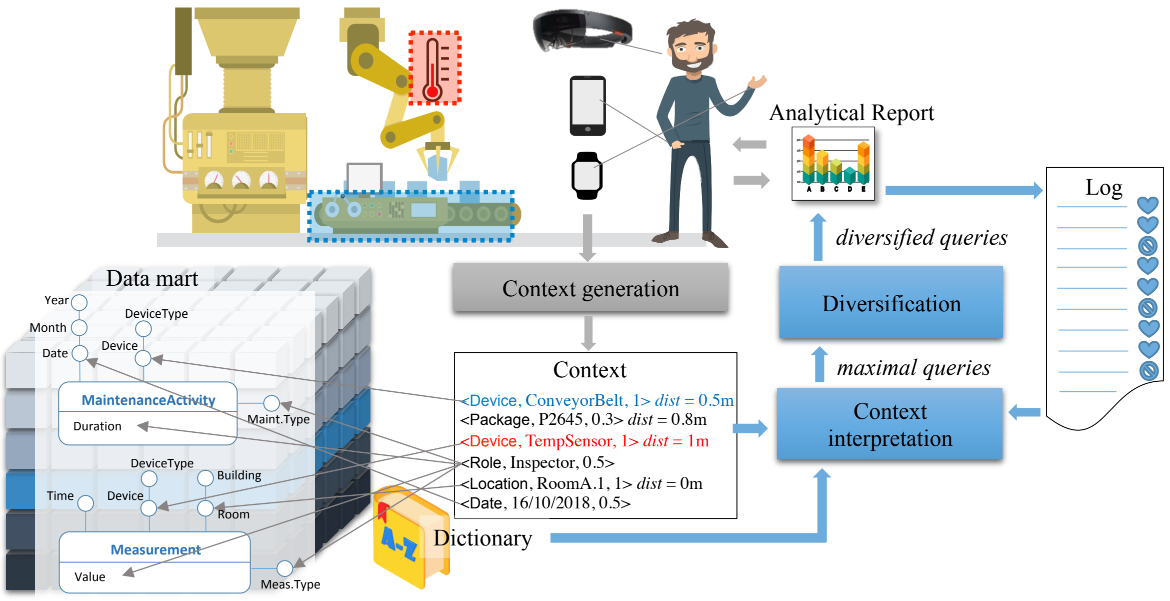


Figure 1: Overview of the A-BI framework; engaged entities are enclosed in dotted rectangles, while grayed elements are outside the paper scope

could be the number of defective packages along time and the dates and types of previous maintenance activities for that machinery. Bob can either execute one of the proposed queries or express a new query to obtain a different report. Finally, Bob gives his feedback on the proposed queries, which is stored in the log.

To sum up, the main contributions of this paper are:

(1) We envision an A-BI framework, its functional architecture, and the user/system interaction process;
(2) We provide a method to model the a-priori system knowledge by mapping context entities to relevant multidimensional elements;
(3) We provide an efficient algorithm to generate relevant and diverse queries to be returned to the user;
(4) We propose a collaborative filtering approach to let the system learn from user feedbacks.

The remainder of the paper is organized as follows. Section 2 describes the related work in the field of context-aware recommendation systems. Section 3 formalizes the A-BI framework. Section 4 explains how the context is used to derive a starting query for the diversification process by also considering the user feedbacks stored in the log, while Section 5 describes how relevant and diverse queries are generated from that query. Section 6 includes the results of experimental tests that measure the performance of A-BI. Finally, Section 7 sums up our contribution and gives future research directions.

2 RELATED WORK

A-BI can be classified as a *recommender system* in the area of of business intelligence, based on a context made of *augmented entities.* Although the huge amount of work in each area, to the best of our knowledge no approach lies at their intersection.

Over the years, scholars have highlighted the importance of exploiting contextual information to provide focused recommendations with the nature of contexts being quite heterogeneous, for instance space and time [7], query logs [8, 9], statistics on results [10] or databases [11], user interests [12], and social data [13]. Given such heterogeneity, other contributions address the integration of contextual data (e.g., [13–15]) to provide a common ground for further analyses and recommendations.

The previous context types have been widely adopted in several recommender system applications where the recommendation process is activated by an explicit user-defined input statement (e.g., query or keywords). Examples of applications are web query categorization [16, 17], recommendation [12, 18], and diversification [19]; query completion [8, 9]; localized web keywords suggestion [7]; and interactive exploration of databases [20]. The two main differences between A-BI and these work are (1) the multidimensional nature of the data handled and returned, and (2) the nature of the context as well as the type of user/system interaction that triggers the recommendation. As to (1), multidimensional and hierarchical data support recommendations at different granularities, which intrinsically makes finding the best recommendation more complex; as to (2), physical contexts require ad hoc solutions to choose the relevant context elements due to application specificities (e.g., engagement) and to the possibility of having in the context elements that are perceived but that are not relevant for the user.

Recommender systems in business intelligence applications are well surveyed in [21]. Recommendations typically involve multidimensional queries [22] or sessions [17] (i.e., query sequences) using query logs as contexts. These approaches are based on collaborative filtering techniques that do not synthesize new queries from existing ones, but pick queries from the log depending on their similarity score. Conversely, A-BI allows the generation of queries not already present in the log by combining similar queries from the log and contextual information into a set of diverse queries. This assumption collocates A-BI as a hybrid approach to recommendation [21], differentiating A-BI from the above-mentioned contributions in multidimensional recommendation systems. Note that diversification and multiple recommendations are used to better meet user interests [23]. A further advantage of A-BI over pure collaborative filtering approaches is that A-BI does not suffer from the so-called cold start problem, since it is able to return an appropriate recommendation even when the log is empty [24].

In the area of AR, contexts play an even more central role. There, a context is the set of entities recognized in the environment that acts as situated stimulus (i.e., object properties) to be translated into inputs for a search query, is augmented with virtual information, and is returned to the user [25]. Scholars focus on finding proper visualization of relevant information [25, 26], with visualization typically embedded in physical objects. While [26] considers multidimensional data, it is not specified how the process of information retrieval and analysis of data at multiple level of details is carried out. Also, these approaches do not include collaborative filtering to discover potentially useful information. Interestingly, although [26] does not consider analytical data, it introduces a mantra for situated analytics: "details first, analysis, then context-on-demand" which contradicts the well-known mantra "overview first, zoom and filter, then details-on-demand" [27] of classical visualization systems. Indeed, when it comes to pure augmented visualizations, information is directly attached to single entities [25, 26], assigning higher priority to details than to generic information.

3 BASICS

We start this section by introducing a formal setting to manipulate multidimensional data; for simplicity we will work on a single cube and consider linear hierarchies.

Definition 3.1 (Hierarchy). A *hierarchy* is defined as a triple $h = (L_h, \succeq_h, \geq_h)$ where:

(i) L_h is a set of categorical *levels*; each level $l \in L_h$ is coupled with a *domain* $Dom(l)$ including a set of *members* (all domains are disjoint);
(ii) $\succeq_h$ is a *roll-up* total order of L_h; and
(iii) $\geq_h$ is a *part-of* partial order of $\bigcup_{l \in L_h} Dom(l)$.

Exactly one level $dim(h) \in L_h$, called *dimension*, is such that $dim(h) \succeq_h l$ for each other $l \in L_h$. The part-of partial order is such that, for each couple of levels l and l' such that $l \succeq_h l'$, for each member $u \in Dom(l)$ there is exactly one member $u' \in Dom(l')$ such that $u \geq_h u'$.

Definition 3.2 (Cube). A *cube* is defined as a triple $C = (H, M, \omega)$ where:

(i) H is a set of hierarchies;
(ii) M is a set of numerical measures, with each measure $m \in M$ coupled with one aggregation operator $op(m) \in \{\texttt{sum}, \texttt{avg}, \ldots\}$; and
(iii) ω is a partial function that maps the tuples of members for the dimensions of H to a numerical value for each measure $m \in M$.

The levels, members, and measures of the cube C are given the generic name of *md-elements* of C.

Example 3.3. Let us consider cube MaintenanceActivity in our working example which, as sketched in Figure 1, includes hierarchies Date, Device, and MaintenanceType. Maintenance activities are described by measure Duration (coupled with the sum operator). A member of the Device attribute is PackagingMachine, while a member of MaintenanceType is Oiling. □

In the A-BI framework, cubes are queried through GPSJ (Generalized Projection / Selection / Join) queries, a well-knows class of queries that was first studied in [28]. A GPSJ query is composed of joins, selection predicates, and aggregations.

Definition 3.4 (Query). A query q on cube $C = (H, M, \omega)$ is a triple $q = (G_q, P_q, M_q)$ where G_q is the query *group-by set*, i.e., a subset of levels in the hierarchies of H; P_q is a set of Boolean clauses whose conjunction defines the selection predicate for q; $M_q \in M$ is the set of measures whose values are returned by q.

Example 3.5. With reference to the MaintenanceActivity cube, the query asking for the average duration of maintenance activities for each month of 2018 and each device type is defined by

$$G_q = \{\text{Month}, \text{DeviceType}\}$$
$$P_q = \{(\text{Year} = 2018)\}$$
$$M_q = \{\text{Duration}\}$$

□

Enterprise cubes are the data source for the analytical reports to be returned to users according to the environment as perceived by the AR device. The set of data possibly perceived are listed in a dictionary that, intuitively, defines the device capabilities. These data are not limited to physical objects recognized in the environment through a pattern recognition process, but may include user-related information such as the user role as well as environmental properties such as the room temperature.

Definition 3.6 (Dictionary). A *dictionary* $\mathcal{D}$ is a set of keys, each coupled with a domain of values (all domains include a NULL value). Each pair $d = \langle key, value \rangle$ such that *value* belongs to the domain of *key* is called an *entry* of $\mathcal{D}$.

NULL values are used whenever the device through which the user perceives the environment successfully classifies an object but is not capable of labelling the specific instance sensed (e.g., a package is recognized but the package code cannot be read).

The power of the A-BI framework comes from the ability to bind the perceived information to the cube md-elements. This capability is rooted in a-priori knowledge that specifies which md-elements can be interesting for the user when a given dictionary entry is perceived. This knowledge is defined through a dictionary-to-cube mapping established by a domain expert at setup time.

Definition 3.7 (Mapping). A *mapping* from dictionary $\mathcal{D}$ to cube C is a (partial) multivalued function μ that maps each entry d of $\mathcal{D}$ to a set of md-elements of C. Each md-element $e \in \mu(d)$ has a mapping weight, $w(d, e) \in (0, 1]$.

Although a discussion about how mappings can be established is out of the scope of this paper, we remark that this does not necessarily have to be done manually for all the cube members, which would be tedious for attributes with large domains, but it can be largely automated (for instance, by providing universally-quantified rules such as $\mu(\langle \text{Device}, value \rangle) = \{value\}\ \forall value \in Dom(\text{Device})$). The mapping function is multivalued since many md-elements may be of interest for each dictionary entry; this typically happens for hierarchy levels, which can be all interesting —even if with different values of w. For example, when some device is perceived, besides showing data for that specific device, also showing aggregated data for the device type could be interesting. Through mapping weights, domain experts give an a-priori quantification of the interest of each md-element for analyses when a given entry is part of the context.

Example 3.8. The dictionary for our example includes, among the others, entries related to machines and their components (key = Device, $value$ = ConveyorBelt, PackagingMachine, etc.), and user roles (key = Role, $value$ = Inspector, Operator, etc.). The mapping from this dictionary to the cube of Example 3.3 may look like this (see Figure 1):

$$\begin{aligned}\mu(\langle \text{Device, ConveyorBelt}\rangle) &= \{\text{ConveyorBelt}\},\\ \mu(\langle \text{Role, Inspector}\rangle) &= \{\text{Duration},\\ &\qquad \text{MaintenanceType}\},\\ \mu(\langle \text{Date, 16/10/2018}\rangle) &= \{\text{Date}\},\\ w(\langle \text{Device, ConveyorBelt}\rangle, \text{ConveyorBelt}) &= 0.5\end{aligned}$$

where the first line refers to a member, the second one to a measure, the third and fourth ones to a level. The mapping weight of member ConveyorBelt when the conveyor belt device is perceived is 0.5. □

4 CONTEXT INTERPRETATION

In this section we show how, given a set of perceived entities, A-BI produces the most relevant query, i.e., the one whose results would be more useful to the user.

4.1 Take the context...

The A-BI starting point is the context: a list of dictionary entries corresponding to the currently perceived environment entities. More formally:

Definition 4.1 (Context). A *context* T over dictionary $\mathcal{D}$ is a set of entries of $\mathcal{D}$; each entry $d \in T$ is coupled with a weight $w(T,d) \in (0,1]$.

The value of the weight for each entry may depend on different factors, depending on the application domain. Non-perceived entities (i.e., for which it would be $w(T,d) = 0$) are not included in the context. In our case study we assume that a subset of entries are *engaged*, meaning that they have explicitly been indicated by the user as being part of her current focus of interest; for these entries, the weight is always 1. For the other entries, the weight is inversely proportional to the distance between the user and the specific object being observed.

Given a context, the mapping function identifies the relevant md-elements, i.e., those that will be involved in the queries to be issued against the cube.

Definition 4.2 (Image). Given context T over dictionary $\mathcal{D}$, cube C, and mapping μ from $\mathcal{D}$ to C, the *image* of T through μ is $I_\mu(T) = \bigcup_{d \in T} \mu(d)$, i.e., the set of md-elements of C that are mapped through μ to at least one entry in T.

Example 4.3. A possible context is the one depicted in Figure 1, where Bob is checking a conveyor belt placed in room A.1:

$$\begin{aligned}T = \{&\langle \text{Device, ConveyorBelt}, 1\rangle,\\ &\langle \text{Role, Inspector}, 0.5\rangle,\\ &\langle \text{Date}, 16/10/2018, 0.5\rangle\}\end{aligned}$$

The ConveyorBelt device is engaged. The image of T through μ is

$$I_\mu(T) = \{\text{ConveyorBelt, Duration, MaintenanceType, Date}\}$$

□

The image includes the set of md-elements relevant to a context according to the mapping, but it does not specify how they will be used to generate the queries to be proposed to the user when that context is sensed. Indeed, given an image, several queries can be generated, each including a subset of the md-elements in the image. This is ruled by the definition of *compatible* query.

Definition 4.4 (Compatible and Equivalent Query). A query $q = \langle G_q, P_q, M_q\rangle$ is *compatible* with context T if it *refers* to *at least one* of the md-elements in the image of T; specifically,

- a level l is referred by q if $l \in G_q$;
- a measure m is referred by q if $m \in M_q$;
- a member u is referred to by q if $(lev(u) = u) \in P_q$ and $lev(u) \in G_q$, being $lev(u)$ the level whose domain includes u.

With a slight abuse of notation, we will write $e \in q$ to state that q refers to md-element e. Let Q_T be the set of queries compatible with T; the query *equivalent* to T is the query $q_{eq}(T) \in Q_T$ that refers to *all* md-elements in $I_\mu(T)$.

4.2 ...add the log...

The a-priori knowledge expressed through a mapping does not enable the system to learn by considering how user interests evolve, which instead could lead to picking different md-elements when proposing queries or to choosing one of them more/less frequently. To this end, A-BI exploits the history of previous interactions, stored in the query log, by means of a collaborative filtering approach. The log stores, for each context, all the queries proposed to the user and the specific one chosen for execution.

Definition 4.5 (Log). A log L is an ordered list of triples $\langle T, q, f\rangle$ where T is a context, q is a query, and f (feedback) is 1 if the user accepted the query, -1 if she rejected the query.

Example 4.6. With reference to the context T proposed in Example 4.3, a possible log is $L = (\langle T, q_1, -1\rangle, \langle T, q_2, 1\rangle)$, where

$$\begin{aligned}q_1 = \langle&\{Date, Device\},\\ &\{(\text{Device} = \text{ConveyorBelt})\},\\ &\{\text{Duration}\}\rangle\\ q_2 = \langle&\{Month, Device\},\\ &\{(\text{Device} = \text{ConveyorBelt})\},\\ &\{\text{Duration}\}\rangle\end{aligned}$$

While q_1 has been rejected, q_2 (which is a roll-up of q_1 on the Date hierarchy) has been accepted. □

A log entry related to context T' should impact the recommendations related to the current context T only if the two contexts are similar, since it is reasonable to assume that the user will have similar behaviors in similar contexts. Context similarity is computed as the similarity between their two equivalent queries, $sim(q_{eq}(T), q_{eq}(T'))$, which in turn is computed as in [29]. The similarity function $sim(q, q')$ between two queries $q = \langle G_q, P_q, M_q\rangle$, $q' = \langle G_{q'}, P_{q'}, M_{q'}\rangle$ combines three components, related to group-by sets, selection predicates, and measures, respectively. In particular, the group-by set similarity considers the distance between the levels involved in the query group-by sets; the selection similarity takes into account both the levels and the members that form the selection predicates; the measure similarity is based on the Jaccard index. Finally, the similarity between two queries is defined as the weighted average of the three similarity components. Following [29], we assume the three components to be equally significant.

Given log L, the image $I_\mu(T)$ of context T is extended to take previous user interactions into account as follows. Let $L_T \subseteq L$ be the subset of triples whose context is similar to T:

$$L_T = \{\langle T', q, f\rangle \mid sim(q_{eq}(T), q_{eq}(T')) \geq \epsilon\}$$

where ϵ is the similarity threshold. Then, $I_\mu(T)$ is extended with all the md-elements referred to by the queries in L_T:

$$I^*_\mu(T) = I_\mu(T) \cup \{e : \exists\langle T', q, f\rangle \in L_T \mid e \in q\}$$

In this way, $I^*_\mu(T)$ includes all the md-elements that are relevant to context T *either according to the mapping or to the previous user experience.* We define the log relevance to T of md-element $e \in I^*_\mu(T)$ as the weighted number of times e has been accepted by the user ($f = 1$) over the number of times it has been proposed; weighting is based on the similarity between the current context T and the considered log context T':

$$\rho_T(L, e) = \frac{1 + \sum_{\langle T',q,f\rangle \in L_T(e), f=1} sim(q_{eq}(T), q_{eq}(T'))}{2 + \sum_{\langle T',q,f\rangle \in L_T(e)} sim(q_{eq}(T), q_{eq}(T'))}$$

where $L_T(e)$ is the subset of tuples in L_T such that q refers to e. To avoid relevance to be 0 when e has never been accepted, a Laplace smoothing is applied in the formula above. Noticeably, the impact of Laplace smoothing decreases as the cardinality of $L_T(e)$ increases, that is, the weight tends to 0 if several queries referring e have been proposed but never accepted by the user. Conversely, it tends to 0.5 if only few queries referring e have been proposed.

It is now possible to define the *relevance* to T of each md-element $e \in I^*_\mu(T)$ by taking into account, for each context entry d that maps to e, not only the entry weight $w(T, d)$ and the mapping weight $w(d, e)$, but also the log relevance $\rho_T(L, e)$:

$$rel_T(e) = \begin{cases} \sum_{d \in T | e \in \mu(d)} w(T, d) \cdot w(d, e), \text{ if } L_T(e) = \varnothing \\ \sum_{d \in T | e \in \mu(d)} w(T, d) \cdot w(d, e) \cdot \rho_T(L, e), \text{ otherwise} \end{cases}$$

(note that log relevance is not considered —i.e., it does not affect the overall relevance— if e is never referred in a log query).

Example 4.7. With reference to the image $I_\mu(T)$ from Example 4.3 and to the log entries in Example 4.6, the extended image is

$$I^*_\mu(T) = \{\text{ConveyorBelt}, \text{Duration}, \text{MaintenanceType}, \text{Date}, \text{Month}\}$$

Month has been added to $I_\mu(T)$ as part of a query with a positive feedback. As to $rel_T(\text{Date})$ and $rel_T(\text{Month})$, assuming that the mapping weight $w(d, \text{Duration}) = 1$, and that the mapping weight for the remaining levels and members is set to 0.5, it is

$$\begin{aligned} rel_T(\text{Device}) &= 0.25, \\ rel_T(\text{ConveyorBelt}) &= 0.25, \\ rel_T(\text{Duration}) &= 0.25, \\ rel_T(\text{MaintenanceType}) &= 0.17, \\ rel_T(\text{Month}) &= 0.13, \\ rel_T(\text{Date}) &= 0.08 \end{aligned}$$

□

4.3 ...get the queries

In principle, given a context T, its equivalent query $q_{eq}(T)$ might be directly proposed to the user. Unfortunately, equivalent queries are often *monster queries*, i.e., quite complex queries with very high cardinalities. A monster query would be obtained when the number of md-elements in the image is high because several entities were sensed in the environment so the context includes a large number of entries. Unfortunately, monster queries are particularly undesirable in AR applications since:

- High-cardinality queries take a long time to be computed, transferred to the user smart device, and visualized.
- While working on the field, users must be quick and reactive, while the results of monster queries are hard to be interpreted.

In the A-BI framework, monster queries are avoided in two ways: (i) by posing an upper bound γ to the query cardinality, and (ii) by considering only the most relevant md-elements in the image when generating the queries to be proposed to the user.

As to (i), given query $q = \langle G_q, P_q, M_q\rangle$, the expected cardinality of its result, denoted $card(q)$, can be estimated as follows:

$$card(q) = card(G_q) \times \prod_{p \in P_q} selectivity(p)$$

where $card(G_q)$ is the cardinality of the query group-by set (estimated for instance using the Cardenas formula [30, 31]) and $sel(p)$ is the selectivity of each simple predicate p belonging to P_q. Note that we can safely use this formula to estimate $card(q)$ because, as a consequence of the way we create queries in our approach, all predicates in P_q are always *external*, i.e., they are expressed on levels that are less or equal to a level in G_q [32] in the roll-up order.

As to (ii), before generating the maximal query for a context we remove from the context image $I^*_\mu(T)$ all the md-elements e whose relevance $rel_T(e)$ is below a given threshold η.

Definition 4.8 (Query Relevance and Maximal Query). Given context T, let Q_T be the set of queries that refer to any subset of md-elements in the image of T through mapping μ. The *relevance* to T of a query $q \in Q_T$ is defined as

$$rel_T(q) = \frac{\sum_{e \in q} rel_T(e)}{\sum_{e \in I^*_\mu(T)} rel_T(e)}$$

The *maximal* query for T is the query $q_{max}(T) \in Q_T$ that has maximum relevance among all those in Q_T such that $card(q) \leq \gamma$.

Clearly, the number of queries in Q_T increases exponentially with $|T|$. Finding the maximal query can be formalized as a Knapsack problem on the md-elements in T, considering γ as the knapsack capacity. The Knapsack problem is an NP-hard optimization problem, so in Algorithm 1 we propose a greedy approach to compute the maximal query in real time. Starting from the image, we first include all the predicates to produce the smallest (i.e., most focused) result set and then, while the result set cardinality is below the threshold, we incrementally extend the group-by set to produce progressively larger result sets.

Given the extended image I^*_μ of context T, the cardinality threshold γ, and the relevance threshold η, the algorithms works as follows. At first, in Line 1 we remove from I^*_μ the md-elements whose relevance is below η. Then, we initialize the maximal query by selecting all the members in I^*_μ to create the conjunctive predicate P_q (intuitively, this is the "most focused" predicate), by adding all the corresponding levels to the group-by set, and

Algorithm 1 Maximal query generation

```
Require: $I^*_\mu$: extended image of context $T$, $\gamma$: cardinality threshold, $\eta$: relevance threshold
Ensure: $q_{max}(T)$: maximal query
 1: $I^*_\mu \leftarrow I^*_\mu \setminus \{e \in I^*_\mu, rel_T(e) < \eta\}$        ▷ Drop low-relevance md-elements from the extended image
 2: $P_q \leftarrow CreatePred(\{e \in I^*_\mu, e$ is a member$\})$ ▷ Create a predicate as the conjunction of IN clauses, each including all the members of the same level in the extended image
 3: $G_q \leftarrow \{lev(e), e \in I^*_\mu, e$ is a member$\}$        ▷ Create a group-by set including the levels of all members in the extended image
 4: $M_q \leftarrow \{e \in I^*_\mu, e$ is a measure$\}$        ▷ Create a set of measures including all those in the extended image
 5: $q \leftarrow \langle G_q, P_q, M_q \rangle$        ▷ Initialize the current maximal query
 6: $A \leftarrow \{e \in I^*_\mu \setminus G_q, e$ is a level$\}$        ▷ Initialize a set of candidate levels to extend $G_q$
 7: $q' \leftarrow q$        ▷ New candidate maximal query
 8: while $card(q') \leq \gamma$ do        ▷ While the candidate query cardinality is below threshold...
 9:     $q \leftarrow q'$        ▷ ...update the current maximal query
10:     if $A = \emptyset$ then        ▷ If the search space is not exhausted...
11:         break
12:     $l \leftarrow argmax_{e \in A}(rel_T(e))$        ▷ ...pick the most relevant candidate level,
13:     $A \leftarrow A \setminus \{l\}$        ▷ remove from the set of candidate levels,
14:     $G_q \leftarrow G_q \cup \{l\}$        ▷ add it to the group-by set,
15:     $q' \leftarrow \langle G_q, P_q, M_q \rangle$        ▷ and update the candidate query
16: end while
17: return $q$
```

by adding all the measures in I^*_μ (Lines 2–5). Now we iterate to progressively add to the group-by set new relevant levels taken from the image, but only as long as the cardinality constraint is met (Lines 8–16). Finally, the last query with a cardinality below γ is returned to the user (Line 17). Note that, if the candidate query defined in line 5 violates the cardinality constraint γ, it is immediately returned since all the transformations that follow would further increase its cardinality. In this specific case, constraint enforcement will be ensured by the diversification phase (see Algorithm 2).

Example 4.9. With reference to the context T from Example 4.3 and to its extended image from Example 4.7, we apply Algorithm 1 to generate the maximal query for T with $\gamma = 20$ and $\eta = 0.1$. At first, Date is pruned from I^*_μ since its relevance is below η. Then P_q is initialized to {(Device = ConveyorBelt)}, G_q to {Device}, M_q to {Duration}; levels MaintenanceType and Month are stored in A for possible further inclusion into G_q. The maximal query is initially set to $\langle G_q, P_q, M_q \rangle$, with 1 being its cardinality. At this time, the algorithm attempts to add MaintenanceType to the query group-by set. Let the cardinality of ⟨{Device, MaintenanceType}, {(Device = ConveyorBelt)}, {Duration}⟩ be 5. Since the cardinality of the candidate maximal query is below threshold, the current maximal query is updated. The algorithm attemps to add Month to the query group-by set. Since this increases the query cardinality by a factor equal to the number of months being monitored in the MaintenanceActivity cube, the cardinality of the new candidate query ⟨{Device, MaintenanceType, Month}, {(Device = ConveyorBelt)}, {Duration}⟩ will supposedly be larger than 20, so the previous query ⟨{Device, MaintenanceType}, {(Device = ConveyorBelt)}, {Duration}⟩ is returned. The SQL representation of the maximal query is

```
SELECT Device, MaintenanceType, sum(Duration)
FROM MaintenanceActivity
WHERE Device = ConveyorBelt
GROUP BY Device, MaintenanceType
```

□

5 QUERY DIVERSIFICATION

Given a context T, the maximal query $q_{max}(T)$ is the more relevant query that meets the cardinality constraint. Nonetheless, to better meet the user's desiderata, it may be useful to return a set of alternative queries that, though being still related to T, are different enough from each other and from $q_{max}(T)$ to offer different flavors of the same information to the user. This general idea is often practiced in the literature and referred to as *diversification* [23].

Given the set of queries Q_T compatible with T, the problem of diversification consists in finding the set of top-N queries $R \subseteq Q_T$ that maximizes diversity and relevance with respect to user analyses [23]. The diversity $div(q, R)$ of a query q with respect to R can be defined as in [33]

$$div(q, R) = \frac{1}{|R|} \sum_{q' \in R} (1 - sim(q, q'))$$

with $div(q, R) = 1$ when R is empty.

Finding the optimal set of diverse queries is NP-hard with reference to the cardinality of Q_T which, in turn, is exponential in the image cardinality $|I^*_\mu(T)|$. Considering the real-time constraint related to every AR application, we propose in the following a greedy approach that starting from $q_{max}(T)$ generates progressively different queries. Diversification is obtained through three OLAP-based generative primitives, each applied to the previous query $q = \langle G, P, M \rangle$ to return a set of new queries:

- *roll*(q) returns a set of queries whose group-by set is coarser than G; each of these queries is obtained by replacing in G one level l with its immediate successor in the roll-up partial order. For each $q' \in roll(q)$, it is $card(q') \leq card(q)$. Only queries whose predicates are all external are returned.
- *drill*(q) returns a set of queries whose group-by set is finer than G; each of these queries is obtained by replacing in G one level l with its immediate predecessor in the roll-up partial order. For each $q' \in roll(q)$, it is $card(q') \geq card(q)$.

Algorithm 2 Diversification

```
Require: $q_{max}(T)$: maximal query for context $T$, $\gamma$: cardinality threshold, $\theta$: diversity threshold, $N$: number of diverse queries
Ensure: $R$: diverse queries
1:  $R \leftarrow \varnothing$                                   ▷ Result set
2:  $Q \leftarrow \{q_{max}(T)\}$                                ▷ Search space
3:  while $(|R| < N) \wedge (Q \neq \varnothing)$ do
4:      $q \leftarrow argmax_{q' \in Q}(div(q', R))$             ▷ Most diverse query
5:      $Q \leftarrow Q \setminus \{q\}$
6:      if $(card(q) \leq \gamma) \wedge (div(q,R) \geq \theta)$ then
7:          $R \leftarrow R \cup \{q\}$                          ▷ $q$ is added to the result
8:          if $q = q_{max}(T)$ then
9:              $Q \leftarrow Q \cup Extend(q)$                  ▷ Apply primitives
10:     else                                                     ▷ Continue the search
11:         $Q \leftarrow Q \cup Extend(q)$                      ▷ Apply primitives
12: end while
13: return $R$
```

- *slice*(q) returns a set of queries whose selection predicate is less selective than P; each of these queries is obtained by replacing one of the `IN` clauses in P, defined on a set of members $u_1, \ldots, u_n$, with a clause on the member(s) that are the immediate successors of $u_1, \ldots, u_n$ in the part-of order. For each $q' \in roll(q)$, it is $card(q') \geq card(q)$.

The queries returned by all these operators are progressively less similar to the maximal query, so their relevance is lower than the one of q. Besides, only queries compatible with the context are returned.

Example 5.1. Given $q = \langle\{Device, Month\}, \{\text{Device} = \text{ConveryorBelt}\}, \{Duration\}\rangle$, examples of queries returned by our three primitives are, respectively,

$$\begin{aligned} q_{roll} &= \langle\{\text{Device, Year}\},\\ &\quad \{(\text{Device = ConveyorBelt})\},\\ &\quad \{\text{Duration}\}\rangle\\ q_{drill} &= \langle\{\text{Device, Date}\},\\ &\quad \{(\text{Device = ConveyorBelt})\},\\ &\quad \{\text{Duration}\}\rangle\\ q_{slice} &= \langle\{\text{Device, Month}\},\\ &\quad \{(\text{DeviceType = PackagingMachine})\},\\ &\quad \{\text{Duration}\}\rangle \end{aligned}$$

□

The diversification process is described in Algorithm 2. Given the maximal query q_{max}, the cardinality threshold γ, a diversity threshold θ, and the number of desired diverse queries N, the set of diverse queries R is initialized to the empty set (Line 1) and the search space Q is initialized to q_{max} (Line 2). The most diverse query, q, is picked from the search space (Line 4); if its cardinality is below γ and its diversity from the queries in R is higher than θ, then q is added to R (Lines 6–7). Note that at the first step, when the search space contains only $q_{max}(T)$, function $Extend(q)$ is invoked to apply our three primitives to $q_{max}(T)$ (Lines 8–9); specifically, $roll(q)$ is alway applied since it decreases the query cardinality, while $drill(q)$ and $slice(q)$ are applied only if $card(q) \leq \gamma$. Otherwise, if either $card(q)$ is too high or q is not diverse enough from the queries in R, the search space is extended with the queries generated by $Extend(q)$ (Line 11). Lines from 4 to 11 are repeated until either the cardinality of the result set overcomes N or the search space Q is empty (Line 3).

Intuitively, Algorithm 2 explores the search space Q_T around $q_{max}(T)$ assuming that the lower the similarity between q' and $q_{max}(T)$, the lower the relevance of q'. The queries added to the result set R are no longer considered for differentiation aimed at maximizing the probability of finding a a new different query with high relevance: indeed, a query q'' obtained by extending $q' \in R$ will probably be very similar to q' and less relevant. This rule does not apply to $q_{max}(T)$, which is the starting point for exploration and must be always expanded and added to R (except the specific case in which it violates the cardinality constraint).

Example 5.2. Let $q_{max}(T) = \langle\{Device, Month\}, \{\text{Device} = \text{ConveryorBelt}\}, \{Duration\}\rangle$ be the maximal query, with $card(q_{max}(T)) \leq \gamma$. At the first iteration, $q_{max}(T)$ (the only query in the search space Q) is picked. Since $div(q_{max}(T), R) = 1$ (R is empty), R is extended with $q_{max}(T)$, and Q is extended, among the others, with q_{roll}, q_{drill}, and q_{slice} (see Example 5.1). At the second iteration, let q_{drill} be the most diverse query. Since q_{drill} is quite similar to the maximal query, it is likely that $div(q_{drill}, R)$ is below θ. In this case, q_{drill} is not added to R, and Q is extended by applying $Extend(q_{drill})$. The iteration continues until N diverse queries are generated, or the search space Q is exhausted. □

6 EXPERIMENTAL TESTS

In this section we evaluate the A-BI framework in terms of both effectiveness (to what extent the proposed queries meet the user's interests) and efficiency. Tests were carried out on a synthetic benchmark since, in this work, we assume the problem of context generation to be addressed by the smart device and, to the best of our knowledge, no AR open dataset exists.

The user-system interaction is as follows. The user is moving in a factory of 10 rooms, and, while moving, she collects one view of each room. In each view the smart device recognizes a set of entities that belong to the dictionary and lists them into a context. Starting from each context, the A-BI framework proposes a set of queries to the user. Finally, the user either chooses one of the proposed queries or formulates an additional query that is slightly different from the ones proposed. After some time, the user ends her exploration of the factory and moves back through the same rooms. Knowing the precedent behavior of the user, the A-BI framework exploits collaborative filtering to propose a new set of queries (generated by taking the query log into account) that better suit her interests. We denote with α the number of times the user has visited the factory (i.e., how many times each context has already been sensed by the user's devices before the current visit).

This interaction is simulated by randomly generating 10 seed contexts, each corresponding to a different room, in such a way that these contexts differ significantly from each other. Then, to simulate multiple visits of each room, small context variations are generated starting from each seed (specifically, one for each visit; the idea is that each room is perceived with slight differences in each visit). The number of entities recognized in each room (i.e., the context cardinality) ranges between 5 and 15. Each test is repeated 10 times and the average behavior is considered.

We executed our tests against a cube including 5 linear hierarchies with 5 levels of details each. Each dimension has 64 members, determining a maximum cube cardinality of about 10^9.

Table 1: Notation summary

Notation	Meaning
T	Context (corresponds to a view of a factory room)
q_{max}	Maximal query
q_u	User query
q_{div}	Diverse query most similar to q_u
q_{log}	Log query most similar to q_u
$\alpha \in [0, 8]$	Number of times the user has already seen a context
$\beta \in [0.6, 1]$	Similarity between q_u and q_{max}
$\gamma = 100$	Query cardinality threshold
$\epsilon = 0.5$	Context similarity threshold
$\eta = 0.5$	Relevance threshold
$\theta \in [0.05, 0.2]$	Diversity threshold
$N \in [2, 8]$	Number of diverse queries generated

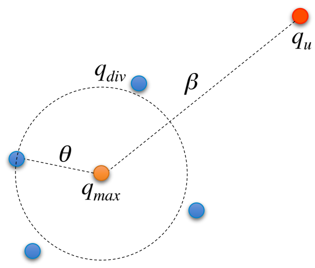


Figure 2: The user query q_u, the maximal query q_{max}, and the set of diverse queries (in blue); among them, q_{div} is the one most similar to q_u

The dictionary includes one key for each md-element (i.e., we assume the smart device can recognize each single element of the cube); each dictionary entry d is one-to-one mapped to the corresponding md-element e with mapping weight $w(d, e)$ randomly ranging in $[0.2, 1]$. Additionally, for each context, we call

- q_{max} the maximal query generated by Algorithm 1.
- q_u the query chosen by the user. We denote with β the similarity between the user query and the maximal query ($\beta = sim(q_u, q_{max})$). The lower the value of β, the higher the difference between the user and maximal queries; if $\beta = 1$, the user exactly chooses the maximal query.
- q_{div} the query most similar to q_u among those generated by the diversification process, when the log is ignored.
- q_{log} the query most similar to q_u among those generated by the diversification process, when the log is taken into account. At each visit, 10 contexts are shown to the user; so, after each iteration, 10 contexts are added to the log with the corresponding user choices.

At the first visit ($\alpha = 0$), the log is empty so $q_{log} \equiv q_{div}$. In theory, q_{div} should be more similar to q_u than q_{max} since it results from diversification, and q_{log} should further improve over q_{div} since it also uses the log.

A notation summary is provided in Table 1. To help the reader understand the role of each query, a visual representation of q_{max}, q_{div}, and q_{log} is depicted in Figure 2, where the query space is represented as a Cartesian plane with Euclidean distances.

6.1 Effectiveness

Figure 3a evaluates the benefit of diversification and of collaborative filtering by showing how similar q_{max}, q_{div}, and q_{log} are to q_u for different values of β and θ. Clearly, the lower β, the less similar q_u from q_{max}. Diversification comes to the rescue, providing at least one diverse query q_{div} that is closer to the user's interests. However, even if a high diversity threshold is set (e.g., $\theta = 0.2$), the diverse queries are not sensibly closer to q_u, as they deviate too much from queries with high relevance. The closest query in the log, q_{log}, is always more similar to q_u than q_{div}.

In Figure 3b we focus on diversification by showing that the higher the number of diverse queries N, the higher the values of $sim(q_u, q_{div})$, hence, the higher the effectiveness of diversification. Finally, Figure 3c shows the capability of collaborative filtering to meet the user's desiderata even when the user query q_u is quite different from the maximal one. Indeed, the closest query in the log converges towards the user one after a few user feedbacks are collected. Already after one user feedback, the similarity between q_{log} and q_u sensibly rises, being very close to 1 when 4 feedbacks are collected.

6.2 Efficiency

We ran the tests on a machine equipped with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU and 16GB RAM, with the A-BI framework being implemented in Scala.

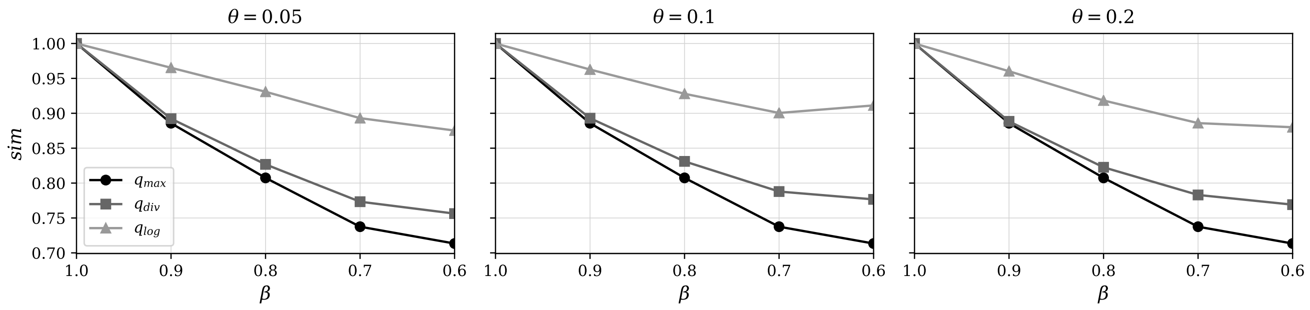
The efficiency results are depicted in Figure 4, with the overall execution time being in the order of 10^{-1} seconds at most. The execution of Algorithm 2 accounts for most computational time. Indeed, the process of diversification is computationally heavy, requiring to find, among the search space, the query with the highest diversity with reference to the current result set (with $O(|Q|)$ being the complexity of Line 4 in Algorithm 2 in case Q is not sorted). Note that the set of diverse queries is increasingly built, thus the *time to first result* is lower since, once a query is added to result set, it can be directly returned. We also varied the number of entries in the context ($|T| \in [5, 15]$), but since this does not significantly affect performances we do not show the results.

We finally emphasize that the execution time corresponds to the time necessary to define the recommended queries, and not to the time to actually execute them. Indeed, the execution of the query is demanded to and strictly depends on the performance of the enterprise data warehouse system.

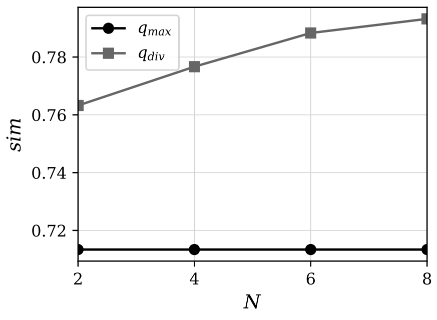
7 CONCLUSION

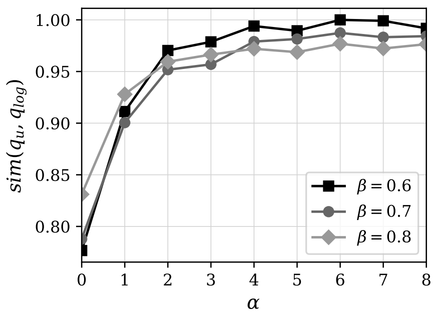
The A-BI framework is a first result in the direction of establishing a tight connection between analytical reporting and augmented reality applications. Besides proposing a reference functional architecture and interaction process, in this paper we have shown that recommendations can be given in real-time.

A-BI can be improved from many points of view, in particular we are working towards producing recommendations that are based on patterns of recognized context entities rather than on single entities (e.g., md-element e is relevant if the context includes d and d' but not d''). Furthermore, in its current implementation, a recommendation involves all the elements in the context while it would be useful to provide separate recommendations related to subsets of elements. Finally, it would be



(a) **Average similarity of q_{max}, q_{div}, and q_{log} with q_u for decreasing β and increasing θ ($|T| = 10$, $N = 4$, $\alpha = 1$)**





(b) Average similarity of q_{max} and q_{div} with q_u for increasing N ($\beta = 0.6$, $\theta = 0.1$, $|T| = 10$)

(c) Average similarity of q_u and q_{log} at increasing iterations for decreasing β ($\theta = 0.1$, $N = 4$, $|T| = 10$)

Figure 3: Effectiveness

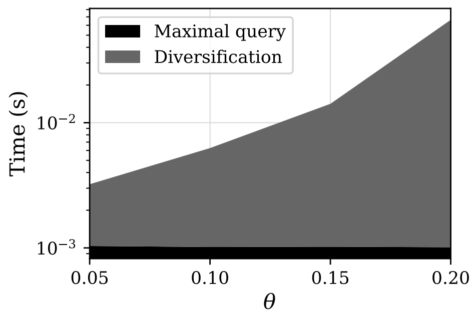


Figure 4: Efficiency for increasing values of θ

interesting to investigate on how to turn A-BI into a purely statistical framework where all weights are expressed in terms of probabilities and reasoning is probabilistic too.

REFERENCES

[1] Ronald Azuma. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.

[2] Angelo Croatti and Alessandro Ricci. Towards the web of augmented things. In *Proc. ICSA*, pages 80–87, Gothenburg, Sweden, 2017.

[3] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.

[4] Kostas Stefanidis, Evaggelia Pitoura, and Panos Vassiliadis. A context-aware preference database system. *Int. J. Pervasive Computing and Communications*, 3(4):439–460, 2007.

[5] Ibrahim A. Ibrahim, Abdullah M. Albarrak, and Xue Li. Constrained recommendations for query visualizations. *Knowl. Inf. Syst.*, 51(2):499–529, 2017.

[6] S. Dobson, M. Golfarelli, S. Graziani, and S. Rizzi. A reference architecture and model for sensor data warehousing. *IEEE Sensors Journal*, 18(18):7659–7670, 2018.

[7] Shuyao Qi, Dingming Wu, and Nikos Mamoulis. Location aware keyword query suggestion based on document proximity. *IEEE Trans. Knowl. Data Eng.*, 28(1):82–97, 2016.

[8] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. Snipsuggest: Context-aware autocompletion for SQL. *PVLDB*, 4(1):22–33, 2010.

[9] Raphaël Thollot, Nicolas Kuchmann-Beauger, and Marie-Aude Aufaure. Semantics and usage statistics for multi-dimensional query expansion. In *Proc. DASFAA*, pages 250–260, Busan, South Korea, 2012.

[10] Marie Le Guilly, Jean-Marc Petit, and Vasile-Marian Scuturici. SQL query completion for data exploration. *CoRR*, abs/1802.02872, 2018.

[11] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya G. Parameswaran, and Neoklis Polyzotis. SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13):2182–2193, 2015.

[12] Houssem Jerbi, Franck Ravat, Olivier Teste, and Gilles Zurfluh. Preference-based recommendations for OLAP analysis. In *Proc. DaWaK*, pages 467–478, Linz, Austria, 2009.

[13] Rafael Berlanga Llavori and Victoria Nebot. *Context-Aware Business Intelligence*, pages 87–110. 2015.

[14] Kaijian Xu, Manli Zhu, Daqing Zhang, and Tao Gu. Context-aware content filtering & presentation for pervasive & mobile information systems. In *Proc. ICST*, page 20, Quebec, Canada, 2008.

[15] Wenwei Xue, Hung Keng Pung, Paulito P. Palmes, and Tao Gu. Schema matching for context-aware computing. In *Proc. UbiComp*, pages 292–301, Seoul, Korea, 2008.

[16] Minmin Chen, Jian-Tao Sun, Xiaochuan Ni, and Yixin Chen. Improving context-aware query classification via adaptive self-training. In *Proce. CIKM*, pages 115–124, Glasgow, United Kingdom, 2011.

[17] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. A collaborative filtering approach for recommending OLAP sessions. *Decision Support Systems*, 69:20–30, 2015.

[18] Xiaohui Yan, Jiafeng Guo, and Xueqi Cheng. Context-aware query recommendation by learning high-order relation in query logs. In *Proc. CIKM*, pages 2073–2076, Glasgow, United Kingdom, 2011.

[19] Di Jiang, Kenneth Wai-Ting Leung, Jan Vosecky, and Wilfred Ng. Personalized query suggestion with diversity awareness. In *Proc. ICDE*, pages 400–411, Chicago, USA, 2014.

[20] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. AIDE: an active learning-based approach for interactive data exploration. *IEEE Trans. Knowl. Data Eng.*, 28(11):2842–2856, 2016.

[21] Patrick Marcel and Elsa Negre. A survey of query recommendation techniques for data warehouse exploration. In *Proc. EDA*, pages 119–134, Clermont-Ferrand, France, 2011.

[22] Arnaud Giacometti, Patrick Marcel, Elsa Negre, and Arnaud Soulet. Query recommendations for OLAP discovery-driven analysis. *IJDWM*, 7(2):1–25, 2011.

[23] Kaiping Zheng, Hongzhi Wang, Zhixin Qi, Jianzhong Li, and Hong Gao. A survey of query result diversification. *Knowl. Inf. Syst.*, 51(1):1–36, 2017.

[24] Elsa Negre, Franck Ravat, Olivier Teste, and Ronan Tournier. Cold-start recommender system problem within a multidimensional data warehouse. In *Proc. RCIS*, pages 1–8, 2013.

[25] Wolfgang Büschel, Annett Mitschick, and Raimund Dachselt. Here and now: Reality-based information retrieval: Perspective paper. In *Proc. CHIIR*, pages 171–180, New Brunswick, USA, 2018.

[26] Neven A. M. ElSayed, Bruce H. Thomas, Kim Marriott, Julia Piantadosi, and Ross T. Smith. Situated analytics: Demonstrating immersive analytical tools with augmented reality. *J. Vis. Lang. Comput.*, 36:13–23, 2016.

[27] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symposium on Visual Languages*, pages 336–343, Boulder, USA, 1996.

[28] Ashish Gupta, Venky Harinarayan, and Dallan Quass. Aggregate-query processing in data warehousing environments. In *Proc. VLDB*, pages 358–369, 1995.

[29] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Similarity measures for OLAP sessions. *Knowl. Inf. Syst.*, 39(2):463–489, 2014.

[30] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. Access path selection in a relational database management system. In *Proc. SIGMOD*, pages 23–34, Boston, USA, 1979.

[31] Matteo Golfarelli and Ettore Saltarelli. The workload you have, the workload you would like. In *Proc. DOLAP*, pages 79–85, New Orleans, USA, 2003.

[32] Stefano Rizzi and Ettore Saltarelli. View materialization vs. indexing: Balancing space constraints in data warehouse design. In *Proc. CAiSE*, pages 502–519, 2003.

[33] Jaime G. Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. SIGIR*, pages 335–336, Melbourne, Australia, 1998.