Learning Analysis Patterns using a Contextual Edit Distance

Clément Moreau, Veronika Peralta, Patrick Marcel, Alexandre Chanson, Thomas Devogele

Learning Analysis Patterns using a Contextual Edit Distance

Clement Moreau, Veronika Peralta, Patrick Marcel, Alexandre Chanson and Thomas Devogele
University of Tours
Blois, France
firstname.lastname@univ-tours.fr

ABSTRACT

This paper presents a proposal for learning users' behavior patterns when they interactively analyse data. Users' explorations (sequences of queries) are compared looking for subsequences of common actions or operations performed by the users during data analysis. We use a hierarchical clustering algorithm to retrieve groups of similar explorations. The main difficulty is to devise a similarity measure suitable to measure similarities between sequences of human actions. We propose to use a Contextual Edit Distance (CED), a generalization of Edit Distance that manages context-dependent edition costs. CED compares two users' explorations, making special emphasis in the similarity of queries with nearby queries in the exploration, which determines a local context. We test our approach on three workloads of real users' explorations, extracting common analysis patterns, both in explorations devised by students and expert analysts. We also experiment on an artificial workload, generated with CubeLoad [19], showing that our approach is able to identify the patterns imposed by the generator. To the best of our knowledge, this is the first attempt to characterize human analysis behavior in workloads of data explorations.

1 INTRODUCTION

Analyzing a database workload offers many practical interests, from the monitoring of database physical access structures [5] to the generation of user-tailored collaborative query recommendations for interactive exploration [10]. There has been much attention lately devoted to the analysis of user past activities to support Interactive Database Exploration (IDE) [12]. OLAP analysis is a particular case of IDE, that takes advantage of simple primitives like drill-down or slice-and-dice for the navigation of multidimensional data. These particularities enable the design of approaches for characterizing user explorations in how focused they are [8], in how contributive a query is to the exploration [7], or even in how to ensure that a sequence of analytical queries forms a coherent exploration [20]. Characterising user behavior while analysing data, i.e. learning the way users analyse data (the type and order of operations, the level of detail, the degree of focus) is a step forward in the understanding of analysis activities.

Identifying analysis behavior has several applications. The more natural one is a better support of IDE, for instance to understand users' information needs, to identify "struggling" during the exploration, or to provide better query recommendations. Notably, IDE systems usually do not offer such facilities. The prediction of next analysis steps is particularly interesting, enabling beforehand execution of probable queries and caching of results, as well as advanced optimization strategies. Another benefit is the design of more realistic workloads for database benchmarking. Classical benchmarks like TPC-H or TPC-DS poorly include interactive exploration activities in their synthetic workloads, and are not appropriate to evaluate modern IDE systems [9]. Identifying analysis patterns would allow to better model user's explorations and mimic such activities in benchmark workloads. Finally, we mention the detection of clandestine intentions [1] as another potential benefit. Indeed, as reported by [1], query sequences may reflect such intentions, where users prefer to obtain information by means of sequences of smaller, less conspicuous queries to avoid direct queries which may disclose their true interests. The identification of typical analysis patterns may help distinguishing normal from clandestine intentions.

In this paper we deal with the identification of analysis patterns in a log of explorations devised by real users. We consider that an exploration is a coherent sequence of queries over a database schema, done by a user with the goal of fulfilling an information need. In [19], Rizzi and Gallinucci described 4 recurrent types of user analyses and propose a tool for generating realistic explorations based on these usage types. Our goal is to go a step forward and learn more analysis patterns from the explorations of real users. Concretely, we aim to **cluster together explorations showing similar analysis patterns**. The idea behind analysis patterns is to look for sequences of common actions or operations performed together when analysing data, as some kind of movements in a data space. From this point of view, OLAP operations (e.g. drilling down, adding a filter, changing a measure) are first class citizens, while the actual analyzed data is less important. For example, we can retain that a user performed a sequence of drills down, disregarding the dimension that was drilled down or the semantics of the underlying data. Explorations can be compared in such terms, i.e. to what extent they share the same sequences of operations and evolve at the same level of aggregation and filtering.

Many distances proposed to compare sequences, for example the Damerau-Levenshtein distance [6] or the Smith-Watermann algorithm [21], part of Edit Distance family, count the minimum number of operations (modification, addition, deletion) required to transform one sequence into the other. They are particularly adapted for sequences of independent symbols, as DNA sequences or strings. However, when symbols represent human behavior, including homogeneous, interconnected and repetitive actions, an appropriate distance should satisfy other requirements. In particular, we want the following requirements:

- (R_1) edition cost depends on the similarity of nearby symbols.
- (R_2) edition of repeated close symbols has little cost.
- (R_3) permutation of close symbols has little cost.

Indeed, while edition cost is constant in classical Edit Distance, for comparing interconnected actions, it should take context (i.e. the nearby actions) into account. For example, removing a measure should be costly within a focused sequence of drills down and filters, while it should be cheaper inside a sequence with other changes in measures. In addition, sequences of filters should be similar, no matter how many filters there are. Furthermore, permuting operations should have little impact, e.g. filtering and then drilling vs. drilling then filtering.

Previous attempts made for measuring the similarity of sequences of OLAP queries (like e.g. [3], that extends the Smith-Watermann algorithmm) were not designed to satisfy the stated requirements. We propose **a Contextual Edit Distance (CED)** specially designed to satisfy them.

Our contributions, sketched in Figure 1 include: (i) a representation of queries and explorations in the space of OLAP operations, including a similarity function among OLAP operations in such space (described in Section 3), (ii) a CED for comparing explorations considering context (Section 4), (iii) a proposal for clustering explorations, based on CED (Section 5), and (iv) a set of experiments showing that CED outperforms state of the art distances allowing the learning of analysis behavior in varied logs of explorations (Section 6).

2 RELATED WORK

The recurrent types of user analyses described by Rizzi and Gallinucci [19] are the first attempt to define analysis patterns in OLAP workloads. Authors claim that obtaining real OLAP workloads by monitoring the queries actually issued in companies and organizations is hard, and propose a parametric generator of OLAP workloads, CubeLoad, based on a four templates that model recurrent types of user analyses:

- *Slice And Drill.* Following the default behavior of several OLAP front-ends, hierarchies are progressively navigated by choosing a member of a current group-by level, creating a selection predicate on such member and drilling down on it. Therefore, explorations of this template contain sequences of filter and drill-down operations.

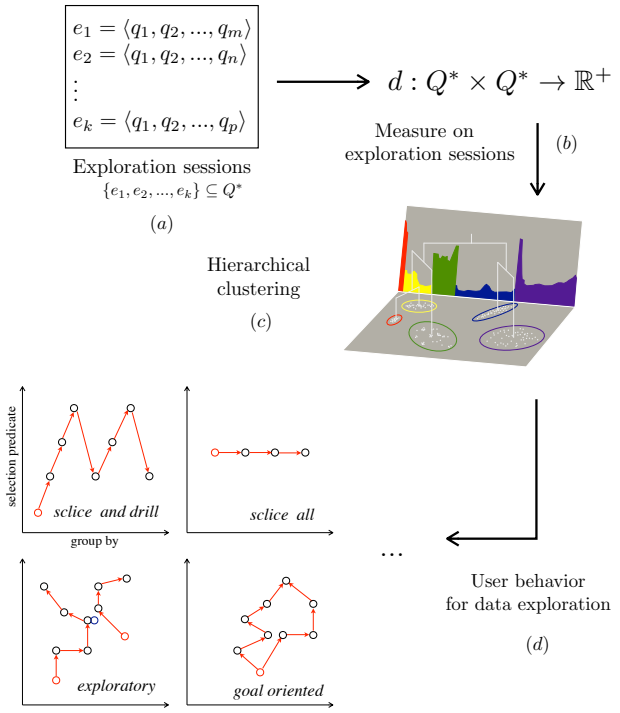

Figure 1: Overview of the approach: (a) explorations and queries are represented in a space of OLAP operations, (b) computation of CED, (c) clustering of explorations based on CED, (d) extraction of patterns of user behavior (here, represented as the patterns of Cubeload).

- *Slice All.* Users are sometimes interested in navigating a cube by slices, i.e., repeatedly running the same query but with different selection predicates. Then, this templates generates sequences of unfilter/filter operations.
- *Exploratory.* The motivation for this template is the assumption that several users, while exploring the cube in search of significant correlations, will be "attracted" by one surprising query and then evolve casually. So, explorations based on this template contain varied random operations.
- *Goal Oriented.* Explorations of this type are run by users who have a specific analysis goal, but whose OLAP skills are limited so they may follow a complex path to reach their destination. Explorations of this template contain varied operations but converging to some specific point.

Other works analyse real workloads and observe tendences or patterns. Aligon et al. [2] analyse a workload of explorations devised by master students and observe some general tendence. They report the number of OLAP operations used between two consecutive queries, the level of detail and the number of filters of queries in the explorations, as well as the queries indicated as relevant by the students. As general behavior, they highlight that explorations are more focused at the end (i.e. the number of filters and level of detail increase along the exploration, while the number of OLAP operations between consecutive queries varies before a final drop at the end of the exploration) and contain more relevant queries at the end. The term focus is used as in [8]: "When focused, an analyst would expect more precise queries, related to what she is currently analyzing. On the contrary, when exploring the data, the analyst would prefer more diverse queries, for a better data space coverage." Djedaini et al. [7] analyse another workload of explorations of master students and propose a model for learning the quality of an exploration (and the skill of the user) based on a set of query features.

To our knowledge, our work is the first devoted to discover exploration patterns in OLAP workloads.

Many recent works aimed at comparing queries and sessions. We mention as good recent surveys [3] for OLAP queries and explorations, and [4] for SQL queries.

Aligon et al. [3] also proposes two similarity measures: one tailored for OLAP queries and another tailored for OLAP sessions. Their measures were designed for satisfying other requirements, in particular capturing the portions of the cube that are more explored for improving query recommendation. Consequently, similarity measures are based on common query parts (e.g. filters and measures) more than common operations, and are strongly dependent on cube schema. To our knowledge, no similarity already proposed in the literature for comparing explorations includes the requirements presented in the previous section.

3 EXPLORATION MODEL

This section introduces the description of queries and explorations used all along the paper as well as their representation in a space of OLAP operations.

3.1 Queries and explorations

In order to keep the formalism simple, we only take into account cubes under a ROLAP perspective, described by a star schema [14]. For convenience, we consider that a dimension consists of a unique hierarchy without branches, i.e., consisting of chains of levels. In this paper, we focus on multidimensional queries

modeled as a collection of fragments extracted from the query expression, as in [3].

Definition 3.1 (OLAP query). An *OLAP query* over cube schema S is a triple $q = \langle G, P, M \rangle$ where:

(1) $G = \{g_1, \ldots, g_j\}$ is the query group-by set, each g_i being a level of a hierarchy of the cube;
(2) $P = \{p_1, \ldots, p_k\}$ is a set of Boolean predicates, of the form $l = v$, with l a level and v a value. Compound predicates are build as the disjunction of predicates on a same level (l), at most one for each hierarchy, whose conjunction defines the *selection predicate* for q;
(3) $M = \{m_1, \ldots, m_l\}$ is the measure set whose values are returned by q.

We intentionally remain independent of presentation and optimization aspects, specially the order in which attributes are projected (and visualized), the order of joins, etc.

Finally, an *exploration* is a coherent sequence of queries over a cube schema, devised by a user with the goal of fulfilling an information need that may not be well defined initially.

Definition 3.2 (Exploration). Let S be a cube schema. An exploration $e = \langle q_1, \ldots, q_p \rangle$ is a sequence of queries over S. We note $q \in e$ if q appears in the exploration e, and $exploration(q)$ to refer to the explorations where q appears.

3.2 Query features

For each query, we extract a set of simple features computed from the query text and its relationship with previous query in an exploration. The set of features is inspired from our previous works [7, 8]. It intends to capture various aspects of OLAP navigation, in particular the set of OLAP operations that express one query w.r.t. the previous one (e.g. a query being a drill down of the previous one), the level of aggregation (i.e. how deep the query is in the aggregation lattice) and the level of filtering (i.e. how filtered is the data space). Table 1 presents an overview of the features, where added (resp., deleted) indicates the modification made compared to the previous query.

For the following definitions let $q_k = \langle G_k, P_k, M_k \rangle$ be the query occurring at position k in the exploration e over cube schema S. All the queries we considered are supposed to be well formed, and so we do not deal with query errors. Features are computed comparing the query q_k to the previous query in the exploration e, $q_{k-1} = \langle G_{k-1}, P_{k-1}, M_{k-1} \rangle$. For the first query of e, i.e. q_1, we consider as predecessor the "empty" query $q_0 = \langle \emptyset, \emptyset, \emptyset \rangle$. All the following features are defined for $k \geq 1$.

Number of Added Levels. $NAL(q_k, q_{k-1})$ counts the number of levels in the group by set of q_k that were not part of the group by set of q_{k-1}.

$$NAL(q_k, q_{k-1}) = |G_k - G_{k-1}| \tag{1}$$

Number of Deleted Levels. $NDL(q_k, q_{k-1})$ counts the number of levels in the group by set of q_{k-1} that are not longer used in q_k.

$$NDL(q_k, q_{k-1}) = |G_{k-1} - G_k| \tag{2}$$

Number of Added Filters. $NAF(q_k, q_{k-1})$ counts the number of filters of q_k that were not filters of q_{k-1}.

$$NAF(q_k, q_{k-1}) = |P_k - P_{k-1}| \tag{3}$$

Feature	Description
NAL	Number of added levels
NDL	Number of deleted levels
NAF	Number of added filters
NDF	Number of deleted filters
NAM	Number of added measures
NDM	Number of deleted measures
Adepth	Aggregation depth
Fdepth	Filter depth

Table 1: Query features

Number of Deleted Filters. $NDF(q_k, q_{k-1})$ counts the number of filters of q_{k-1} that are not longer used in q_k.

$$NDF(q_k, q_{k-1}) = |P_{k-1} - P_k| \tag{4}$$

Number of Added Measures. $NAM(q_k, q_{k-1})$ counts the number of measures of q_k that were not measures of q_{k-1}.

$$NAM(q_k, q_{k-1}) = |M_k - M_{k-1}| \tag{5}$$

Number of Deleted Measures. $NDM(q_k, q_{k-1})$ counts the number of measures of q_{k-1} that are not longer used in q_k.

$$NDM(q_k, q_{k-1}) = |M_{k-1} - M_k| \tag{6}$$

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

$$Adepth(q_k) = \sum_{g_i \in G_k} depth(g_i) \tag{7}$$

Filter Depth. $Fdepth(q_k)$ measures the number of filters appearing in q_k.

$$Fdepth(q_k) = |P_k| \tag{8}$$

In what follows, we represent an OLAP query in the space of query features, i.e. as a 8-dimensional vector, each position corresponding to one of the features described above. This representation is at the core of our proposal for computing the similarity between queries and then between explorations. It focuses in operations between queries and is independent of the underlying data cube, i.e. a given sequence of operations, even on different data cubes, will result in the same sequence of query vectors.

Example 1. *Consider an exploration e_1 composed of 4 queries:*
$q_1 = \langle \{year\}, \emptyset, \{qty\} \rangle$ – sales quantity per year;
$q_2 = \langle \{year\}, \{year = \text{“2019”}\}, \{qty\} \rangle$ – adds a filter;
$q_3 = \langle \{year, country\}, \emptyset, \{qty\} \rangle$ – unfilter, drill-down;
$q_4 = \langle \{year, city\}, \emptyset, \{qty, amount\} \rangle$ – drill-down, adds measure;
Vector for q_1, $\langle 1, 0, 0, 0, 1, 0, 1, 0 \rangle$, indicates an added level (year) and an added measure (qty) w.r.t. the empty query; last positions correspond to aggregation and filter depths. Vectors for q_2, q_3 and q_4 indicate the differences w.r.t. previous queries and the changes in aggregation and filter depths: $\langle 0, 0, 1, 0, 0, 0, 1, 1 \rangle$, $\langle 1, 0, 0, 1, 0, 0, 2, 0 \rangle$, $\langle 1, 0, 0, 0, 1, 0, 3, 0 \rangle$, resp.

Finally, we use cosine similarity for computing similarity between queries. This measure is adapted to compute similarity of two vectors and is normalized in $[0, 1]$. In this way, it privileges the nature of OLAP operations and not their number.

4 CONTEXTUAL EDIT DISTANCE

This section describes our proposal of Contextual Edit Distance: definition and implementation issues.

CED is a generalization of the Edit Distance that incorporates the following requirements:

(1) *Context-dependent cost*: Edition cost depends on the similarity of nearby queries. The more similar and closer the queries, the lower the cost of operation.
(2) *Repetition*: Edition of repeated close queries has low cost.
(3) *Permutation*: Similar and close queries can be exchanged with a low cost.

Example 2. *Consider an exploration reflecting an exploratory behavior at the beginning (many changes in measures and group by set) and more focus at the end (drilling and filtering). We can sketch it as follows (where L, F and M means level, filter and measure, + means addition and - means deletion; we skip aggregation and filter depth for simplicity):*
⟨ *+D+M, +M, +M, +D, +M-M, -M+D, -D+D, +F+D, +F+D, +F, +F* ⟩.

Consider the insertion of a query adding an additional measure (+M). The edition cost should be low if the query is inserted at the beginning (as it is similar to near queries), even lower at positions 2 to 4 (because repeating the same operations), but high at the end.

This requirements ensure that explorations reflecting a given pattern (e.g. sequences of drill-downs) are judged to be very similar no matter the exploration length (i.e. how many drill-downs) nor the underlying data (which data was drilled-down).

We remark that although this paper deals with explorations, CED definition and properties are independent on the type of explorations, the type of queries and even the nature of data. Indeed, CED can be adapted to any type of sequence on an alphabet of symbols provided that exists a similarity metric among them.

4.1 Definition of Contextual Edit Distance

The main contribution of CED is the modification of the cost function γ which generalizes the classical definition of Edit Distance and takes into account the local context of each query in the exploration. Intuitively, the cost of an edit operation will be lower if the edited query (e.g. a query to be added to an exploration) is similar to nearby queries (e.g. the queries that are near the position where the query is added). This notion of local context is modeled as a context vector, which controls the zone of influence of the context. This subsection presents these concepts and a formal definition of CED.

Let Q be the set of all possible queries over a cube schema S and Q^* the set of all possible explorations on Q. For the following definitions, let $e = \langle q_1, ..., q_n \rangle$ be an exploration, q_k be a query in e, $1 \leq k \leq n$, and q_x be a new query to be edited in e. These notations are partially adapted from the formal language theory community and particularly from [22].

CED extends the set of edit operations of Edit Distance (usually modification, addition, deletion) to take context into account.

Definition 4.1 (Contextual edit operation). A contextual edit operation o is a function $o : (Q^* \times Q \cup \{\varepsilon\} \times \mathbb{N}) \to Q^*$ whose arguments are an exploration, a new query to be included in the exploration (or none) and the index (position) in the exploration where the edition takes place. We consider the following set $O = \{\texttt{mod}, \texttt{add}, \texttt{del}\}$ of edit operations:

- $\texttt{mod} : (e, q_x, k) \mapsto q_1, ..., q_{k-1}, q_x, q_{k+1}, ..., q_n$
  Replace the query at index k by the query q_x.
- $\texttt{add} : (e, q_x, k) \mapsto q_1, ..., q_{k-1}, q_x, q_k, ..., q_n$
  Insert query q_x at index k. The queries at and after index k are shifted forward.
- $\texttt{del} : (e, \varepsilon, k) \mapsto q_1, ..., q_{k-1}, q_{k+1}, ..., q_n$
  Delete the query at position k. The queries after position k are shifted backward.

Given an operation $o(e, q_x, k)$, a *context vector* is a numeric vector that indicates the level of influence of nearby queries for this operation, being stronger near index k and softening farther. We use the context vector for weighting the similarity between queries. Intuitively, the context vector quantifies the relationship between a query q_x and another query q_i. Thus, the greater v_i, the greater the impact of query q_i on q_x. A seed function is used to generate context vectors.

Definition 4.2 (Context function and context vector). Consider a contextual edit operation $o(e, q_x, k)$ and a seed function $f_k : \mathbb{N} \to [0, 1]$ which holds the following properties:

(1) $f_k(k) = 1$.
(2) f_k is a monotonically increasing function on $]-\infty, k]$.
(3) f_k is symmetrically centered on k.

The third property guarantees to take with the same importance previous and future queries located at equal distance from q_k.

A context function $\varphi_o : \mathbb{N}^* \to [0, 1]$ is a transformation of f_k stretching the function according to the type of operation o. We distinguish three cases for o in $\{\texttt{mod}, \texttt{add}, \texttt{del}\}$:

- $\varphi_{\texttt{mod}}(x) = f_k(x)$
- $\varphi_{\texttt{add}}(x) = \begin{cases} f_k(x+1) & \text{if } x \leq k-1 \\ f_k(x) & \text{if } x \geq k \end{cases}$
- $\varphi_{\texttt{del}}(x) = \begin{cases} f_k(x+1) & \text{if } x \leq k-1 \\ 0 & \text{if } x = k \\ f_k(x-1) & \text{if } x \geq k+1 \end{cases}$

Finally, a context vector $v : O \to [0, 1]^n$ is defined as

$$v(o) = \langle v_1, \ldots, v_n \rangle$$

where $v_i = \varphi_o(i)$.

About add and del contextual vector functions:

- For an insertion $\texttt{add} : (e, q_x, k)$, query q_{k-1} and q_k are fully taken into account for the addition of q_x in e because we insert q_x between index $k-1$ and k i.e. $\varphi_{\texttt{add}}(k-1) = 1$ and $\varphi_{\texttt{add}}(k) = 1$.
- For a deletion $\texttt{del} : (e, \varepsilon, k)$, the absence of the query q_k in e is quantified in relation to the remaining of e i.e. $\varphi_{\texttt{del}}(k) = 0$.

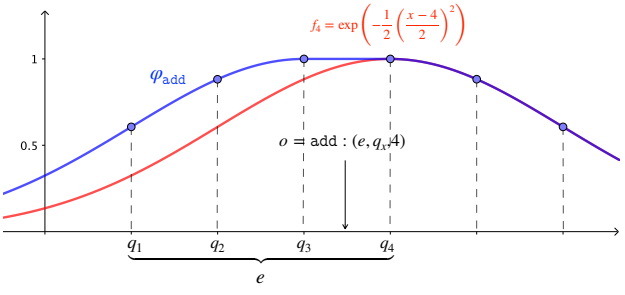

Figure 2: Add a new query q_x in position 4 in e.

Example 3. *Consider the operation $o = \mathsf{add}(e, q_x, 4)$, adding a query at index 4 of an exploration. Figure 2 illustrates the computation of φ_{add} (plotted in blue) from a given seed function (in red). The corresponding context vector is $v(o) = \langle 0.61, 0.88, 1, 1\rangle$. It indicates that the similarity score of queries at indexes 3 and 4 is fully considered (weight of 1) while a lower score is considered for query at index 1.*

The cost function γ of CED generalizes the classical definition of Edit Distance and takes into account the local context of each query in the exploration.

Definition 4.3 (Cost function γ). Given an operation $o(e, q_x, k)$, a cost function $\gamma : O \to [0, 1]$ for the contextual edit operations is defined as:

$$\gamma(o) = \alpha \times \delta(o) + (1-\alpha)\left(1 - \max_{i \in [\![1,n]\!]} \{sim(q_i, q_x) \times v_i(o)\}\right) \quad (9)$$

where:

- $\alpha \in [0, 1]$ is a contextual parameter.
  If $\alpha \to 0$ the contextual part is maximal and therefore the distance between two queries will be strongly evaluated according to the content of the exploration being edited ; if $\alpha \to 1$ then cost of edition is fixed.
- $\delta(o) = \begin{cases} 1 - sim(q_k, q_x) & if\ o = \mathsf{mod} \\ 1 & else \end{cases}$
  is the local cost of the Edit Distance.
- $sim : Q \times Q \to [0, 1]$ is a similarity measure between two queries, computed as the cosine of query vectors.

Example 4. *Consider the exploration e_1 of Example 1 and consider the insertion of query q_x, with vector $\langle 0, 0, 1, 0, 0, 0, 2, 1\rangle$ (addition of a filter), at position 4, as shown in Figure 2, with $\alpha = 0.1$. Then, the cost $\gamma(o)$ is such that:*

$$\gamma(o) = 0.1 + 0.9\left(1 - \max\begin{Bmatrix} \cos(q_1, q_x) \times \varphi_{\mathsf{add}}(1) \\ \cos(q_2, q_x) \times \varphi_{\mathsf{add}}(2) \\ \cos(q_3, q_x) \times \varphi_{\mathsf{add}}(3) \\ \cos(q_4, q_x) \times \varphi_{\mathsf{add}}(4) \end{Bmatrix}\right)$$
$$= 0.1 + 0.9\left(1 - \max\begin{Bmatrix} 0.47 \times 0.61 \\ 0.94 \times 0.88 \\ 0.66 \times 1 \\ 0.74 \times 1 \end{Bmatrix}\right)$$
$$= 0.1 + 0.9 \times (1 - 0.83) = 0.25$$

The insertion of q_x at position 2 (i.e. closer to other filter operation) has cost 0.15.

Definition 4.4 (Edit path). Given two explorations $e, e' \in Q^*$, an edit path $P = \langle o_1, o_2, ..., o_m\rangle$ from e to e' is a sequence of operations that transform e in e'. We note $\mathcal{P}(e, e')$ the set of all edit paths to transform e in e'.

An important point to be mentioned here is that, for efficiency reasons, the context is static, i.e. it only considers the original queries in exploration e. This means that the contextual edit operation o_i has no impact on the cost of operation o_{i+1}. If it was otherwise, i.e., dealing with dynamic context, the edition problem would have been NP-hard by reduction to Job shop scheduling problem [13]. Also note that as the add operator is not the reverse of del operator, the edition is asymmetric. Thus, by re-using the definition of the Edit Distance in [22], we can define the one-side distance from an exploration e_1 to an exploration e_2.

Definition 4.5 (One-sided Contextual Edit Distance). Let $\tilde{d}_{CED} : Q^* \times Q^* \to \mathbb{R}^+$ be the Contextual Edit Distance from e_1 to e_2 such that:

$$\tilde{d}_{CED}(e_1, e_2) = \min_{P \in \mathcal{P}(e_1, e_2)} \left\{ \sum_{i=1}^{|P|} \gamma(o_i) \right\} \quad (10)$$

In this form, CED would be very similar to Hausdorff distance [11], but would remain asymmetric. This is why we use the same trick as Hausdorff distance and we apply the max operator on each one-sided contextual edit distance to recover the symmetry.

Definition 4.6 (Contextual Edit Distance). Let $d_{CED} : Q^* \times Q^* \to \mathbb{R}^+$ be the Contextual Edit Distance such that:

$$d_{CED}(e_1, e_2) = \max\left\{\tilde{d}_{CED}(e_1, e_2), \tilde{d}_{CED}(e_2, e_1)\right\} \quad (11)$$

4.2 Implementation of CED

We can compute d_{CED} using a Dynamic Programming approach like the classical Edit Distance [22]. This solution has a polynomial complexity in $O(n \times p)$ and can be adapted easily for the computation of CED.

Algorithm 1 computes the context vector and the cost function (as in Equation 9). Note that $\varphi_o : \mathbb{N}^* \to [0, 1]$ and $\alpha \in [0, 1]$ are fixed parameters and operator $\cdot$ represents vector concatenation. Algorithm 2 computes the one-sided Contextual Edit Distance (Equation 10), and Algorithm 3 recovers the symmetry.

Algorithm 1: Cost function γ

Data: Contextual edit operator $o : (e, q_x, k)$.
Result: Cost $\gamma(o)$ of the operation o.
$v(o) \leftarrow \langle\rangle$
for $i \in [\![1, |e|]\!]$ **do**
  $v(o) \leftarrow v(o) \cdot \langle\varphi_o(i)\rangle$
$v_{sim} \leftarrow \langle\rangle$
for $i \in [\![1, |e|]\!]$ **do**
  $v_{sim} \leftarrow v_{sim} \cdot \langle sim(q_i, q_x) \times v_i(o)\rangle$
if $e = \mathsf{mod}$ **then**
  $\delta(o) \leftarrow 1 - sim(q_k, q_x)$
else
  $\delta(o) \leftarrow 1$
return $\alpha \times \delta(o) + (1 - \alpha) \times (1 - \max(v_{sim}))$

Next, we prove that the computation of CED is polynomial in time:

Theorem 4.7. *CED is in $O(n \times p \times \max(n, p))$ where $|e_1| = n$ and $|e_2| = p$.*

Proof : Let's note $T(A_i)$ the big O time complexity of the algorithm A_i, with $i \in \{1, 2, 3\}$.
So we have :

- $T(A_1(o)) = |e|$ where e is the edited exploration.
- $T(A_2(e_1, e_2)) = n \times p \times T(A_1(o : e_1, q_x, k))$
  $= n^2 \times p$
- $T(A_2(e_2, e_1)) = n \times p \times T(A_1(o : e_2, q_x, k))$
  $= n \times p^2$
- $T(A_3(e_1, e_2)) = T(A_2(e_1, e_2)) + T(A_2(e_2, e_1))$
  $= n^2 p + p^2 n$
  $\in O(n \times p \times \max(n, p))$

As a result, the Algorithm 3 has a time complexity in $O(n \times p \times \max(n, p))$.

□

Algorithm 2: One-sided Contextual Edit Distance $\tilde{d}_{CED}$

Data: Exploration couple (e_1, e_2).
Result: One-sided Context Edit Distance $\tilde{d}_{CED}(e_1, e_2)$
$D \leftarrow [0...|e_1|][0...|e_2|]$
for $i \in [\![0, |e_1|]\!]$ **do**
  for $j \in [\![0, |e_2|]\!]$ **do**
    if $i = 0 \vee j = 0$ **then**
      $D[i, j] \leftarrow i + j$
    else
      $o_{\texttt{mod}} \leftarrow \texttt{mod} : (e_1, q^{(2)}_{j-1}, i - 1)$
      $o_{\texttt{del}} \leftarrow \texttt{del} : (e_1, q^{(1)}_{i-1}, i - 1)$
      $o_{\texttt{add}} \leftarrow \texttt{add} : (e_1, q^{(2)}_{j-1}, i - 1)$
      $D[i, j] \leftarrow \min\{$
        $D[i - 1, j - 1] + \gamma(o_{\texttt{mod}})$,
        $D[i - 1, j] + \gamma(o_{\texttt{del}})$,
        $D[i, j - 1] + \gamma(o_{\texttt{add}})$
        $\}$

Algorithm 3: Contextual Edit Distance d_{CED}

Data: Exploration couple (e_1, e_2).
Result: Context Edit Distance $d_{CED}(e_1, e_2)$.
return $\max\left\{\tilde{d}_{CED}(e_1, e_2), \tilde{d}_{CED}(e_2, e_1)\right\}$

5 CLUSTERING OF EXPLORATIONS

Our objective is to cluster together explorations showing the same user behavior. To this end, we pair CED to an off-the-shell clustering algorithm, and we test it against several workloads concerning users with varied analytical skills and different UI, aiming to discover different types of patterns. In addition, as some datasets come with a ground truth, they allow for the quantification of clustering quality and the comparison to state of the art distances. The following subsections describe the workload used in experiments, the experimental protocol and implementation details.

5.1 Workloads

In our experiments, we reuse several workloads described in the literature [2, 7, 15] consisting of navigation traces of real users on real data. We chose to test our proposal in several workloads to avoid learning specific behavior of a set of users. The 3 workloads concern users with different analysis skills (students, experts), using different analysis tools (an OLAP tool, a research prototype and an advanced IDE interface) and accessing data cubes of different sizes and complexities. We are not aware of other public analytical workloads, specially from senior analysts, whose analysis activity is jealously guarded by companies [19]. We also generated artificial explorations on artificial data using a state-of-the-art workload generator [19].

*Real explorations on **ipums data**.* The first workload, henceforth dubed *Ipums*, consists of navigation traces of OLAP users collected during the testing phase of the development of Falseto [2], a tool meant to assist query and exploration composition, by letting the user summarize, browse, query, and reuse former analytical explorations. The 17 OLAP users engaged in the test were students of two Master's programs specialized in Business Intelligence. The test was not part of the programs, was not graded and all the participants were volunteers. They developed explorations for answering four analytical questions on the IPUMS cube. The IPUMS cube integrates data from the IPUMS (Integrated Public Use Microdata Series) website [1]. It is organized as a star schema with 5 dimensions, 12 (non-top) levels, 25 measures, and contains 500,000 facts recorded in the fact table. From this experiment, we reuse 27 explorations and 306 queries, with an average of 11 queries per exploration.

During a preliminary analysis of the workload, Aligon et al. labelled explorations distinguishing five analysis styles:

- *FOCUS*. The exploration is more focused as time passes,
- *OSCILLATE-FOCUS*. The exploration is more exploratory (the levels of detail and filtering oscillate) at the beginning but is more focused at the end,
- *OSCILLATE*. The exploration is always exploratory,
- *FIX*. The exploration keeps constant levels of detail and filtering,
- *ATYPICAL*. The exploration has atypical or erratic behavior.

*Real explorations on **open data**.* The second workload, henceforth dubed *Open*, consists of navigation traces collected in the context of a French project on energy vulnerability. These traces were produced by 8 volunteer students of a Master's degree in Business Intelligence, answering fuzzy information needs defined by their lecturer, to develop explorative OLAP navigations using Saiku[2] over three cubes instances [7]. The main cube is organized as a star schema with 19 dimensions, 68 (non-top) levels, 24 measures, and contains 37,149 facts recorded in the fact table. The other cubes are organized in a similar way. From this experiment, we reuse 28 explorations and 941 queries, with an average of 34 queries per exploration. A particularity of some third party OLAP tools, like Saiku, is that their user interface submits a new query for each user action (including intermediate drag-and-drops), resulting in very long explorations in the log. Nevertheless, there were some extremely short explorations (6 explorations counting less than 10 queries), which mainly correspond to incomplete studies.

*Real explorations on **cyber security data**.* The third workload, henceforth dubed *Security*, consists of analysis sessions made by real analysts in the context of the "Honeynet Project" [15]. 56 analysts specialized in the domain of cyber-security were recruited (via dedicated forums, network security firms, and volunteer senior students from the Israeli National Cyber-Security Program) and asked them to analyze 4 different datasets using a prototype web-based analysis platform. Each dataset contains between 350 to 13K rows of raw network logs that may reveal a distinct security event, e.g. malware communication hidden in network traffic, hacking activity inside a local network, an IP range/port scan, etc. (there is no connection between the tuples of different datasets). The analysts were asked to perform as many analysis actions as required to reveal the details of the underlying security event of each dataset. They used a web-based analysis platform developed for the project [15].

Even if there is no ground truth for this workload, it is interesting because queries were devised by expert analysts.

***Artificial** explorations.* The last workload, with artificial data, comes from the Star Schema Benchmark [17], and was used with

[1] Minnesota Population Center. Integrated Public Use Microdata Series. http://www.ipums.org, 2008.
[2] http://meteorite.bi/products/saiku

artificial explorations. The Star Schema Benchmark (SSB) is a variation of TPC-H, a popular benchmark from the Transaction Processing Performance Council (TPC). SSB cube consists of a relational database under the form of a star schema, with one fact table and 4 dimension tables.

Instead of using the rather limited SSB workload, we generated artificial explorations using CubeLoad [19].

5.2 Protocol

In order to cluster explorations, we execute an off-the-shelf clustering algorithm using CED as distance function. For comparison, we execute the same clustering algorithm with two alternative distances: (i) the classical Edit Distance (henceforth dubed ED) as a baseline, and (ii) Aligon et al.'s distance [3] (henceforth dubed AD), a state of the art metric for session similarity. We analyze the obtained clusters under several angles:

Firstly, when we have some knowledge qualifying explorations, even if it is not exactly a ground truth, we compare our results to such knowledge. For the artificial explorations we compare the obtained clusters with the Cubeload templates used for the generation of the workload. This experiment aims to show that our approach is able to cluster together all the explorations corresponding to a given template. For the ipums workload we compare to the preliminary labels assigned by Aligon et al. Actually, such labels are not a ground truth, as there were not produced with the goal of clustering explorations, but they may provide a nice idea of the quality of the exploration. We report Adjusted Rand Index (ARI) and V-measure (harmonic mean of clusters homogeneity and completeness) scores[3], and we compare our clustering scores to those obtained with ED and AD distances.

Second, for the four workloads, we report further scores concerning intrinsic cluster quality. Indeed, too few clusters will mix different behaviors, too many clusters will overfit user behavior. We aim to balance: number of clusters, cluster diameter (the distance between the farthest objects in the cluster) and mean Silhouette Coefficient (a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation)). Silhouette scores[3] are merely informative in out tests, as the metric is more adapted to hyper-spherical clusters.

Finally, we study the medoids of each cluster (the exploration that is the most similar to all other explorations in the cluster) and we manually observe the OLAP operations of the medoid for providing an explanation of the concerned behavior pattern.

5.3 Implementation and setting

Our methods are implemented in Python, using Scipy, Sklearn [18] and Matplotlib libraries. Code and data are available from Github [4]. CED parameters were tuned and set as follows:

- Cosine similarity is used to compare two queries in an exploration.
- α parameter is set to 0 to give fully priority to context.
- We use the following seed function.

$$f_k(x) = \exp\left(-\frac{1}{2}\left(\frac{2\sqrt{k+1}(x-k)}{|e|}\right)^2\right)$$

It is a Gaussian function centered at k, therefore satisfying the properties announced in Definition 4.2. Furthermore, it is based on a Gaussian function with a standard deviation coefficient equal to $\frac{2\sqrt{k+1}}{|e|}$. This coefficient is interesting because, as it depends on k, it allows to vary context size along the exploration. In particular, when k is small (at the beginning of the exploration, when user intents are less defined and behavior is more exploratory), the standard deviation is high (i.e. the curve of f_k is flattened) which allows to include in local context, some queries that are far from the index k. On the other hand, when $k \to |e|$ (at the end of the exploration, when behavior is more focused), the curve of f_k narrows around k reducing context size. Figure 3 illustrates the context function for several values of k.

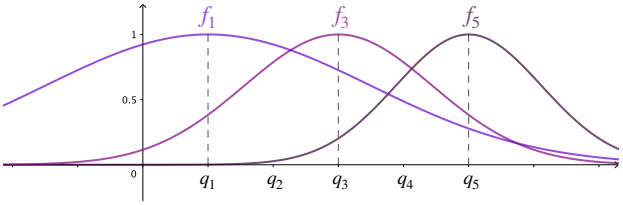

Figure 3: Example of context function with $k = 1, 3, 5$ and $|e| = 5$

As we do not know, a priori, the form of clusters, nor their density, we use a hierarchical clustering algorithm, which provides more flexibility than hyper-spherical and density-based algorithms. In addition, it outputs a dendrogram that allows to parameter the setting of number of clusters and eases the visual analysis of clusters. In order to find a good balance, we experimentally combine some criteria to cut the dendrogram: relative loss of inertia, cluster diameter and minimum number of clusters. We use the *Ward* method as agglomeration criteria.

Finally, a correlation study among query features revealed that Fdepth was highly correlated with Adepth in all workloads. In consequence, we excluded Fdepth from query vector.

6 EXPERIMENTS

In this section we report the results of the experiments conducted to validate our proposal.

6.1 Comparison against ground truth

In this experiment on Artificial and Ipums workloads, we compared the clusters obtained with our method to the available ground truth (i.e. the template used to generate each exploration and the preliminary classification of Ipums).

Figure 4 (a and d) shows the obtained dendrograms. Explorations (identified by numbers) are arranged in the horizontal axis, plotting similar explorations close (according to CED). Links indicate which explorations are clustered together, shorter links meaning more similar explorations (vertical axis reports distances). Links of the same color represent a cluster, while dotted links just indicate inter-cluster distances. For easing the interpretation we also color explorations ids, according to ground truth labels. We deliberately chose the same set of colors as clusters to visually highlight the good matches.

Artificial workload. The dendrogram exhibits a perfect match among CubeLoad templates (Slice and Drill, Slide All, Exploratory and Goal Oriented, described in Section 2) constituting well separated clusters. We expected a good result with this workload, as CubeLoad templates are well differentiated.

[3]Metrics for clustering performance evaluation are well described in https://scikit-learn.org/stable/modules/clustering.html sect. 2.3.10. [18]

[4]https://github.com/ClementMoreau-UnivTours/CED_Dolap

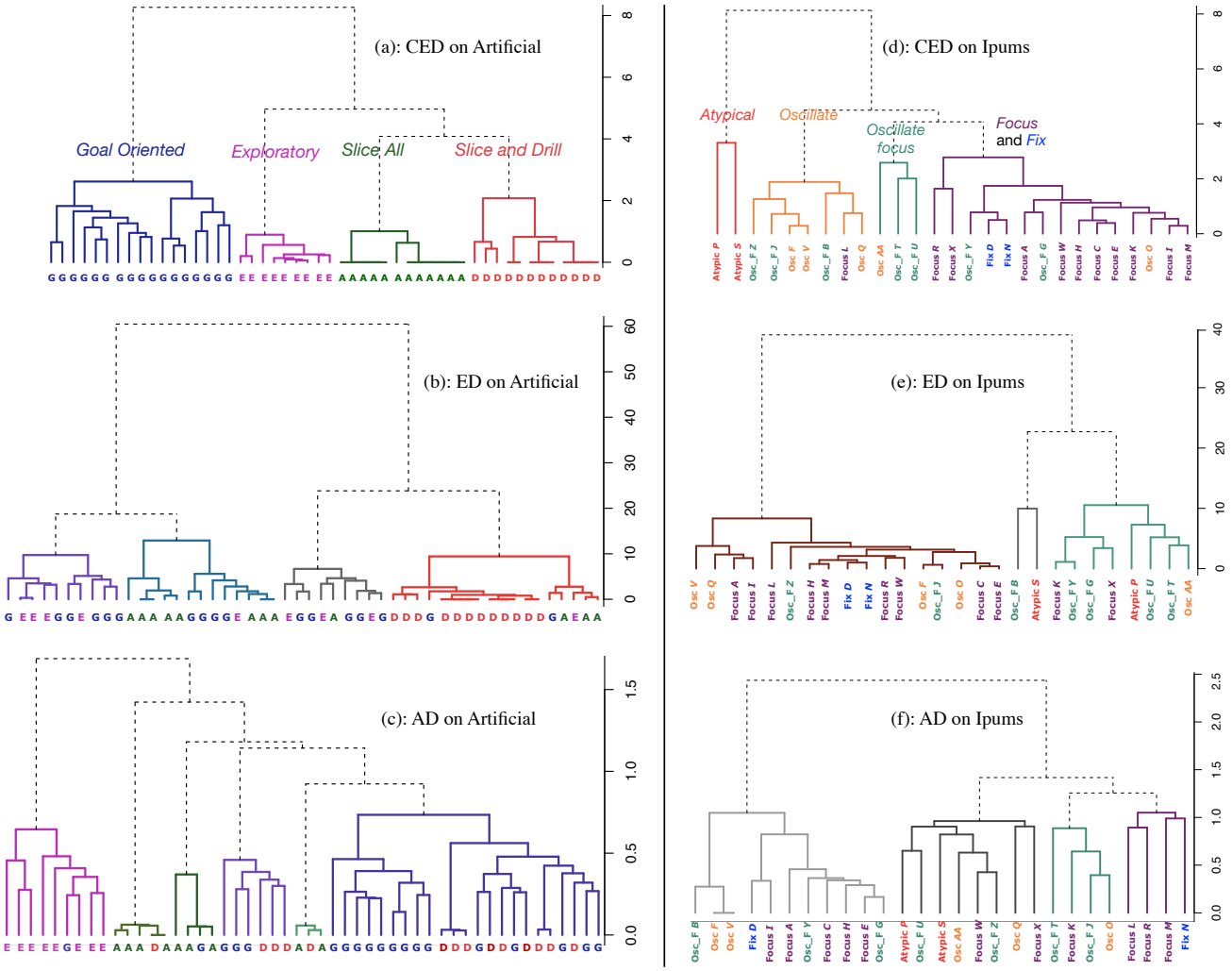

Figure 4: Dendrogram results on Artificial (on left) and Ipums (on right) workloads.

In addition, many explorations of the Slice All template (and some of the Slice and Drill template) are highly similar (distance near 0) as they contain sequences of the very same operations, even if exploration size is variable. This is one of the characteristics that makes CED a well-adapted distance for this problem.

Ipums workload. CED correctly classes most FOCUS and ATYPICAL explorations. However, it fails to distinguish between OSCILLATE-FOCUS and OSCILLATE explorations, the frontier being quite fuzzy, and FIX explorations are not distinguished from FOCUS ones. We remind that these labels are not a real ground truth, but a preliminary classification for other purpose. Table 2 indicates ARI and V-measure scores.

Dataset	Distance	Nb clusters	ARI	V-measure
Artificial	CED	4	1	1
	ED	4	0.26	0.36
	AD	6	0.76	0.88
Ipums	CED	4	0.29	0.42
	ED	3	0.09	0.23
	AD	4	0	0.19

Table 2: Comparison of clustering results for CED, ED and AD distances

6.2 Comparison with other distances

In this experiment we compare the clusters obtained using 3 distances: CED, ED and AD. Results are reported in Table 2 and Figure 4. In Table 2 we can see that CED outperforms AD and ED in both Artificial and Ipums workloads on both quality metrics.

With ED, clusters reflect exploration sizes instead of query operations. For example, in Figure 4(e) the first cluster includes short explorations, the second cluster contains the longest ones, and the last cluster contains medium ones. Conversely, AD relies more on the actual query parts to establish similarity, and tends to cluster together explorations navigating in the same portion of a cube. Consequently, very different behaviors (e.g. those of Slice and Drill and Goal Oriented templates) are clustered together (see Figure 4 (b)). On the other hand CED is solely based on the structural properties of the explorations.

6.3 Other quality considerations

In addition to ARI and V-measure scores (calculated w.r.t. a ground truth), we computed cluster diameters and Silhouette

Dataset	Nb clusters	Max diameter	Silhouette	ARI	V-measure
Artificial	4	2.22	0.49	1	1
Ipums	4	3.31	0.28	0.29	0.42
Open	6	4.53	0.37		
Security	5	3.81	0.16		

Table 3: Nb of clusters, diameter, Silhouette, ARI and V-measure scores for each workload using CED

scores to complete our quality analysis. Results are reported in Table 3 for the 4 datasets.

Globally, we observe that most diameters are low, indicating that clusters are compact. Therefore, medoids are good representatives of each cluster. Most Silhouette scores are also positive, which is a good result given that our clusters are not hyperspherical. In particular, we note that even if CED was able to generate a pure partition for the Artificial workload, we observe a low Silhouette score.

6.4 Interpretation of students' behavior

The next experiment clusters students explorations of the Ipums and Open workloads. We manually examine some explorations of each cluster, including the centroid, with the goal of abstracting students explorations patterns.

Ipums workload. The 27 explorations are arranged in 4 clusters of different sizes.

Clusters of explorations represent the following behavior:

- The 2 explorations in the first cluster start by many changes in the measure set (as trying to choose the good measures), followed by a long period of filter/unfilter operations, combined with some drill-downs and roll-ups, but with little changes in the level of detail.
- Globally, explorations in the second cluster alternates drill-downs and roll-ups. Some of them include a few filters, but most of them (this is the case of the medoid) do not filter any data.
- The 3 explorations of the third cluster start alternating drills down and rolls up (as in cluster 2), then alternate filters/unfilters, some of them focusing at the end (as in cluster 4). This cluster has common points with clusters 2 and 4, some kind of intermediate behavior.
- The last cluster includes focused explorations, which regularly increase the level of detail and filtering by adding drill-down and filter operations. Some of them mix other operations, mostly at the beginning, but drill-downs and filters are the predominant operations.

From a more general perspective, our study shows that half of the explorations follow a focused pattern (cluster 4) translating that those students have developed a particular type of analysis skills, while other students are more exploratory (cluster 2), perhaps translating a lack of maturity in their analysis skills, perhaps just showing their style. Clusters 1 and 3 depicts outlier behaviors.

Open workload. Our method organizes the 28 explorations in 6 clusters, 3 of them containing an outlier exploration, the first one being very different from all the others (inter-cluster distance around 50).

The interpretation of clusters is harder for this workload, as Saiku tool produce very longs explorations (the longest in this workload counts 127 queries). So manual inspection of explorations is tedious and may lead to judgement errors. With this disclaimer, we summarize the behavior represented by clusters as follows:

- The outlier in first cluster is the shortest exploration (4 queries), whose queries are fully aggregated (only ALL levels), and operations only change measures.
- The second cluster contains 10 explorations, many ones being very short, including the medoid. A general behavior is a constant or lightly increasing level of detail, sometimes being high, sometimes medium. Most explorations (except one) have little filters, but exhibit some changes in the measure set.
- The third cluster contains 7 explorations, all of them continuously increasing the level of detail and filtering. There are multiple drill-downs and multiple filters all along the explorations.
- Cluster 4 contains another outlier clustered alone. It exhibits several abrupt changes in the level of detail with several peaks, continuous changes of measures and some filters in the middle.
- Another exploration also clustered alone. It also shows many abrupt changes in the level of detail, with some changes in filters and measures at the beginning and an increase in the filter level at the end.
- The last cluster contains 8 explorations. Its medoid has no filters nor changes in the measure set. OLAP operations include only drill-downs and roll-ups, with an increase of the level of detail in the middle, decreasing at the end of the exploration. Other explorations in the cluster include other OLAP operations. The common behavior resides in the increasing-decreasing pattern in the level of detail.

In conclusion, this clustering confirmed some of the already identified patterns and enabled to discover a new one. Specifically, cluster 3 corresponds quite well to the Focus cluster of Ipums workload and the Slice and Drill template of CubeLoad; while cluster 6 corresponds to the Oscillate cluster of Ipums (having no equivalent template in CubeLoad). The new pattern, reflected by cluster 2, corresponds to less skilled students, making timid usage of OLAP operations (some drill-downs and roll-ups, few filters, some changes of measure). The recognition of outlier behavior is another strong point of our method.

6.5 Interpretation of experts' behavior

This experiment shows the application of our approach to a larger workload, whose explorations were devised by expert analysts using an advanced IDE interface. Surprisingly, many explorations of the Security workload contain only one query (260 out of 723); we excluded them from the study. Another interesting point is that there are long sequences of repeated queries (i.e. the exploration contains many times the same query). This may reflect movements in the way query results are arranged and visualized, while referring to the same underlying query.

There is no ground truth and a manual observation of the 723 explorations is not doable, however, we provide some general comments and a detailed analysis of the medoids of the 5 retrieved clusters (and some randomly picked explorations):

- Explorations in cluster 1 do many movements in the group by set, oscillating the level of detail, with some peaks in

Pattern	Artificial	Ipums	Open	Security
Slice and Drill	4	4	3	
Slice All	3			
Exploratory	2			
Goal Oriented	1			
Oscilating		2	6	1
Oscilate+Focus		3		
Constant Agg. level			2	5
Add-Delete Fragment				2
Few operations				3
Repeted queries				4
Outliers		1	1,4,5	

Table 4: Summary of discovered patterns and ids of the concerned clusters.

the middle. There are other operations; the medoid does many changes in measure set.

- Cluster 2 contains many long explorations, which characteristic is the alternation of adding and deleting one fragment (a level in the group by set, a filter or a measure).
- Explorations in the third cluster are highly similar (many distances are around 0). It includes many short explorations (as the medoid), with few operations, mainly drill-downs.
- Explorations in cluster 4 have very few operations, and globally exhibit long subsequences of identical queries.
- In the last cluster, queries have constant level of detail (generally low), with some movements in the group by set and few filters.

As expected, analysts' behavior is different from students'. Globally, their explorations exhibit less operations, with more emphasis in the grouping of data, probably also in their arrangement and visualization (which is not captured in our method) of the data; while students are more click-oriented and produce longer explorations with much more operations.

Clusters 3 and 4 evidence this behavior and Cluster 2 is some kind of generalization of the Slice All pattern of CubeLoad. Clusters 1 and 5 coincide with those of the other workloads. Table 4 summarizes the discovered patterns.

7 CONCLUSION

This paper addressed the problem of learning analysis patterns in OLAP explorations, which is a hot topic for the understanding of human behavior and providing IDE support. Concretely, we propose to cluster similar explorations using a Contextual Edit Distance, and then extract analysis behavior from clusters. CED is a new distance that is well-suited for comparing explorations taking into account their local context, and then lowering the edition cost of similar queries, repetitions and permutations. Our experiments with four workloads allowed to detect CubeLoad templates and to learn some new analysis patterns from students and expert analysts explorations. Even if these results are promising, our method needs to be tested in larger workloads, with varied skilled and non-skilled users and different types of user interfaces, before abstracting more general patterns.

As future work, we plan to tune our method and study its sensibility and robustness with respect to CED parameters, query features and query similarity. We would also like to compare our results to other clustering methods and distances. Our long term goal is to automatically classify explorations and qualify users skills, allowing the recommendation of pertinent next queries, among other applications.

Finally, we remark that CED is not endemic to analysis behavior and it can be adapted to study other types of human behavior (provided that such behavior could be represented as a sequence of symbols). In particular, we have used CED for comparing human daily moves [16] and we are currently discovering and analysing peculiar patterns of children mobility.

ACKNOWLEDGMENTS

The authors would like to thank Nicolas Labroche for their useful insights on clustering quality. This work is partially supported by French ANR Agency, in the context of Mobi'kids project.

REFERENCES

[1] Aybar C. Acar and Amihai Motro. 2004. Why Is this User Asking so Many Questions? Explaining Sequences of Queries. In *Eighteenth Annual Conference on Data and Applications Security*. 159–176.
[2] Julien Aligon, Kamal Boulil, Patrick Marcel, and Verónika Peralta. 2014. A Holistic Approach to OLAP Sessions Composition: The Falseto Experience. In *DOLAP 2014*. 37–46.
[3] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. 2014. Similarity measures for OLAP sessions. *KAIS* 39, 2 (2014), 463–489.
[4] N. Arzamasova, K. Böhm, B. Goldman, C. Saaler, and M. Schaeler. 2019. On the Usefulness of SQL-Query-Similarity Measures to Find User Interests. *TKDE* (2019), 1–1.
[5] Surajit Chaudhuri and Vivek R. Narasayya. 2007. Self-Tuning Database Systems: A Decade of Progress. In *VLDB*. 3–14.
[6] F. J. Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM* 7, 3 (1964), 171–176.
[7] Mahfoud Djedaini, Krista Drushku, Nicolas Labroche, Patrick Marcel, Verónika Peralta, and Willeme Verdeaux. 2019. Automatic assessment of interactive OLAP explorations. *Inf. Syst.* 82 (2019), 148–163.
[8] Mahfoud Djedaini, Nicolas Labroche, Patrick Marcel, and Verónika Peralta. 2017. Detecting User Focus in OLAP Analyses. In *ADBIS*. 105–119.
[9] Philipp Eichmann, Emanuel Zgraggen, Zheguang Zhao, Carsten Binnig, and Tim Kraska. 2016. Towards a Benchmark for Interactive Data Exploration. *IEEE Data Eng. Bull.* 39, 4 (2016), 50–61.
[10] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. 2014. QueRIE: Collaborative Database Exploration. *TKDE* 26, 7 (2014), 1778–1790.
[11] D. P. Huttenlocher, W. J. Rucklidge, and G. A. Klanderman. 1992. Comparing images using the Hausdorff distance under translation. In *CVPR*. 654–656.
[12] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of Data Exploration Techniques. In *SIGMOD*. 277–281.
[13] Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*. 85–103.
[14] Ralph Kimball. 1996. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. John Wiley.
[15] Tova Milo and Amit Somech. 2018. Next-Step Suggestions for Modern Interactive Data Analysis Platforms. In *KDD*. 576–585.
[16] C. Moreau, T. Devogele, V. Peralta, and L. Etienne. 2020. A Contextual Edit Distance for Semantic Trajectories. *Proc. of 35th ACM/SIGAPP* (2020).
[17] Patrick E. O'Neil, Elizabeth J. O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The Star Schema Benchmark and Augmented Fact Table Indexing.. In *TPCTC* (2009-10-28). 237–252.
[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[19] Stefano Rizzi and Enrico Gallinucci. 2014. CubeLoad: A Parametric Generator of Realistic OLAP Workloads. In *CAiSE 2014*. 610–624.
[20] Oscar Romero, Patrick Marcel, Alberto Abelló, Verónika Peralta, and Ladjel Bellatreche. 2011. Describing Analytical Sessions Using a Multidimensional Algebra. In *DaWaK*. 224–239.
[21] TF. Smith and MS. Waterman. 1981. Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147 (1981), 195–197.
[22] R. Wagner and M. Fisher. 1974. The String-to-String Correction Problem. *J. ACM* 21 (1974), 168–173.