User Interests Clustering in Business Intelligence Interactions

Krista Drushku, Julien Aligon, Nicolas Labroche, Patrick Marcel, Veronika Peralta, Bruno Dumant

User interests clustering in Business Intelligence interactions

Krista Drushku[1,2] Julien Aligon[2,3], Nicolas Labroche[2], Patrick Marcel[2], Veronika Peralta[2], and Bruno Dumant[1]

[1] SAP Research, France
firstname.lastname@sap.com
[2] University of Tours, France
firstname.lastname@univ-tours.fr
[3] University of Toulouse 1 Capitole, France
julien.aligon@ut-capitole.fr

Abstract. It is quite common these days for experts, casual analysts, executives or data enthusiasts, to analyze large datasets using user-friendly interfaces on top of Business Intelligence (BI) systems. However, current BI systems do not adequately detect and characterize user interests, which may lead to tedious and unproductive interactions. In this paper, we propose to identify such user interests by characterizing the intent of the interaction with the BI system. With an eye on user modeling for proactive search systems, we identify a set of features for an adequate description of intents, and a similarity measure for grouping intents into coherent interests. We validate experimentally our approach with a user study, where we analyze traces of BI navigation. We show that our similarity measure outperforms a state-of-the-art query similarity measure and yields a very good precision with respect to expressed user interests.

Keywords: User interest, Feature construction, Clustering, BI analyses

1 Introduction

BI system users range from executives to data enthusiasts who share a common way of interaction: they navigate large datasets by means of sequences of analytical queries elaborated through user-friendly interfaces. For example, users may express their information needs via keywords, and let the system infer from them the most probable formal queries (generally MDX or SQL) to be sent to the underlying data sources (generally data warehouses or databases). As information needs do not have a status per se, it usually takes many interactions with the system to satisfy an information need, and the overall session is often a tedious process, especially in the case when the information need is not even clear for the user. This bears resemblance with web search where users typically need to repeatedly query the search engine to determine whether there is an interesting content.

Being able to automatically identify user interests from BI interactions is a challenging problem that has many potential applications: collaborative recommendation (of data or dashboards), repetitive task prediction, alert raising, etc. that would participate in reducing the tediousness of the analysis. The difficulty of this problem lies in the fact that user interests are hidden in the interactions, and two users with the same interest would probably interact with the system differently. As in web search where users may have no idea of the retrieval algorithm, BI user are generally ignorant of the data sources and the formal queries they trigger. However once logged, all this information (keywords, sources, formal queries, etc.) provide a rich basis for discovering user interests.

In web search, state-of-the-art approaches [6, 12, 16] characterize user interests by means of features extracted from user traces, and classify them to group queries related to the same information needs. We consider that an interaction relies on a sequence of keyword queries over some data sources. Each keyword query produces an ordered set of formal queries suggested from the set of keywords. One of these formal queries, chosen by the user, is evaluated over the data source and the answer retrieved is displayed to the user. All this (keyword query, suggestions and chosen query) is called an observation. We extract a set of features that describe each observation of all user interactions. To group observations into coherent user interests, we first use supervised classification to define a similarity measure that basically assigns a weight to each of the features. Then, we use our measure with an off-the-shelf clustering algorithm to group observations.

If our approach is inspired by the work Guha et al. did in the context of web search [6], it deviates from it on major aspects. First we present our own formal model tailored to BI interactions and we address a specific type of intents. Consistently, we use a specific set of features. Contrarily to [6] we focus more on the expressiveness of the model rather than on specific optimizations for scaling to web data volumes. Finally, our approach is automatic and we present our own evaluation of it, that includes a user study.

More precisely, our contributions include:

- a simple formal model of BI interactions,
- the identification of a set of features for characterizing BI user interests,
- the learning of a similarity measure based on these features,
- an approach to automatically discover user interests based on our measure and an off-the-shelf clustering algorithm,
- an extensive set of experiments for the tuning and validation of our approach, the comparison of our measure with a state-of-the-art metric tailored for OLAP queries [3], and the study of its behaviour in various practical situations.

The paper is organized as follows: Section 2 presents our formal model of BI interactions and user interests. Section 3 details the set of features used to characterize user interests and our algorithm for discovering coherent cross-interaction interests. Section 4 presents our experimental validation. Section 5 gives an overview of related work and Section 6 concludes the paper.

2 Formal model of BI interactions

This section presents our model of BI interaction. Given the proximity of BI interactions in modern BI systems and web searches, our modeling of BI interactions is inspired by the modeling of web search sessions. Note that the generation of formal (MDX or SQL) queries from keywords is out of the scope of this paper.

2.1 BI Questions, Suggestions and Queries

Let D be a database schema, I an instance of D and Q the set of formal queries one can express over D. For simplicity, in this paper, we consider relational databases under a star schema, queried with multidimensional queries [14]. Let A be the set of attributes of the relations of D. Let $M \subset A$ be a set of attributes defined on numerical domains called measures. Let $H = \{h_1, \ldots, h_n\}$ be a finite set of *hierarchies*, each characterized by (1) a subset $Lev(h_i) \subset A$ of attributes called levels, (2) a *roll-up* total order $\succeq_{h_i}$ of $Lev(h_i)$. Let $adom(I)$ be the set of constants of the instance I of D. We call a **database entity** an element of the set $A \cup adom(I)$. The result (or answer) of a query q over a database instance I is denoted $q(I)$.

Let T be a countably infinite set of keywords named tokens. A **BI question** (or question for short) K, is a set of tokens entered by a user. Each token can be matched with the entities in $A \cup adom(I)$ to generate queries. To simplify, we describe a multidimensional query q in Q as a set of query parts, as in [2]. A **query part** is either a level of a hierarchy in H used for grouping, a measure in M, or a simple Boolean predicate of the form $A = v$ involving an attribute A.

For example, starting from BI question *"Revenue for France as Country"* the following tokens {"*Revenue*","*France*","*Country*"} can be identified by excluding stop words. Then, a query may contain the following query parts: *Revenue* is a measure, *Country* a level in a hierarchy, and *France* is a constant, resulting in *Country=France* being a Boolean predicate.

If a query part p is a selection predicate of the form $A = v$, or a grouping attribute A, we use $level(p)$ to denote attribute A. Given two query parts p_1 and p_2, $FD(p_1, p_2)$ denotes that there is a functional dependency $level(p_1) \rightarrow level(p_2)$. Given two queries q_1 and q_2, the boolean expression $OP(q_1, q_2)$ indicates if they differ in at most one query part. This allows to detect OLAP operations when users navigate along hierarchies or change selection conditions.

As keywords are entered, a BI system might on the fly suggest further tokens to complete the current ones, letting the user choose among them, as in web search engines. The underlying idea is that a suggestion completes the original BI question in order to obtain a well-formed query over a database. We formalize the notion of suggestions as follows. A **suggestion** S is a triple $\langle K, D, q \rangle$ where K is a BI question, D is a database schema (called source) and q is a query over D. For short, given a suggestion $S = \langle K, D, q \rangle$, we note $tokens(S)$, $source(S)$ and $query(S)$ for refering to K, D and q respectively.

2.2 Observations, Interactions and user interests

In web search, search histories (i.e., interactions with a search engine) are analyzed to identify coherent information needs, as basis for recommendation generation. For instance, Guha et al. [6] propose to model information needs as sequences of observations, an observation being a search engine query with its associated web results (Search Engine Result Page or SERP for short) and clicks. We adapt the model of [6] to model contexts of BI interactions. This adaptation relies on the following simple analogy: (i) the search engine query corresponds to the BI question, (ii) the SERP corresponds to the set of suggestions associated with the BI question, and (iii) a click on one SERP link corresponds to the choice of a suggestion and hence to the evaluation of the query associated with the suggestion.

Formally, an **observation** o is a triple $o = \langle K, S, s \rangle$ where K is a question, $S = \{s_1, \ldots, s_n\}$ is a set of suggestions for question K, and $s \in \{s_1, \ldots, s_n\}$ is the suggestion selected by the user. Given an observation o, we note K^o the question K of o, $suggestions(o)$ its set of suggestions, and $chosen(o)$ the chosen suggestion. We note $query(o) = query(chosen(o))$, the query of the chosen suggestion, and $result(o) = query(o)(I)$, the result set of the query over a data source instance I. In addition, we annotate each observation o with a binary property indicating the expertise of the user who interacted with the system, denoted $expertise(o)$. For example, consider the question "Revenue for France" of an observation o. There are several suggestions proposed, whose respective questions are: "Revenue for France as Country", "Revenue for France as Market Unit", "Revenue Closed for France as MU/Country/Super Reg", "Revenue Closed for France as Country", etc. Assuming the first suggestion is chosen by the user, it is chosen(o) and the result of the formal query $query(o)$ is $result(o)$.

An **interaction** of length v is a sequence of v observations $i = \langle o_1, \ldots, o_v \rangle$ that represents the user interaction with the BI system. E.g., other questions as: "revenue for France 2010" or "revenue for France 2015" or "revenue closed for France" can follow our question K to create a complete interaction of the user with the system, analyzing the economic growth of France.

Without loss of generality and to keep the formalism simple, we assume that an observation is part of only one interaction. The function $interaction(o)$ returns the interaction to which o belongs. Given two observations o_x and o_y in an interaction, we say that o_y refines (is a **refinement** of) o_x if o_x precedes o_y and either $K^{o_x} = K^{o_y} \cup \{t\}$ or $K^{o_y} = K^{o_x} \cup \{t\}$ or $K^{o_y} = K^{o_x} \setminus \{t\} \cup \{t'\}$, where $t, t' \in T$.

A **user interest** is a finite set $U = \{o_1, \ldots, o_n\}$ of observations that represents one particular information need.

Table 1 presents the basic characteristics we use in our features to describe user interests. Note that $\cup^B$ denotes bag union (preserving duplicates to compute frequencies), P is a set of query parts and $matches(t, p)$ is a binary function indicating if token t matches query part p.

Characteristics	Definition	Interpretation
$questions(U)$	$\cup_{o \in U}\{K^o\}$	all the questions
$tokens(U)$	$\cup^B_{o \in U} K^o$	all the tokens
$suggestions(U)$	$\cup_{o \in U} suggestions(o)$	all the suggestions
$chosenSuggest(U)$	$\cup_{o \in U} chosen(o)$	all the chosen suggestions
$queries(U)$	$\cup^B_{o \in U}\{query(o)\}$	all the chosen queries
$qParts(U)$	$\cup^B_{o \in U} query(o)$	all the chosen query parts
$interactions(U)$	$\cup^B_{o \in U} interaction(o)$	all the interactions
$results(U)$	$\cup_{o \in U} result(o)$	all the results
$sources(U)$	$\cup_{o \in U} source(chosen(o))$	all the sources
$expertise(U)$	$\cup_{o \in U} expertise(o)$	all the expertises
$refTok(U)$	$\{t \in tokens(U) \mid \exists o, o' \in U,$ $t \in (K^o \setminus K^{o'}), o$ refines $o'\}$	tokens that refine other ones
$matchTok(U, P)$	$\{t \in tokens(U) \mid \exists p \in P,$ $matches(t, p)\}$	tokens that match a given set of query parts

Table 1. Basic characteristics of user interests

3 Characterizing and clustering user interests

Following [6], we formalize the problem of discovering coherent user interests as a clustering problem, for which a similarity measure is learned over a set of descriptive features. These features allow to group observations (and user interests) not only based on their intentions expressed by the BI question, but also based on their objectives as expressed by the chosen suggestion, and on their knowledge as provided by the evaluation of the chosen query. To compare two user interests, a global similarity is computed as a weighted sum of feature-based similarity measures. We first define the set of features we consider, together with their similarities, then explain how the features are weighted and how contexts are clustered.

3.1 User interest description features

To provide the best characterization of user interest, we define a set of candidate features, that we subsequently analyze to identify those maximizing the accuracy from the user's perspective. We considered three groups of features. The first group of features relates to the BI questions and suggestions (features 1-6). The second group relates to the chosen suggestions, and especially their query parts (features 7-9). Both groups proved effective in identifying interests in the context of Web searches [6]. The third group consists of specific BI features, and relates to formal queries and their answers (features 10-15).

Table 2 details the features by giving their formal definition and the feature-based similarity measure used for comparing two user interests. The definition is given for a user interest $U_1 = \{o^1_1, \ldots, o^1_n\}$ to be compared to user interest $U_2 = \{o^2_1, \ldots, o^2_m\}$. Given a bag of elements x, $freq(x)$ is a vector counting the number of occurrences of each element of x. For each feature, we propose a similarity

#	*Feature*	*Formal definition*	*Similarity*
1	Frequency of tokens	$freq(tokens(U_1))$	Cosine
2	Frequency of refining tokens	$freq(refTok(U_1))$	Cosine
3	Suggestions	$suggestions(U_1)$	NormInt.
4	BI questions	$questions(U_1)$	NormInt.
5	U_1 questions that are sub-questions in U_2	$\{K \in questions(U_1) \mid \exists K' \in questions(U_2), K' \subset K\}$	MaxFrac.
6	U_1 questions in the same interaction as a question in U_2	$\{K^o \mid o \in U_1, \exists o' \in U_2, interactions(o) = interactions(o')\}$	MaxFrac.
7	Frequency of chosen query parts	$freq(qParts(U_1))$	Cosine
8	Frequency of tokens of U_1 that match chosen query parts of U_2	$freq(matchTok(U_1, qParts(U_2)))$	Cosine
9	Chosen suggestions	$chosenSuggest(U_1)$	NormInt.
10	Levels in chosen query parts	$\{Level(p) \mid p \in qParts(U_1)\}$	Jaccard
11	Tuples retrieved by chosen queries	$results(U_1)$	NormInt.
12	Queries in U_1 that differ by one query part from a query in U_2	$\{q \in queries(U_1) \mid \exists q' \in queries(U_2), OP(q, q')\}$	MaxFrac.
13	Sources	$sources(U_1)$	MaxFrac.
14	Attributes of U_1 functionally identifying attributes in U_2	$\{level(p) \mid p \in qParts(U_1) \exists p' \in qParts(U_2), FD(p, p')\}$	MaxFrac.
15	Expertise of users	$expertise(U_1)$	MaxFrac.

Table 2. Features considered

measure that is the most suited for it (e.g., cosine for vectors of frequencies, Jaccard for sets). The definition of similarity measures MaxFrac and NormInt are drawn from [6]. MaxFrac measures the maximum fraction of observations of each user interest that match an observation in the other user interest. Given two interests U_1 and U_2, it is defined by: $MaxFrac(U_1, U_2) = max(\frac{|O_1^s|}{|O_1|}, \frac{O_2^s}{|O_2|})$, where O_i^s are the observations that satisfy some property s over the total number of observations O_i of U_i. NormInt is a version of Jaccard similarity, that aims at evaluating the number of features two user interests share. It is defined by $NormInt(U_1, U_2) = \frac{|F_1 \cap F_2|}{min(|U_1|, |U_2|)}$, where F_i are the features of U_i and $|U_i|$ is the number of the set of features for the i^{th} user interest.

3.2 Clustering user interests

Grouping observations into user interests, and then grouping similar user interests, requires addressing two problems: (i) determining a similarity measure between user interests and (ii) finding a clustering algorithm that can work on the sole basis of this similarity.

Regarding problem (i), our aim is to distinguish among the candidate features presented above those who are the most suitable to identify coherent interests from a user standpoint. To this end, we formalize the problem as a classification task, which proved effective in [15, 6]. We use a simple linear combination of

feature-based similarity score. The similarity $S(U_1, U_2)$ between user interests U_1 and U_2 is defined by:

$$S(U_1, U_2) = \sum_{i=1}^{n} w_i v_i(U_1, U_2) \quad (1)$$

where n is the number of features, v_i is the similarity measure indicated in Table 2 for feature i and ω_i is a weight representing this feature's importance in the comparison. To set the weights ω_i we use an off-the-shelve SVM linear classifier paired with some ground truth knowledge about user interests to learn the predictive value of the feature. More precisely, for a feature i, the weight ω_i is set to the conditional probability that two observations correspond to the same user interest knowing that they coincide on feature i. This way we solve the tuning problem of finding an appropriate balance between all the features based on the interests that are to be discovered.

Problem (ii) is addressed by experimenting with off-the-shelves well-known and trusted relational clustering algorithms implementing different strategies: centroid-based clustering, connectivity-based clustering and density-based clustering, as explained in the next Section.

4 Experiments

Our objective is to determine a metric based on the features introduced in Section 3.1 that allows, when paired with a clustering algorithm, to group user observations into clusters that reflect accurately user interests. In this regard, the first experiment aims at determining and validating the best subset of features from the set presented in Table 2. Then, a comparative experiment with the state-of-the-art similarity measure for OLAP sessions proposed in [3] shows the effectiveness of our proposal in the particular context of user interests discovery. Incidentally, our experiments also reveal that considering the reference metric [3] as a feature in our similarity measure in some cases improves the overall quality of our approach.

Finally, we propose several side experiments to further validate our approach: (i) sensitivity to the clustering algorithm, (ii) behaviour of our metric when confronted to observations or clusters of observations related to a business need, (iii) behaviour of our metric when confronted to unseen business needs, and (iv) behaviour of our metric in detecting intra-interaction interests.

4.1 Experimental protocol

Data set The data used for our experiments consists in navigation traces of 14 volunteers of SAP covering a range of skills in data exploration, classed, based on their position in the company, in two expertise groups: beginners and expert users. In order to evaluate to which extent actual user interests were discovered by our method, we set 10 business needs (named Q_1 to Q_{10}), each corresponding

to a specific user interest. Users were asked to analyze some of the 7 available data sources to answer each of the 10 business needs, using a SAP prototype that supports keyword-based BI queries[4]. The business needs were grouped in different business cases like: *"For each European country, detect which genres of films did not reach the expected sales"* or *"In which Income Group would you classify a candidate country with a GDP of $6 billion?"*. In order to be more realistic, business needs were defined expecting some overlap in terms of accessed data and queries. In the context of user interest discovery, the business needs Q_1 to Q_{10} serve as our ground truth, our objective being to cluster together observations (potentially from different user interactions) that addressed the same business need.

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}
Difficulty	low	med	med	med	low	high	low	low	med	high
Number of interactions	19	11	10	10	10	8	9	9	9	8
Number of queries	84	65	60	41	50	43	61	51	26	49
Number of relevant queries	34	26	30	16	26	10	27	24	24	9
Queries / interaction	4.4	5.9	6.0	4.1	5.0	5.4	6.8	5.7	2.9	6.1
Relevant queries / interaction	1.8	2.4	3.0	1.6	2.6	1.25	3.0	2.7	2.7	1.1

Table 3. Analysis of business needs

In total, our data set named COMPLETE hereafter contains 23 user interactions, each one possibly concerning several business needs, accounting for 530 queries. Table 3 describes, for each business need, its difficulty, estimated by an expert (in terms of time, number of queries and exploited sources expected in its resolving), the number of interactions devised for solving it, the number of queries and the number of queries perceived as relevant by users in their own activity. In order to have several difficulty settings, we also build two reduced data sets named REDUCED 1 and 2, each corresponding to 4 business needs and 4 distinct data sources, which in turn removes most of the potential overlap. Each of them contains 225 observations. Importantly, REDUCED 1 and 2 are not related to the same business needs. When dealing with these data sets only 4 well separated clusters are to be found, contrary to the COMPLETE data set in which 10 clusters with overlap are expected.

Evaluation of results Our objective is to build groups of observations that are only related to a single user interest. The main indicator of success in our case is thus the precision of the clustering when compared to the theoretical grouping of observations provided by the business questions. At a second level, recall allows to determine to which extent each cluster covers all of the observations related to a user interest. Finally, we classically use the Adjusted Rand Index (ARI) to evaluate the overall quality of the clustering. The values of this index range from

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

around 0 (when the clustering performs badly and produces a partition close to a random clustering) and 1 (when the clustering is perfect) [4].

Metric learning The feature weights are learned over 50% of all observations chosen randomly, with a balance in the number of observations per business needs. Our objective is two-fold and aims at finding the smallest subset of features to avoid any problem of over-fitting when the number of dimensions increases, while still maximizing the quality of the discovery of user interests. To this aim, we tested several subsets of features and trained the weights of the metric with a linear SVM algorithm as presented in Section 3.2 on the sole basis of these features. The subsets of features are selected as follows. We consider all 15 features described in Table 2 and learn the metric. The linear SVM outputs weights that traduce the relative importance of each feature. It is thus possible to order features by the absolute value of their weights. This ranking allows to form subsets of features starting from those with only highly weighted features to subsets that cover more widely the whole set of features. In order to limit to a few subsets, we give results for the following subsets. G2=$\{1, 3, 7, 8, 9\}$, G3=G2$\cup\{5, 10, 11, 13, 14\}$ and ALL respectively include the features with the highest relative importance (the top-5, top-10 and all features). We also constitute a group G1=$\{7, 8, 9, 10, 13\}$ that includes top-5 features selected by repetitively adding to the group those features that increase precision, similarly to [6]. Note that G3 includes both G1 and G2. Finally, groups G4=$\{1, 2, 3, 4, 5, 6\}$ and G5=$\{7, 8, 9, 10, 12, 13, 14\}$ are specific groups of features related only to keywords (G4) and query parts (G5).

Clustering algorithms As no hypothesis can a priori be made on the shape of expected groups of observations, we use in our tests various clustering algorithms that are representative of the diversity of common methods from the literature. The only constraint imposed by the formulation of our problem is that these methods must be relational i.e., only based on the expression of a distance or dissimilarity between pairs of data instances. The first method is the PAM algorithm [8] that is a k-medoids algorithm that finds hyperspherical clusters centered around the k most representative observations. We also use agglomerative hierarchical clustering algorithms [7] with single and complete linkage criterion to either allow for elongated or compact clusters. Finally, we use the traditional DBSCAN algorithm [5] that is not restricted to a specific shape of cluster but constraint clusters to share the same density of points.

Implementation Our approach is implemented in Java but also uses Python Scikit Learn [11] linear SVM to learn the weights of our similarity measure and R clustering packages `cluster` for k-medoids and hierarchical clustering, as well as `fpc` for DBSCAN.

4.2 Results

Determining the best subset of features Table 4 shows that the quality of the discovered groups of observations heavily depends first on the subset of features as expected, but also on the clustering algorithm used. It can be seen that approaches like the hierarchical clustering with single link criterion and DBSCAN algorithms that allows for elongated clusters achieve very poor precision results ($Prec = 0.11$). This can be explained by the fact that these two algorithms are sensitive to potential overlapping between clusters. In our case, similarities between user interests cause early unwanted merging between groups of observations. The stability in precision traduces the fact that these two approaches constantly built a majority of mono-observation clusters and one cluster with almost all the observations, whatever the group of features considered. At the opposite, clustering algorithms that favor compact clusters like the hierarchical clustering with complete link or the k-medoids PAM algorithms perform better. PAM performs significantly better than the hierarchical complete link algorithm, knowing that standard deviations (not reported here for the sake of clarity) do not exceed 10^{-2} and are most of the time around 10^{-3}. Finally, when considering only PAM, it can be seen that the subset of features $G2$ outperforms all the others. Interestingly, these features are those that had the most discriminating behaviour based on the SVM weights observed on all our 15 features (see Section 4.1). Adding more features only slightly increases recall. Other strategies (not mixing features from different specific groups or using the strategy of [6]) can dramatically harm precision. It is also important to note that subset $G2$ does not include BI specific features, which indicates that enough semantics is beared by the other features in detecting user interests. From the previous findings, we define $G2$ as the set of features and we use PAM clustering in the remaining tests, unless otherwise stated.

	H. Single			**H. Complete**			**PAM**			**DBSCAN**		
Features	Rec.	Prec.	ARI	Rec.	Prec.	ARI	Rec.	Prec.	ARI	Rec.	Prec.	ARI
ALL	0.96	0.11	0.002	0.49	0.34	0.315	0.52	**0.46**	0.42	0.82	0.11	0.008
G1	0.90	0.11	0.0004	0.67	0.12	0.026	0.43	**0.40**	0.35	0.86	0.11	0.006
G2	0.92	0.11	-0.0001	0.68	0.11	0.006	0.51	**0.50**	0.44	0.73	0.11	0.017
G3	0.97	0.11	0.001	0.38	0.28	0.23	0.52	**0.47**	0.43	0.77	0.11	0.007
G4	0.96	0.11	-0.0005	0.67	0.14	0.06	0.47	**0.29**	0.26	0.85	0.11	-0.0008
G5	0.91	0.11	0.0004	0.39	0.28	0.23	0.45	**0.42**	0.37	0.75	0.11	0.01

Table 4. Clustering results with distinct subset of features on COMPLETE data set. For short, *Rec*, *Prec* and *ARI* denote respectively recall, precision and ARI scores.

G2 metric behaviour While our metric is learned on observations, our experimental protocol aims at grouping together observations participating in the analysis of a business need. To understand the behaviour of our G2 metric, we tested how it degrades when applied to analyses and then to observations. Analyses are defined as sets of observations participating to answering the same need. This is unlikely to be detected in practice, and this information was explicitly asked to the users when they answered the different needs. Obviously, as shown in Table 5, when applied on analyses, our metric achieves optimal to very good performance. In the easiest case, when user interests are clearly distinct from each others and rich information is provided to our algorithm with analyses rather than observations, the clustering fits perfectly, with precision, recall and ARI scores equal to 1. Interestingly when we cluster analyses based on the metric learned on observations, the results are identical to the previous results. On the contrary, learning metric weights on the basis of analyses (although not realistic) does not conduct to good clusters of observations, with significantly lower scores. As a conclusion, this experiment validates our choice of learning weights on observations and our choice of the G2 features. It is left to future work to address the problem of evaluating the metric on a mixed clustering situation with observations or groups of observations at the same time.

		Complete			**Reduced 1**		
Input	**Weighting**	**Recall**	**Precision**	**ARI**	**Recall**	**Precision**	**ARI**
Observations	Observations	0.51	0.50	0.44	0.70	0.64	0.54
Analyses	Analyses	0.80	0.74	0.74	1.0	1.0	1.0
Analyses	Observations	0.80	0.74	0.74	1.0	1.0	1.0
Observations	Analyses	0.44	0.42	0.36	0.61	0.59	0.45

Table 5. Behaviour of G2 set of features with PAM clustering when learning weights over observations or analysis. Column *"Weighting"* indicates whether weights are learned over observations or analysis.

Comparative experiments Table 6 shows how our metric compares to a reference metric from the literature [3] designed for OLAP queries. This metric has been validated by user tests that showed its effectiveness in grouping queries in accordance to what a human expert would have done. Table 6 reveals 2 distinct behaviours depending on whether we consider the COMPLETE data set or the REDUCED 1 (where clusters are well separated). With the COMPLETE data set, our metric with G2 features performs better than the other metrics as it only relies on the most discriminating features. Indeed, we know from the protocol that groups of observations heavily overlap. Thus, our metric, based on SVM, cannot find a proper linear separation between observations related to different user interests. In this particular context, adding more features makes the problem even more complex to solve for SVM as it has to determine a compromise solution over 15 dimensions rather than 5 in the case of G2 features, and with only a few

	Complete			Reduced 1		
Features	Recall	Precision	ARI	Recall	Precision	ARI
ALL	**0.52**	0.46	0.42	**0.73**	**0.64**	**0.56**
G2	0.51	**0.50**	**0.44**	0.70	**0.64**	0.54
Metric [3]	0.39	0.20	0.14	0.41	0.33	0.10
ALL + [3]	0.40	0.40	0.32	**0.78**	**0.65**	**0.63**
G2 + [3]	0.45	0.43	0.38	0.69	0.62	0.52

Table 6. Comparison of our metric based on G2 features with other metrics when paired with PAM clustering. ALL denotes the set of 15 features, [3] is the state-of-art metric and "+" indicates a metric with added features and corresponding weights.

training instances. On the contrary, with the REDUCED 1 set of observations, groups are clearly separable, the problem is much easier for the linear SVM and adding features may help finding a better solution by fine tuning the separation hyper plane. Consequently, in this case, slightly better results may be achieved with other features than G2's. However, we expect our approach to be the most efficient in any scenarios and the hypothesis that clusters of observations are clearly separated is too strong for us. Thus, the metric based on G2 features seems to be the most appropriate among those that we evaluated but also when compared to state-of-the-art metric like [3].

Handling unseen business needs In this experiment, we study how our method handles previously unseen business needs and how general is the metric learned on the G2 features. To this aim, we consider both REDUCED data sets and use one to train the metric and the other to test with PAM clustering. Recall that reduced data sets cover different business needs, with no overlap among them. Results in Table 7 show that our metric is indeed general and can adapt to new business needs as there is no drop in performance between each of the generalization tests. Moreover, the results are comparable to those observed in previous tests as reported in Table 6. Finally, it can be seen that testing on REDUCED 2 leads to better results than with REDUCED 1. This is expected as REDUCED 2 contains observations related to business need Q9 that has more relevant queries than Q10 contained in the REDUCED 1 data set (see Table 3).

Discovering intra-interaction interests To illustrate one practical interest of our metric, we conducted a test that consists of successively increasing the number of clusters and we checked how many users of different expertise are represented in each cluster. The aim is to show that our metric is good not only at grouping observations that participate to the resolution of a particular business need, but also at identifying parts of the resolution that are shared by users with different expertises. To emphasize on the evolution of precision (which indicates the coherence of clusters), we use the (G2 + [3]) configuration, which is a good

Training	Testing	Recall	Precision	ARI
Reduced 2	Reduced 1	0.76	0.67	0.61
Reduced 1	Reduced 2	0.73	0.71	0.62

Table 7. Generalization of our approach. Each test correspond to the training of the metric and discovery of user interests on different subsets of business needs.

compromise in previous experiment, and test on the well separated Reduced 1 data set, starting with 10 clusters. The results reported in Table 8 show how the mixing of users decreases while precision increases (and consequently recall and ARI decrease) as we increase the number of clusters. It can be noted that for high precisions, the composition of clusters in terms of users with different expertises remains very acceptable. For instance, when precision reaches 95%, more than 63% of clusters have users with different expertise. In other words, this shows that our metric can be used to identify shared sub-tasks (or intra-interaction interests) where some experts' queries could be recommended to beginner users having to solve the same business need.

# clusters	Recall	Precision	ARI	Dense UI	Expertise
10	0.35	0.86	0.41	10 (100%)	0 (100 %)
15	0.24	0.90	0.31	14 (93.3%)	1 (93.33 %)
20	0.20	0.92	0.26	14 (70%)	2 (90 %)
25	0.18	0.92	0.24	13 (52%)	6 (76 %)
30	0.17	0.95	0.23	13 (43.3%)	11 (63.33 %)
35	0.16	0.95	0.22	12 (34.3%)	16 (54.29 %)
50	0.14	0.96	0.19	11 (22%)	30 (40 %)

Table 8. Increasing the number of clusters to detect intra-interaction interests. Dense UI indicates the number of clusters with more than 5 different users. Expertise indicates the number of clusters with both types of users (beginners and experts).

5 Related work

Analyzing web search sessions for personalizing user experience has attracted a lot of attention, varying in models for session, similarities and clustering algorithms [9]. As users information needs span multiple search sessions, state-of-the-art approaches attach importance to both intra and inter-session similarities. Various forms of user interests have been defined, like contextual intent, task repetition or long term interests, and methods have been proposed to identify them. Sun et al. [13] are interested in contextual intent. Contextual intent attaches

importance to context with a particular emphasis on external physical environment, and complex context-intent relationships are modeled. Consequently, intent tracking is done in real-time. In our work, we are not interested in modeling context, nor real-time tracking, but in user interest in certain data to answer a particular business question, which is generally context-independent. Song and Guo [12] address the problem of predicting task repetition, i.e., whether a task represents one-time information need or exhibits recurrent patterns. A feature-based approach is used to train a deep neural network classifier to recognize the characteristics of task repetition patterns. The features incorporate information on queries, clicks, and attach a particular importance on time, with the underlying assumption that similar users often perform similar activities at similar time. A similar approach is proposed by Guha et al. [6]. The goal is to discover new intent and obtain content relevant to users' long-term interests. They develop a classifier to determine whether two search queries address the same information need. This is formalized as an agglomerative clustering problem for which a similarity measure is learned over a set of descriptive features (the stemmed query words, top 10 web results for the queries, the stemmed words in the titles of clicked URL, etc.). One advantage of this approach is that it allows to build contexts that span over several user sessions or only a portion of one session. Thus, contexts provide insights on short and long terms information needs and user habits, to build accurate user profiles.

To the best of our knowledge, our work is the first attempt to automatically discover BI users' interests in a multi-user environment. Some collaborative recommendation approaches for BI exist, but they are limited to clustering OLAP queries or sessions without treating user interest as a first class citizen (see e.g., [1]). A similarity measure tailored for OLAP queries is proposed in [3]. This work also reviews query similarity measures described in the literature, and showed through user studies that the proposed measure better respects the similarity perceived by users over the other measures. This led us to compare our measure to that one. Nguyen et al. [10] deal with discovering the most accessed areas of a relational database. Their notion of user interest relies on the set of tuples that are more frequently accessed, and is expressed as selection queries (mostly range queries). They use DBSCAN to cluster user interests. Their similarity metric relies on Jaccard coefficient of the accessed tables and on overlapping of predicates. Being tailored for range queries, their metric is inappropriate for OLAP queries that are mostly dimensional (i.e., point based), due to the nature of the hierarchical dimensions used to select data. In particular, consistently with the study of [3], the query log used for our tests feature no range queries.

6 Conclusion

We have presented an approach for identifying coherent interests of BI users with various expertise querying datasources by means of keyword-based analytical queries. Our approach relies on the identification of discriminative features for characterizing BI interactions and on the learning of a similarity measure based

on these features. We have shown through user tests that our approach is effective in practice and could benefit beginner analysts whose interests match those of expert users. Overall, our results show that keyword-based interaction systems provide semantically rich user traces well adapted to the detection of coherent BI user interest.

Building upon these results, our long term goal is to go beyond keyword-based interaction systems. We envision the implementation of an intelligent assistant that raises alerts when the datasources are refreshed or when user information needs and expertise change. To this end, our future works include the development of interest and skill-based recommendation approaches and their validation via larger user studies.

References

1. J. Aligon, E. Gallinucci, M. Golfarelli, P. Marcel, and S. Rizzi. A collaborative filtering approach for recommending OLAP sessions. *DSS*, 69:20–30, 2015.
2. J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia. Mining preferences from OLAP query logs for proactive personalization. In *ADBIS*, pages 84–97, 2011.
3. J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia. Similarity measures for olap sessions. *KAIS*, 39(2):463–489, 2014.
4. B. Desgraupes. Clustering indices. Technical report, University Paris Ouest - Lab Modal'X, April 2013.
5. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, USA, 1996. AAAI Press.
6. R. Guha, V. Gupta, V. Raghunathan, and R. Srikant. User modeling for a personal assistant. In *WSDM*, pages 275–284, Shanghai, China., 2015.
7. L. Kaufman and P. Rousseeuw. *Finding groups in Data: An introduction to Cluster Analysis*. Wiley.
8. L. Kaufman and P. Rousseeuw. Clustering by means of medoids. In *Statistical Data Analysis based on the L1 Norm*, pages 405–416. Elsevier, 1987.
9. B. Mobasher. Data mining for personalization. In *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *LNCS*, pages 90–135. 2006.
10. H. V. Nguyen and al. Identifying user interests within the data space - a case study with skyserver. In *EDBT*, pages 641–652, 2015.
11. F. Pedregosa and al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. Y. Song and Q. Guo. Query-less: Predicting task repetition for nextgen proactive search and recommendation engines. In *WWW*, pages 543–553, 2016.
13. Y. Sun, N. J. Yuan, Y. Wang, X. Xie, K. McDonald, and R. Zhang. Contextual intent tracking for personal assistants. In *SIGKDD*, pages 273–282, 2016.
14. A. A. Vaisman and E. Zimányi. *Data Warehouse Systems - Design and Implementation*. Data-Centric Systems and Applications. Springer, 2014.
15. H. Wang, Y. Song, M.-W. Chang, X. He, R. W. White, and W. Chu. Learning to extract cross-session search tasks. In *In WWW*, 2013.
16. L. Yang, Q. Guo, Y. Song, S. Meng, M. Shokouhi, K. McDonald, and W. B. Croft. Modeling user interests for zero-query ranking. In *ECIR*, pages 171–184, 2016.