Interest-based recommendations for business intelligence users

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

Interest-based recommendations for Business Intelligence users

Krista Drushku[a,b,*], Julien Aligon[c,b], Nicolas Labroche[b], Patrick Marcel[b], Verónika Peralta[b]

[a] *SAP Research, France*
[b] *University of Tours, France*
[c] *University of Toulouse 1 Capitole, France*

Abstract

It is quite common these days for experts, casual analysts, executives and data enthusiasts, to analyze large datasets through 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 a collaborative recommender system for BI interactions, specifically designed to take advantage of identified user interests. Such user interests are discovered by characterizing the intent of the interaction with the BI system. Building 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 clusters. On top of these automatically identified interests, we build a collaborative recommender system based on a Markov model that represents the probability for a user to switch from one interest to another. We validate our approach experimentally with an in-depth user study, where we analyze traces of BI navigation. Our results are two-fold. First, 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. Second, we compare our recommender system to two state-of-the-art systems to demonstrate the benefit of relying on user interests.

Keywords: User interest, Feature construction, Clustering, BI analyses, Collaborative recommender systems

*Corresponding author
Email addresses: Krista.Drushku@sap.com (Krista Drushku), julien.aligon@ut-capitole.fr (Julien Aligon), Nicolas.Labroche@univ-tours.fr (Nicolas Labroche), Patrick.Marcel@univ-tours.fr (Patrick Marcel), Veronika.Peralta@univ-tours.fr (Verónika Peralta)

1. Introduction

BI system users range from executives to data enthusiasts who share a common way of interaction, i.e., 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 likely formal queries (generally MDX or SQL) to be sent to the underlying data sources (generally data warehouses or databases). 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 interesting content.

Being able to automatically identify user interests from BI interactions is a challenging problem that has many potential applications, such as suggestion of interesting data found by other users, repetitive task prediction, alert raising, etc. that would help reduce the tediousness of the analysis. In this article, we are particularly interested in showing how user interest detection helps devise recommendations of queries for the users to pursue their interaction with the BI system. To this end, we first describe a method for BI user interest detection and then we propose an original interest-based recommender system.

The challenge of user interest detection lies in the fact that 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 users are generally ignorant of the data sources and the formal queries they trigger. However, once logged, all this information (keywords, sources, formal queries, etc.) provides a rich basis for discovering user interests. In Web Search, state-of-the-art approaches [1, 2, 3] characterize user interests by means of features extracted from user traces and classify them in order to group queries related to the same information needs. In the context of BI, 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 then, 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 the observations. This modeling of BI interactions and this detection of user interests were the focus of our earlier work presented in [4], and were inspired by the work Guha et al. did in the context of Web Search [1]. While the present paper extends this work by providing additional details and examples of user interest detection, as well as new tests to better characterize the clusters obtained, its main objective is to demonstrate the practical benefit of clustering user interests in the context

of BI interactions and, more specifically, in implementing interest-based query recommender systems.

Recommending queries suitable for BI interactions has indeed attracted much attention recently, either in the context of On-Line Analytical Processing (OLAP) explorations (see e.g., [5]) or SQL sessions (see e.g., [6]). However, to our knowledge, none of the proposed approaches considered clustering user interests to generate recommendations. To leverage discovered user interests for the purpose of query recommendation, we present an original interest-based recommender system, named *IbR*, that we test and compare to two state-of-the-art approaches. *IbR* is a collaborative recommender system that adopts a technique first proposed in the literature for query prediction [7] and later extended for query recommendation [8]. This latter work uses an unsupervised learning approach to cluster OLAP queries based on their syntax, then builds a Markov model whose states are the clusters, and transitions between clusters represent the moving from one query to another, less similar one. While our present approach also builds a Markov model on top of predetermined clusters, it differs from the one proposed in [8] on a major point: we ensure that clusters correspond to coherent interests thanks to the supervised learning of the metric used for the clustering.

Precisely, the contributions presented in this article 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,
- *IbR*, a simple yet effective user interest-based collaborative recommender system inspired by earlier works [8] in OLAP query prediction, specifically designed to take advantage of the clusters identified, that can recommend a sequence of queries to complement an existing interaction,
- an extensive set of experiments for the tuning and validation of our approach through a user study, including the comparison of our similarity measure with a state-of-the-art metric tailored for OLAP queries [9] and the comparison of our recommender system with two state-of-the-art recommender systems agnostic of user interests [6, 5]. To our knowledge, this is the first time such a comparison is performed. Note that we do not compare our recommender system to that proposed by Aufaure et al. in [8], since our recommender can be seen as a natural evolution of the latter to more robust clusters.

The paper is organized as follows: Section 2 gives an overview of related work. Section 3 presents our formal model of BI interactions and user interests. Section 4 details the set of features used to characterize user interests and our

algorithm for discovering coherent cross-interaction interests. Section 5 introduces our interest-based recommender system. Sections 6 and 7 present our experimental validation and Section 8 concludes the paper.

2. Related work

Analyzing web search sessions for personalizing user experience has attracted much attention, varying in models for session, similarities and clustering algorithms [10]. As the 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, such as contextual intent, task repetition or long term interests, and methods have been proposed to identify them. Sun et al. [11] are interested in contextual intent. Contextual intent attaches importance to context with a particular emphasis on the 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 the context, nor real-time tracking, but in modeling the user's interest in certain data to answer a particular business question, which is generally context-independent. Song and Guo [2] address the problem of predicting task repetition, i.e., whether a task represents a 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 to time, with the underlying assumption that similar users often perform similar activities at similar times. A similar approach is proposed by Guha et al. [1]. 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 for the building of contexts that span over several user sessions or only a portion of one session. Thus, contexts provide insights on short and long term information needs and user habits, to build accurate user profiles. Our approach of discovering user interests is inspired by the work Guha et al. did in the context of Web Search [1]. However, our work deviates from it on major aspects. First, we present our own formal model tailored to BI interactions and we address a specific type of intent. Consistently, we use a specific set of features. Second, we focus more on the expressiveness of the model rather than on specific optimizations for scaling to web data volumes. Finally, our approach is entirely automatic, and we present our own evaluation of it, which includes a specific user study.

While many recent works propose techniques to assist query formulation for the purpose of interactive database exploration [12], few works address the issue of discovering user interests based on past queries without asking for explicit

feedback. Many approaches are more focused on extracting navigational behavior than user interests. The SnipSuggest approach [13] analyzes a log of past queries to extract correlations between SQL clauses, and use them to suggest SQL snippets for auto-completing SQL queries on the fly. In the DICE approach [14], the query log is used to prioritize speculative queries transparently sent for the purpose of pre-fetching. With the same pre-fetching motivation, the Promise approach [7] builds Markov models out of a query log to represent a user's behavior when querying a data cube. Two Markov models are built. In the first one, the states represent query patterns, i.e., groups of queries with the same group by set. This model allows the probabilities of a pattern to be obtained when given another one. The second Markov model focuses on the selection of predicate values. This model obtains the probabilities to change a selected value by another one. At run time, these two models are used to obtain the query most likely to follow the query just executed. The Promise approach has influenced later work on query recommendation [8]. In this work, Aufaure et al. probabilistically model user behavior using a Markov model built on top of query clustering using the similarity metric proposed in [9]. Identified clusters are the states of the Markov model and log sessions define transition probabilities. Like in Promise, this model is used at runtime to find the most likely query to follow a given current query. Perhaps the closest to our interest detection approach is the work of Nguyen et al. [15] dealing 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 the 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, consistent with the study by Aligon et al. [9], the query log used for our tests includes a very small fraction of range queries.

Recommender systems are now well established as an information filtering technology and field of research, with application in a wide range of domains [16, 17]. The most prominent technique in recommendation is collaborative recommendation, which is often implemented for predicting missing ratings in a user-item matrix recording user preferences. Recommender systems are classically evaluated with predictive accuracy-based measures, coverage, which is the degree to which recommendations can be generated to all users or items, or diversity, which is the capacity to suggest relevant items of a different nature. The limitations of collaborative recommenders are well known, in particular new items cannot be recommended without relying on some additional knowledge source. It is also well known that the application domain exerts a strong influence over the types of methods that can be successfully applied.

Recent approaches value models of users or items that can be described in terms of, e.g., semantic technologies, concepts and ontologies [18, 19], or using contextual information [20].

Many recent proposals investigated the use of database query logs for query

recommendation in the context of interactive database exploration. One of the most prominent proposals is the QueRIE approach [6], where collaborative filtering is used to recommend a set SQL queries. Interactions in QueRIE are sequences of SQL queries called sessions. QueRIE constructs a matrix with logged sessions, where all past queries and sessions are vectorized using the database tuples or the SQL fragments (selections, projections, etc.). When a new session is created and a recommendation for it is sought, a KNN approach is used with classic Jaccard or Cosine similarities to find the queries closest to this session in the matrix.

BI can be seen as a prototypical context of interactive database exploration. In OLAP systems, for instance, the typical interaction between the user and the system consists of a sequence of queries answering a business question. Studying such sequences can help in suggesting what the next query of the session could be. A survey of query recommendation approaches for BI is proposed in [5], showing that user interest is poorly considered in state-of-the-art approaches. In [5], Aligon et al. propose a recommender system tailored for exploratory OLAP over a datacube, whose sessions can be seen as a particular type of BI interactions, where queries are regular dimensional queries [21]. This system recommends a sequence of queries and proceeds in three steps. When a new session is created and a recommendation for it is sought, a KNN approach is used, with a customized session similarity, to search in the log for the closest past sessions. Then one subsequence that best fits the new session is determined. Finally, this subsequence is adapted to the new session using patterns identified in the new session and the subsequence. This approach builds upon a review of query similarity measures and the proposal of two similarity measures: one tailored for OLAP queries and another tailored for OLAP sessions [9]. The authors showed through user studies that the proposed measures better respect the similarity perceived by users over the other measures reviewed.

To the best of our knowledge, our work is the first attempt to automatically discover BI users' interests in a multi-user environment and to show that detecting user interests in past user traces helps query recommendation.

3. 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.

3.1. 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 star schemata, queried with multidimensional queries [22]. Without lack of generality, we consider D to be a global schema resulting

from the integration of several data sources. We simply note $sources(q)$ the set of sources used by a query q over D.

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 and (2) a *roll-up* total order $\succeq_{h_i}$ of $Lev(h_i)$. Consistent with the literature on database theory [23], we denote by $adom(I)$ the set of all constants appearing the instance I of D, i.e., the constants that are used to form the tuples of instance I. 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 by $q(I)$.

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. Note that tokens may contain stop words (e.g. "by" or "as") that have no equivalent in formal queries but that the BI system may use in formal queries generation. The reason for including such stop words in our model is that use of stop words may convey a certain user behavior or expertise with the system. Each token may 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 [24]. 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 $a\ op\ v$, where a is a level of a hierarchy in H, op is an operator in $\{=, <, >, \leq, \geq, \neq\}$ and v is a constant in $adom(I)$. In what follows, queries are confounded with their sets of query parts, unless otherwise stated.

Example 1. *Starting from the question "Revenue for France as Country" the following tokens* $K_1 = \{"Revenue", "for", "France", "as", "Country"\}$ *are identified. A corresponding formal query contains 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 $a\ op\ v$, 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 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 for the detection of 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 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 and q is a query over D. For short, given a suggestion $S = \langle K, D, q\rangle$, we note $tokens(S)$ for referring to K, $query(S)$ for referring to q, and $sources(S)$ for referring to $sources(q)$.

Example 2. *The question* $K_1 = \{"Revenue","for","France","as","Country"\}$ *is completed to focus on year 2017. The corresponding suggestion,* $S_{11} = \langle K, D, q\rangle$*, consists of question* $K = \{$*"Revenue", "for", "France", "as", "Country", "and", "2017", "as", "Year"*$\}$*, schema* D*, that includes a relation Sales, and the formal query* q*, represented by its three query parts* $\{Revenue, Country = France, Year = 2017\}$*, whose SQL code is:*
SELECT sum(*Revenue*) FROM Sales WHERE *Country*='*France*' AND Year=2017;

3.2. 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. [1] propose modeling 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 proposed by Guha et al. in [1] 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 by $expertise(o)$.

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.

Example 3. *Consider question* $K_2 = \{"Revenue","for","France"\}$ *of an observation* o_2*. Several suggestions are proposed, with respective questions:* $\{$*"Revenue", "for", "France", "as", "Country"*$\}$*,* $\{$*"Revenue", "for", "France", "as", "Market Unit"*$\}$*,* $\{$*"Revenue Closed", "for", "France", "as", "Country"*$\}$*, etc. Assume the first suggestion is chosen by the user, its formal query* $query(o_2)$ *is evaluated and its result* $result(o_2)$ *is displayed to the user. Other questions* $\{$*"Revenue", "for", "France", "2010"*$\}$ *or* $\{$*"Revenue", "for", "France", "2015"*$\}$*,* $\{$*"Revenue Closed", "for", "France"*$\}$ *follow, to create a complete interaction of the user with the BI 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

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
$sources(U)$	$\cup_{o \in U} sources(chosen(o))$	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

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.

3.3. Running example

Consider two user interests U_1 and U_2, with U_1 containing a unique observation $o_1 = \langle K_1, \{S_{11}\}, S_{11}\rangle$ of a unique interaction i_1 (described in Example 2), and U_2 containing two observations $o_2 = \langle K_2, \{S_{21}, S_{22}, S_{23}\}, S_{21}\rangle$ and $o_3 = \langle K_3, \{S_{31}, S_{32}\}, S_{32}\rangle$ that are part of the interaction i_2 (described in Example 3). The observations are summarized in Table 2 by listing questions and suggestions (by means of their questions and query parts). For the sake of readability, Table 2 describes the suggestions only by means of their queries. Assume that all suggestions access the same database and the same underlying source *Sales* and that all users are beginners.

o_1 :	K_1	"Revenue for France as Country"
	S_{11} :	{"Revenue", "Country=France", "Year=2007"}
o_2 :	K_2	"Revenue for France"
	S_{21} :	{"Revenue", "Country=France"}
	S_{22} :	{"Revenue", "Market Unit=France"}
	S_{23} :	{"Revenue Closed", "Country=France"}
o_3 :	K_3	"Revenue for France 2010"
	S_{31} :	{"Revenue", "Year", 'Country=France"}
	S_{32} :	{"Revenue", "Country=France", "Year=2010"}

Table 2: Summary of observations in the running example

Table 3 presents the characteristics of both user interests. For the sake of space, we do not show the query results. RefToken is defined when an observation refines another one, in which case it contains the tokens involved in the refinement. MatchTok contains the tokens that match some set of query parts; in Table 3 we used $P = \{$"$Revenue$", "$Country = France$", "$Year = 2007$"$\}$, arbitrarily.

	U_1	U_2
questions	{{Revenue, for, France, as, Country}}	{{Revenue, for, France}, {Revenue, for, France, 2010}}
tokens	{Revenue, for, France, as, Country}	{Revenue, Revenue, for, for, France, France, 2010}
suggestions	$\{S_{11}\}$	$\{S_{21}, S_{22}, S_{23}, S_{31}, S_{32}\}$
chosenSuggest	$\{S_{11}\}$	$\{S_{21}, S_{32}\}$
queries	{{Revenue, Country=France, Year=2007}}	{{Revenue, Country=France}, {Revenue, Country=France, Year=2010}}
qParts	{Revenue, Country=France, Year=2007}	{Revenue, Revenue, Country=France, Country=France, Year=2010}
interactions	$\{i_1\}$	$\{i_2, i_2\}$
sources	{Sales}	{Sales}
expertise	{beginner}	{beginner}
refTok	$\emptyset$	{2010}
matchTok	{Revenue, France, Country}	{Revenue, France}

Table 3: Characteristics of user interests in the running example

4. Characterizing and clustering user interests

Following Guha et al.'s approach [1], 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 observations (and user interests) to be grouped based not only 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 the contexts are clustered.

4.1. 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, listed in Table 4. 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 [1]. The third group consists

of specific BI features and relates to formal queries and their answers (features 10-15).

#	*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 4: Features considered

Contrary to most works where features are descriptive and relate to each data object independently of the others, in our proposal, features should be understood as dimensions on which it is possible to compare two user interests, $U_1 = \{o_1^1, \ldots, o_l^1\}$ and $U_2 = \{o_1^2, \ldots, o_m^2\}$. As a consequence, each feature has a proper semantic attached to it, as it describes a particular aspect of a relation between the user interests - for example the number of occurrences of tokens in the questions of each user intent. Then, each feature f is also paired with a similarity measure denoted by $v_f \in \mathbb{R}$, which hereafter quantifies this relation.

Table 4 details the features by giving their formal definition and the feature-based similarity measure used for comparing two user interests. 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 measure that is the most suited for it (e.g., cosine for vectors of frequencies, Jaccard for sets). We follow the same logic as in [1], and, in particular, the definition of similarity measures MaxFrac and NormInt are drawn from [1]. 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 $NormInt(U_1, U_2) = \frac{|F_1 \cap F_2|}{min(|F_1|, |F_2|)}$, where F_i is the set of features of U_i, the i^{th} user interest, and $|F_i|$ is the cardinality of this

set of features. It is important to note that, different from Jaccard similarity, NormInt favors the merging of U_1 and U_2 whenever $U_1 \subseteq U_2$.

Example 4. *Let consider the user interests described in the running example (Section 3.3) and the feature 1 from Table 4.*

$freq(tokens(U_1)$ = *<(Revenue,1),(for,1),(France,1),(as,1),(Country,1)>*
$freq(tokens(U_2)$ = *<(Revenue,2),(for,2),(France,2),(2010,1)>*

By following a bag-of-word representation for both user interests, we end-up with the vectors t_1 and t_2 representing respectively tokens' frequency for users U_1 and U_2.

	Revenue	*for*	*France*	*as*	*Country*	*2010*
t_1	*1*	*1*	*1*	*1*	*1*	*0*
t_2	*2*	*2*	*2*	*0*	*0*	*1*

Following Table 4, the similarity for feature 1 is a cosine measure defined as: $v_1(t_1, t_2) = \frac{<t_1,t_2>}{\sqrt{<t_1,t_1>}*\sqrt{<t_2,t_2>}}$ where $<,>$ denotes the inner product. Here, the similarity would be $v_1(t_1, t_2) = 0.74$, meaning that on this particular feature user interests U_1 and U_2 are relatively close to each other.

4.2. 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. As we are expecting an understandable model that provides the relative importance of each feature in the process of comparing two user interests U_1 and U_2, we rely on a linear aggregation for our similarity $Sim(U_1, U_2)$ defined as follows:

$$Sim(U_1, U_2) = \sum_{f=1}^{n} \omega_f v_f(U_1, U_2) \tag{1}$$

where n is the number of features, v_f is the similarity measure indicated in Table 4 for feature f and ω_f is a weight representing this feature's importance in the comparison.

With this formulation, the problem of designing a similarity naturally translates into a problem of determining the set of weights ω_f paired with each similarity measure v_f.

To this end, we formalize the problem of discovering ω_f as a classification task, which proved effective in [1, 25]. Indeed, we are able to train a classifier (X, Y) in which each entry $x \in X$ corresponds to a couple of user interests, the

descriptive features of each entry being the one introduced in Table 4 and the output $y \in Y$ being set to 1 if two users, U_1 and U_2, relate to the same interest, and -1 otherwise.

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. For a feature f, the weight ω_f is set to the conditional probability that two observations correspond to the same user interest knowing that they coincide on feature f. The absolute value of ω_f reflects how discriminant feature f is (a large value indicates that feature f is very influential in the decision process), while the sign of ω_f denotes that feature f will either act in favor of grouping user interests or, conversely, to separate them. In particular, the descending list of the absolute value of weights ranks the features, stating from the most important one.

Noticeably, some preprocessing and optimizations have been performed to ensure that our SVM is accurate. First, our data set of user interests' couples has been balanced to guarantee that there were the same number of couples related to the same user interest (labeled 1) as the couples related to different user interests (labeled -1). Second, the hyper parameter C, which traditionally determines the balance between the flatness of solution weights and the amount up to which it is possible to deviate from the regulation term in the SVM optimization model, has been tuned to its best possible value by an extensive cross validated random search. Finally, we note that we disregard the weight ω_0 learned by the linear SVM classifier, as it only plays the role of an offset in the similarity S.

Problem (ii) is addressed by experimenting with off-the-shelf well-known and trusted relational clustering algorithms implementing different strategies, i.e., centroid-based clustering, connectivity-based clustering and density-based clustering, as explained in Section 6.1.4.

5. Interest-based recommender system

To illustrate the practical use of our approach, in this section, we present *IbR* (Interest-based Recommender), a simple recommender system specifically designed to exploit the clusters that represent user interests. IbR is inspired by and adapts previous approaches proposed to predict or recommend OLAP queries.

First, inspired by the collaborative recommender system described in [5], IbR recommends a *sequence of queries* representing the sequence of moves that is expected to best complete the beginning of an interaction. As remarked by Aligon et al. in [5], it is expected that users, especially non-expert ones, benefit from a sequence of recommended queries, in that it gives them a compound and synergic view of a phenomenon, carries more information than a single query or set of queries by modeling the potential expert users behavior after seeing the result of the former query.

Second, we borrowed from the work of Sapia [7] and the work of Aufaure et al. [8] the idea of using an order-1 Markov model to probabilistically represent user

behaviors. Like in the latter [8], the states of the Markov model are clusters constructed from a set of past interactions, with the notable difference that observations are used in our case, instead of queries. IbR can be seen as a model that guides the user's next moves based on the probabilities of moving between discovered user intents.

By construction, we expect our recommender system to have two types of benefits: the first one is sharing expertise between different users, and the second one is recommending queries that are diversified in terms of interest.

5.1. Principle

The principle of the recommender system follows the same two-step approach as that of Aufaure et al. [8]. The first step is off-line and consists of clustering the observations to detect user interests, as detailed in the previous sections. The second step consists in treating these clusters as states of a Markov chain model and in computing the probabilities of the most likely next state as explained below. The only on-line phase of the recommender is when a new interaction begins, each observation of the interaction is used to compute the most likely query in the sense of the Markov model. An exhaustive calculation is needed for the first observation to define its current state by comparing it with every observation of every state. For the rest of observations, an exhaustive search of its current state is not needed since the recommendation only derives from the previous recommendations, i.e., the last state calculated by the Markov model.

5.2. Learning the Markov model

The creation of the Markov model is done as follows. Let U be the set of clusters expressing user interests. The states of the Markov model are the clusters of U. The transition probability distribution is given by $\Pr(X_{n+1} = x \mid X_n = y) = \frac{n_{xy}}{n_y}$ where x and y are clusters in U, n_y is the size (the number of observations) of cluster y and n_{xy} is the number of interactions that contain two adjacent observations o_i, o_{i+1} such that o_i is in cluster y and o_{i+1} is in cluster x. We use a special state to represent the end of interactions, which is used to obtain the probability of ending the recommendation.

5.3. The prediction algorithm

Given an observation, called the *current observation* (whose chosen query is called the current query) from now on, we identify the user interest (i.e., the cluster) that this observation is the closest to by computing the average similarity between the current observation and all the observations of each cluster. Noticeably, instead of all pairwise distance calculations, a possible optimization would be to directly compute the distance between the current observation and the representative of each cluster. However, as this optimization is not guaranteed to lead to the same result as that obtained with all pairwise calculations, especially in the presence of overlapping clusters, we deliberately chose not to implement it.

Once we have identified the cluster, the Markov model gives the most likely next state. By construction, since states coincide with user interests, it is expected that the most likely next state is the current one. To distinguish between the two types of benefits our recommender can have, we devised two strategies for generating the recommended sequence, reflected in the two modes our recommender can operate. Mode 1, named IbR1, tries to benefit from the expertise coming from this next probable cluster only. Conversely, Mode 2, named IbR2, tries to anticipate when users change their focus and, for instance, address other business questions in their explorations. IbR2 fully combines the Markov model with interest detection, with a two-fold purpose, which is: i) to anticipate the users' change of focus, and ii) to propose recommendations diversified in terms of user interests. We now describe these two modes precisely.

IbR1 forces the recommender to choose the queries for the recommended observations in the next probable state only. In other words, IbR1 does not use the Markov model but uses only interest identification. The chosen queries are ordered by decreasing similarity to the current query. The length of the recommended sequence is ruled by a similarity threshold that ends the sequence if the similarity between two consecutive queries is considered too small.

The second mode, named IbR2, fully uses the Markov model based on user interests. In other words, IbR2 acknowledges the fact that user interactions may span across different interests and composes the recommended sequence of queries as follows:

1. the first query of the sequence is the chosen query of the observation that is the most similar to the current observation;
2. this observation is used as the new current observation for which the next interest is identified with a random draw using the Markov model, which means that the probability to reach another interest is low, not null, which is different from IbR1;
3. the most similar observation of the next probable state, according to the Markov Model, that has not been yet recommended, is identified and is added to the sequence;
4. this algorithm iterates until the final state of the Markov model is reached.

6. Testing user interest detection

This section presents the empirical evaluation of our approach for detecting user interests. It starts with the experimental protocol (Section 6.1) and exposes our results (Section 6.2).

Our first objective is to determine a metric based on the features introduced in Section 4.1 that allows, when paired with a clustering algorithm, the grouping of user observations into clusters that accurately reflect user interests. The main goal is to use these clusters for recommending queries that share the same user's interests. In this regard, the first experiments (Section 6.2.1 to Section 6.2.4) aim at determining and validating the best subset of features from the set presented in Table 4. We test its sensitivity to the clustering algorithm as well as its

behavior when confronted with observations or clusters of observations related to a business need. Then, a comparative experiment (Section 6.2.6) with the state-of-the-art similarity measure for OLAP sessions proposed in [9] shows the effectiveness of our proposal in the particular context of user interests discovery. Incidentally, our experiments also reveal that considering the reference metric [9] as a feature in our similarity measure in some cases improves the overall quality of our approach.

After that, we propose two side experiments to further validate our clustering approach, as follows: (i) the behavior of our metric when confronted with unseen business needs (Section 6.2.5), and (ii) the behavior of our metric in detecting intra-interaction interests (Section 6.2.7).

6.1. Experimental protocol

6.1.1. Data set

The data used for our experiments consist of navigation traces of 14 volunteers at SAP, covering a range of skills in data exploration, divided into two groups, namely, beginners and experts, based on their position in the company. To evaluate to what 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 an SAP prototype that supports keyword-based BI queries[1]. The business needs were grouped in different business cases, such as: *"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?"*. All business needs are listed in Appendix A. To be more realistic, business needs were defined to expect 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 5: Analysis of business needs

In total, our data set, named COMPLETE hereafter, contains 24 user interactions, each one possibly concerning several business needs, accounting for 530

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

queries. Table 5 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. To have several difficulty settings, we also built 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.

6.1.2. Assessing the user intents quality

Our objective is to build groups of observations that are only related to a single user interest. As in any clustering problem, there are two main solutions to assess the quality of the results, i.e., either based on some external knowledge of the ground truth clusters or based on some internal criterion evaluation. In our case, we evaluate both as follows.

Concerning the external evaluation, the main indicator of success in our case is the precision of the clustering when compared to the theoretical grouping of observations provided by the business questions. At a second level, recall allows for determining to which extent each cluster covers all of the observations related to a user interest. Finally, we use the classic Adjusted Rand Index (ARI) to evaluate the overall quality of the clustering. The values of this index range from below 0 (when the clustering performs badly and produces a partition close to a random clustering) and 1 (when the clustering is perfect) [26].

The internal evaluation comprises measuring the quality of the obtained clusters through the classic Silhouette coefficient. We observe, for each cluster, the intra similarity between its composing observations and the inter-cluster similarity, so that we can conclude on the compactness of the clusters. Being aware of possible observation overlapping between clusters, we should ensure that the distance between the observations within a cluster is shorter compared to the distance with other clusters' observations. The Silhouette coefficient is a way of validating this consistency of clusters. It ranges from -1 to 1, where the higher the positive value is, the better the observation is classified in its own cluster and the further it is from the other clusters. Negative values show that most of the observations of the clusters are nearer to another one rather than in the cluster in which they are categorized.

6.1.3. 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 4.2, on the sole basis of these features. The subsets of features are selected as follows. We consider all 15 features described in Table 4 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 the forming of subsets of features starting from those with only highly weighted features to subsets that more widely cover the whole set of features. We give the results for the following meaningful subsets: G2=$\{1, 3, 7, 8, 9\}$, G3=G2$\cup\{5, 10, 11, 13, 14\}$ and ALL, that include the features with the highest relative importance (the top-5, top-10 and all features, respectively). We also constitute a group G1=$\{7, 8, 9, 10, 13\}$ that includes the top-5 features selected by repetitively adding to the group those features that increase precision, similar to the work in [1]. 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).

6.1.4. Choice of the 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 [27], which is a k-medoid algorithm that finds hyperspherical clusters centered around the k most representative observations. We also use agglomerative hierarchical clustering algorithms [28] with single and complete linkage criterion to either allow for either elongated or compact clusters. Finally, we use the traditional DBSCAN algorithm [29] that is not restricted to a specific cluster shape but constraints clusters to share the same density of points.

6.1.5. Implementation

Our approach is implemented in Java. We also use Python Scikit Learn [30] 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.

6.2. Lessons learned

6.2.1. Determining the best subset of features and the clustering algorithm

Table 6 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 that allow for elongated clusters, such as hierarchical clustering with single link criterion and DBSCAN algorithms, 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 is because 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. Conversely, clustering algorithms that favor compact clusters, such as hierarchical clustering with complete link or 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 readability) do not exceed 10^{-2} and are usually 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 behavior based on the SVM weights observed on all our 15 features (see Section 6.1.3). Adding more features only slightly increases the recall. Other strategies (not mixing features from different specific groups or using the strategy proposed in [1]) can dramatically harm the 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 6: Clustering results with distinct subset of features on COMPLETE data set. For short, *Rec*, *Prec* and *ARI* denote respectively recall, precision and ARI scores.

6.2.2. G2 metric interpretation

Several features were proposed to describe the differences between the observations. The first two groups presented in Table 4 are inspired by the features used in the Web while the last one is a new proposal to show specific differences related to the BI context. The succeeding experiments concluded that a particular group of features presented in Table 7, named G2 in Section 6.1.3, principally composed of features related to business objects and formalized queries, in collaboration with PAM, identifies better the user interest.

The BI tool used for these experiments assisted juniors to reach the information needed, despite their non-precise questions in natural language, by proposing well formalized query suggestions. The user intent is expressed by the keywords (Tokens (Feature 1)) and the suggestions (Feature 3) proposed, but the real difference between user observations is specified by the chosen suggestion (Feature 9) with the query parts composing it (Feature 7) and their

G2 Features	Tokens	Suggestions	Query Parts	Token-Query Parts	Chosen
Weights	0.39	0.41	1.23	0.38	0.4

Table 7: Feature weights

matching tokens (Feature 8). Consequently, the features of G2 in Table 7 are the best selection that achieve identification of the user interests, as they directly represent the user choices. Adding more dimensions in the observations comparison reduces the accuracy of the measured dissimilarity between them, which leads to interests that are non well-defined.

6.2.3. G2 metric behavior

While our metric is learned from observations, our experimental protocol aims at grouping together observations participating in the analysis of a business need. To understand the behavior 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 8, when applied to analyses, our metric achieves optimal to very good performance. In the easiest case, when user interests are clearly distinct from each other 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. In contrast, learning metric weights on the basis of analyses (although not realistic) do not lead to good clusters of observations with significantly lower scores. Therefore, this experiment validates our choice of learning weights on observations and our choice of the G2 features.

		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 8: Behaviour of G2 set of features with PAM clustering when learning weights over observations or analyses. Column *"Weighting"* indicates whether weights are learned over observations or analyses.

	1	2	3	4	5	6	7	8	9	10
1	**0.716**	0.798	0.968	0.957	0.919	0.935	0.973	0.913	0.942	0.892
2		**0.722**	0.963	0.950	0.887	0.908	0.972	0.886	0.922	0.915
3			**0.869**	0.960	0.934	0.941	0.929	0.928	0.963	0.983
4				**0.764**	0.913	0.910	0.969	0.945	0.970	0.975
5					**0.763**	0.873	0.962	0.892	0.933	0.962
6						**0.781**	0.955	0.937	0.951	0.967
7							**0.834**	0.969	0.981	0.984
8								**0.701**	0.809	0.967
9									**0.691**	0.973
10										**0.634**

Table 9: Average dissimilarities between clusters.

	Cluster IDs									
	1	2	3	4	5	6	7	8	9	10
#observations	48	77	53	74	58	60	78	34	30	18
Silhouette	0.09	0.08	0.01	0.14	0.09	0.08	0.1	0.13	0.15	0.27
Diameter	0.71	0.72	0.86	0.76	0.76	0.78	0.83	0.70	0.69	0.63

Table 10: Silhouette coefficients and diameter for 10 clusters obtained with PAM using G2 based dissimilarities.

6.2.4. G2 metric robustness

The choice of the clustering algorithm and the features to learn the similarity between observations is decisive, in the sense that they have to fit the data that we collected. The main goal is to separate observations in compact clusters, as distant as possible from each-other to identify clear user intents. As shown in Table 9, the intra clusters dissimilarities, presented in the diagonal, are lower than inter clusters dissimilarities. While these differences are not significant because of high standard deviation (not reported for the sake of readability), this result is confirmed by the Silhouette coefficient [31]. This coefficient is positive for all the clusters, as presented in Table 10, verifying that the built clusters are cohesive and the majority of observations in each of them are well classified in their own clusters. In view of the business questions corresponding to the cluster interests, we notice a higher compactness for the groups of observations responding to well separated questions, as in clusters 9 and 10. As expected, increasing the number of finer interests we want to discover, i.e., augmenting the number of clusters, results in more compact groups of observations. For instance, for 50 clusters, better Silhouette coefficients are obtained for most of the clusters, reaching the highest value of 0.5, with the notable exception of two small clusters (having 6 and 7 observations, respectively) that manifest a negative coefficient, close to 0, showing the inconsistency of the observations composing these clusters. Regarding the diameter, it is well balanced among the discovered clusters, with some minor differences being explained by outliers.

6.2.5. Handling unseen business needs

In this experiment, we study how our method handles previously unseen business needs and how general the metric learned on the G2 features is. 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. The results in Table 11 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 12. 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, which has more relevant queries than Q10 contained in the REDUCED 1 data set (see Table 5).

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

6.2.6. Comparative experiments

Table 12 shows how our metric compares to a reference metric that Aligon et al. [9] designed for OLAP queries. This metric has been validated by user tests that showed its effectiveness in grouping queries in accordance with what a human expert would have done. Table 12 reveals 2 distinct behaviors 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 the 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 solution that compromises over 15 dimensions for ALL features rather than 5 in the case of G2 features, and with only a few training instances. In contrast, 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 in finding a better solution by fine tuning the separation hyper plane. Consequently, in this case, slightly better results may be achieved with features other than G2's.

If we dig into the details of the features involved in each result, it can be seen that good results are achieved when the weights of the features related to queryParts, tokenQPart or Chosen are important. For the COMPLETE data set, this is observed for G2 features that reach the best overall clustering results. In the case of ALL features, queryParts is slightly below, which may explain the small difference in performance with G2 features. Adding the similarity

	Complete 10 clusters			Reduced 1 4 clusters		
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 [9]	0.39	0.20	0.14	0.41	0.33	0.10
ALL + [9]	0.40	0.40	0.32	**0.78**	**0.65**	**0.63**
G2 + [9]	0.45	0.43	0.38	0.69	0.62	0.52

Table 12: Comparison of our metric based on G2 features with other metrics when paired with PAM clustering. ALL denotes the set of 15 features, [9] is the state-of-art metric and "+" indicates a metric with added features and corresponding weights.

proposed by Aligon et al. [9] to G2 or ALL decreases the weights of queryParts or tokenQPart, which in turn degrades the results. These observations are coherent with what is observed for the REDUCED 1 data set. In this case, adding the similarity proposed by Aligon et al. [9] to G2 only slightly lowers the score on tokenQPart and queryPart, which again may explain the small difference observed. Finally, with ALL features, it can be seen that adding the similarity proposed by Aligon et al. [9] causes a drop in the previous important features, but that is compensated by the emergency of new features, such as the OLAP operation. 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 in practice. Thus, the metric based on G2 features seems to be the most appropriate among those that we evaluated, in particular when compared to the state-of-the-art metric [9].

6.2.7. Discovering intra-interaction interests

In this test, we successively increase the number of clusters and we check 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 in the resolution of a particular business need but also at identifying parts of the resolution that are shared by users with different expertise. To emphasize the evolution of precision (which indicates the coherence of clusters), we use the (G2 + [9]) configuration, which is a good compromise in the previous experiment, and test it on the well separated REDUCED 1 data set, starting with 10 clusters. The results reported in Table 13 show how the mixing of users decreases while the 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 expertise 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%)	10 (100 %)
15	0.24	0.90	0.31	14 (93.3%)	14 (93.33 %)
20	0.20	0.92	0.26	14 (70%)	18 (90 %)
25	0.18	0.92	0.24	13 (52%)	19 (76 %)
30	0.17	0.95	0.23	13 (43.3%)	19 (63.33 %)
35	0.16	0.95	0.22	12 (34.3%)	19 (54.29 %)
50	0.14	0.96	0.19	11 (22%)	20 (40 %)

Table 13: 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).

7. Testing the interest-based recommender system

Finally, we take advantage of the identified user interests to recommend a sequence of observations to users, to help them continue their explorations. We experiment with the two recommendation modes that use the Markov model over the clusters (introduced in Section 5), respectively called *IbR1* and *IbR2*, and we compare them with two state-of-the-art query recommender systems: the one proposed by Aligon et al. [5] and one of the recommenders proposed by Eirinaki et al. [6], adapted to recommend a sequence of queries instead of a set of queries (Section 7.1.3). We first present our protocol (Section 7.1), and then the experimentation results (Section 7.2).

7.1. Experimental protocol

7.1.1. Recommender system construction and analysis

Our recommender system is built as a Markov model over the interactions from which observations have been clustered to identify user interests. Consistent with the protocol proposed in [8], we remove from the set of interactions the ones consisting of only one observation. For the constructed Markov model, we report the transition probabilities between clusters and check if, as expected for consistent user interest, the highest probabilities are for transitions leading from one cluster to itself.

7.1.2. Evaluation of recommendation results

To evaluate our approach, we rely on the literature on recommender systems [32, 33, 34] as well as on a recent protocol specially conceived for comparing recommendations of query sequences [5]. We measure two of the most commonly employed criteria to judge the recommendation quality to assess whether our recommender is able to achieve a good balance between the ability to recommend and the quality of its recommendations, namely:

- accuracy, i.e., the degree to which recommendations correspond to what is expected in terms of queries, and

- coverage, i.e., the degree to which recommendations can indeed be generated.

We enrich this set of measures with the following criteria to understand whether our recommender system favors expertise sharing between users and interest diversity:

- expected diversity, i.e., the degree to which recommendations correspond to what is expected in terms of user interests,
- expected user, i.e., the degree to which the current user is retrieved in the recommendations,
- expertise, i.e., the degree to which recommendations come from experts, and
- expertise benefit, i.e., the degree to which beginners can benefit from expert recommendations.

Regarding accuracy, the protocol acknowledges the fact that finding the exact next query of an interaction is very unlikely, as our set of interactions consists of mostly unique observations. The protocol therefore implements extended versions of precision and recall measures to incorporate similarity between interactions.

For our tests, we use the similarity between sessions defined and proposed in [9], that is itself based on the similarity between queries. In our adaptation we assimilate interactions to sessions and we use the similarity between queries to compare the chosen queries of interactions to recommended queries. We use this similarity measure because it is independent from our proposal and can fit any recommender system under testing, contrary to ours, which needs proper interactions to work.

Definitions. The underlying idea is, given the beginning of an interaction, to compare the recommended queries with the unseen queries (that actually continued the interaction) and calculate the precision, recall and F-measure. In what follows, depending on the criteria considered, we use $\sim$ to denote a similarity function between sequences of queries, which is instantiated differently in each measure (accuracy, expertise, expected diversity, expected user).

Let I be a set of interactions and I_C be a set of current interactions for which recommendations are to be computed. Given an interaction $i \in I_C$, let f_i be its actual future (i.e., the sequence of queries the user would have formulated after the last query of i if they had not been given any recommendation) and r_i be a recommended future. Recommendation r_i is considered to be *correct* when $r_i \sim f_i$, i.e., when it is similar to the actual future of i.

Let $FI = \{f_i | i \in I_C\}$ and $RI = \{r_i | i \in I_C\}$. The set of true positives is then defined by

$$TP = \{r_i | i \in I_C \; ; \; r_i \sim f_i\} \tag{2}$$

i.e., the set of recommended futures similar to their actual counterparts. The set of false positives is $FP = RI \setminus TP$ and the set of false negatives is $FN = FI \setminus TP$. Then, $Recall = \frac{|TP|}{|TP|+|FN|} = \frac{|TP|}{|FI|}$, $Precision = \frac{|TP|}{|TP|+|FP|} = \frac{|TP|}{|RI|}$ and $F\text{-}measure = 2\frac{Precision \cdot Recall}{Precision+Recall}$.

Let RQ bet the set of queries in RI, FQ be the set of queries in FI, TQ be the set of queries in TP, I_E be the set of interactions written by expert users, Q_E be the set of queries of I_E, Q_{U_j} be the set of queries of interactions of interest U_j for $j \in \{1, c\}$, where c is the number of discovered interests, and Q_{user_k} be the set of queries of interactions of user k, for $k \in \{1, |S_u|\}$, where S_u is the set of users.

Our measures are defined as follows:

- *accuracy* is measured using the *F-measure*, where $\sim$ is the similarity between sessions proposed in [9],
- *coverage* is the number of recommendations divided by the number of interactions: $\frac{|RI|}{|I_C|}$,
- *expected diversity* is measured using F-measure, where $\sim$ is the Jaccard similarity on interests of interactions, defined by:
  $r_i \sim f_i = \frac{|\{Q_{U_j} | \exists q \in Q_{U_j} \cap r_i\} \cap \{Q_{U_j} | \exists q \in Q_{U_j} \cap f_i\}|}{|\{Q_{U_j} | \exists q \in Q_{U_j} \cap r_i\} \cup \{Q_{U_j} | \exists q \in Q_{U_j} \cap f_i\}|}$,
- *expected user* is measured using the F-measure, where $\sim$ is the Jaccard similarity of users participating in interactions. Note that observations in f_i correspond to a unique user (the actual user), while recommended observations in r_i may come from several users.
  $r_i \sim f_i = \frac{|\{Q_{user_k} | \exists q \in Q_{user_k} \cap r_i\} \cap \{Q_{user_k} | \exists q \in Q_{user_k} \cap f_i\}|}{|\{Q_{user_k} | \exists q \in Q_{user_k} \cap r_i\} \cup \{Q_{user_k} | \exists q \in Q_{user_k} \cap f_i\}|}$,
- *expertise* is measured with the F-measure, but for a version of precision and recall that incorporate expert queries, as follows: precision is $\frac{|TQ \cap Q_E|}{|TQ|}$, recall is $\frac{|TQ \cap Q_E|}{|Q_E|}$,
- *expertise benefit* is measured as the probability to recommend a query made by an expert to a session made by a beginner:
  $P(q \in Q_E \cap r_i | i \in I_C \setminus I_E)$.

Protocol. To create sets FI, RI and TP, our protocol uses cross-validation as follows. We iterate over a set L of interactions with a leave-one-out approach by (i) picking one interaction $i \in L$; (ii) taking one of its prefix i_n of size n as one current interaction of i and the remaining subsequencef_i as one actual future of FI, with $n \in \{1, |i|\}$; (iii) finding a recommendation r_i for i_n using the remaining interactions, $L \setminus \{i\}$. If such a r_i exists, it is added to RI. r_i is considered correct and added to TP when $r_i \sim f_i$ and is incorrect otherwise. For accuracy and expertise, the similarity between interactions is parametrized by a threshold varying in [0,0.9]. This threshold controls the extent to which

two interactions should be considered similar. Hence these criteria are measured by progressively increasing this threshold ruling the minimal demanded similarity between the expected future and the recommendation. The same is done to compare systems in terms of expected diversity and expected user, with a threshold ruling the similarity of sets of recommended queries in terms of interests and users, respectively.

7.1.3. Comparison with state-of-the-art recommendation algorithms

In order to illustrate the added value of user interest detection, as well as to show that our approach is agnostic of the query language used, we compare it with two state-of-the-art, session-based recommendation algorithms, both based on collaborative filtering and kNN. These approaches are i) an OLAP session recommendation approach [5, 35], referred to as Falseto in what follows, and ii) a SQL query recommendation approach, QueRIE [6]. Precisely:

1. **QueRIE**. We use the fragment-based, non-binary version of QueRIE, which handles queries (both in the log and in the current session) represented by their SQL fragments, i.e., projected attributes (levels and measures), selections, and group-by expressions, that are easily extracted from the query-parts. This representation of user sessions based on query fragments is called the signature. The recommended queries are selected from the closest session to the current one based on their respective signatures. As the number of queries to recommend is a parameter in QueRIE, we tuned it and selected the value of the parameter that achieved the best accuracy, precisely 2 queries. Finally, the set of recommended queries is ranked using the similarity to the current session and arranged in a sequence as was done for IbR1. Note that this transformation of QueRIE output is necessary to ensure that it is comparable to the other recommenders and is under the same conditions.
2. **Falseto**. In order to use Falseto, an OLAP schema (a constellation) was reverse engineered from data sources (schemata and constraints) following DFM methodology [21], pruning attributes not accessed by users (based on the query log). Interfacing with Falseto is straightforward, as we use the same format for representing queries. However, it should be noted that different from IbR1, IbR2 and QueRIE, Falseto does not directly recommend queries that are simply picked in a log file of past queries. Indeed, it picks queries from a log file and then modifies these queries to align them with the current interaction. Therefore, to measure the diversity and expertise related criteria, we disregarded that alignment and looked at the original queries picked in the log. We expect this recommender system to explore more globally the space of possible queries as it builds new queries (not necessarily existing in the logs) based on current queries.

We use our own implementation of QueRIE and Falseto.

7.1.4. Impact of user interests on the recommendation strategy

Finally, we aim at investigating whether leveraging user interests calls for a tailored recommendation strategy or can benefit an existing one. To this end, we give Falseto and QueRIE, which are agnostic of user interests, the chance of knowing the user interest beforehand by restricting their input to one particular user interest. We repeat this test for each discovered user interest. We report their accuracy and coverage in each restricted log and compare them to their own results on the whole log.

7.2. Lessons learned

7.2.1. Recommender system construction and analysis

The Markov model at the heart of our recommender is built from a set of 24 interactions corresponding to 530 unique observations. We removed 17% of the interactions from it, which contain only one observation and do not provide any information to the Markov model.

Table 14 presents the transition probabilities between clusters (states), sources in rows and targets in columns. Note that, as this model is recreated several times in our tests, we present here the model learned over the whole log. It is easily perceived that, as expected, observations of a cluster are mainly followed by observations of the same cluster, meaning that interactions tend to remain within the same user interest.

	State 1	**State 2**	**State 3**	**State 4**	**State 5**	**State 6**	**State 7**	**State 8**	**State 9**	**State 10**	**Final State**
State 1	**0.40**	0.13	0.06	0.0	0.08	0.13	0.02	0.04	0.0	0.0	0.14
State 2	0.10	**0.58**	0.06	0.04	0.05	0.04	0.03	0.03	0.01	0.04	0.02
State 3	0.0	0.06	**0.43**	0.02	0.09	0.13	0.06	0.0	0.13	0.04	0.04
State 4	0.0	0.03	0.05	**0.68**	0.05	0.03	0.05	0.0	0.05	0.02	0.04
State 5	0.02	0.07	0.02	0.05	**0.54**	0.03	0.15	0.10	0.02	0.0	0.0
State 6	0.02	0.17	0.02	0.08	0.07	**0.46**	0.15	0.03	0.0	0.0	0.0
State 7	0.04	0.0	0.10	0.14	0.0	0.1	**0.55**	0.03	0.0	0.0	0.03
State 8	0.03	0.03	0.08	0.0	0.06	0.03	0.15	**0.50**	0.06	0.0	0.06
State 9	0.17	0.03	0.03	0.0	0.03	0.07	0.03	0.10	**0.50**	0.0	0.04
State 10	0.11	0.06	0.11	0.0	0.0	0.06	0.0	0.0	0.0	**0.39**	0.27

Table 14: Transition probabilities for the 10 states (clusters) of the recommender's Markov model

7.2.2. Evaluation of recommendation results

Figure 1, 2 and 3 report the measures of the various criteria defined to assess the quality of the recommenders. We start by discussing these measures for IbR1 and IbR2. The coverage is as expected. By design, IbR1 achieves a perfect coverage, while IbR2, using a probability for ending the session, may not recommend, particularly for a longer current session. Regarding accuracy, both recommenders perform very well, with IbR1 performing the best, when the similarity threshold is set low (0.4 or below). Below this threshold, both recommenders show the same behavior. In terms of expected diversity, as expected, IbR2 outperforms IbR1 since the latter cannot move outside a current interest. Notably, even for quite demanding similarity thresholds, IbR2 performs reasonably well in predicting interest switches. The very low scores for both

recommenders in terms of the expected user is expected in that it confirms that none of them were designed to stick to the current user. Nevertheless, we note that both IbR1 and IbR2 still do better than state-of-the-art recommenders for low similarity thresholds, which can be interpreted as a side effect of user interest detection. Both recommenders perform well in terms of expertise, with IbR2 being more robust than IbR1 to the similarity threshold. This is due to IbR2 being more likely to find expert queries in clusters other than the current one. Finally, both recommenders perform fairly in recommending expert queries to beginners. We note that they were not designed to do so and good performances for this criterion would have been a side effect of clustering interests. However, extending the recommenders to favor this behavior can be done easily if expertise is recorded or can be deduced from the observations. In summary, IbR1 performs slightly better in terms of accuracy and coverage, while IbR2, with its global exploration of the user interests, is better at identifying interest switching and proposing recommendations coming from expert users.

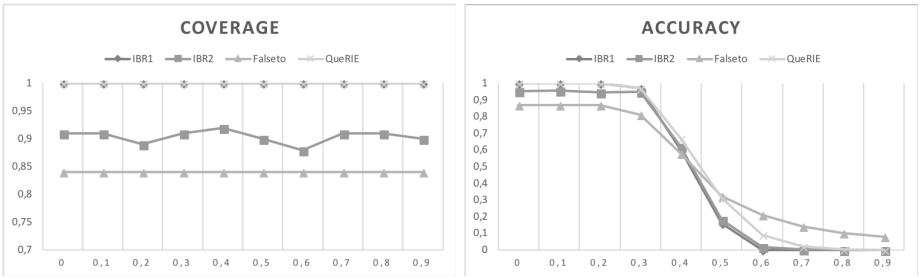

Figure 1: Coverage and accuracy for IbR1, IbR2, Falseto and QueRIE.

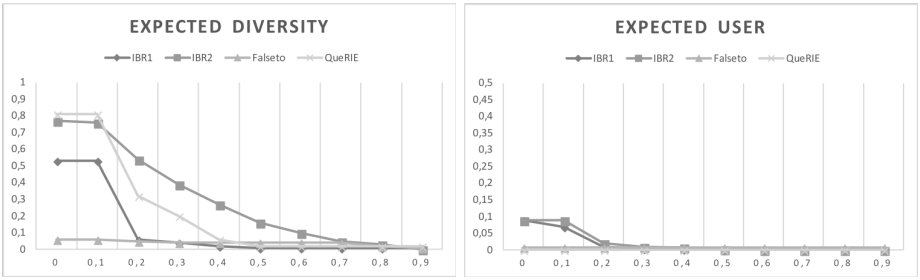

Figure 2: Expected diversity and expected user for IbR1, IbR2, Falseto and QueRIE.

7.2.3. Comparison with state-of-the-art algorithms

We now discuss how IbR1 and IbR2 compare to two state-of-the-art recommenders.

Comparison with QueRIE. QueRIE achieves perfect coverage, as does IbR1, and it is similar to it in terms of accuracy for low similarity thresholds, being

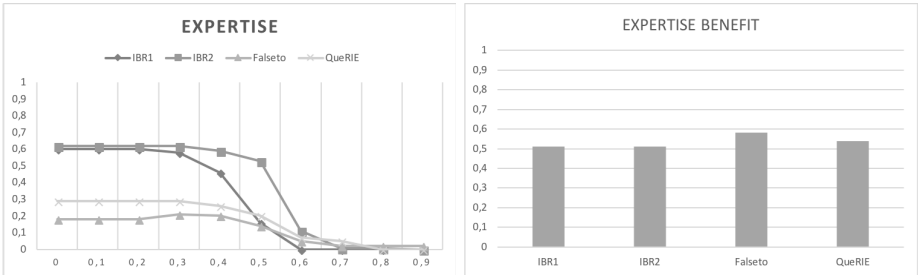

Figure 3: Expertise and expertise benefit for IbR1, IbR2, Falseto and QueRIE.

slightly more robust to more demanding thresholds. This similarity of behavior can be explained by the nature of both recommender systems, which are very similar as they tend to locally explore the user intent based on current queries. Interestingly, in this case, the difference in the similarities used to rank potential recommended queries in QueRIE and IbR1 (fragment-based versus feature-based) has no influence since potential queries to recommend are already issued from the same user interest. We note that QueRIE is always better than IbR1 for expected diversity, since the latter is bound to a specific interest, and is slightly better than IbR2 for very low similarity thresholds but is expectedly less robust than it to high thresholds. Finally, as expected, its results in terms of expected user, expertise or expertise benefit show that it has not been specifically designed to take these features into account.

Comparison with Falseto. Among all the recommenders, Falseto achieves the worst performances both in terms of coverage and accuracy for low similarity thresholds. Regarding coverage, it is clearly impacted by the demanding session similarity measure that Falseto internally uses to align current and past sessions to generate candidate recommendations. Indeed, when query similarity is below Falseto's built-in threshold, no past session is found to be similar to the current one, which results in no candidate recommendations, which disables the recommendation. Remarkably, Falseto is more robust in terms of accuracy when the similarity threshold becomes more demanding. This can be explained by its fitting phase, which aligns the recommendations with the current interaction, i.e., even if the candidate recommendation picked from the log is not the one expected, the fitting phase is able to sufficiently modify it to bring it closer to the expected future. As expected, Falseto is outperformed in terms of expected diversity, expected user, and expertise, but surprisingly achieves the best expertise benefit. This can be because its candidate recommendations are sequences that are similar to others in the log and that such sequences are more likely produced by expert users.

7.2.4. Impact of user interests on the recommendation strategy

In this last test, we observe the behavior of Falseto and QueRIE in the presence of discovered user interests. More precisely, we force them to recommend

queries inside each cluster separately and to simulate their behavior if they were not agnostic of user intent. The goal of these tests is to compare the accuracy and coverage of the recommender over a user interest, with itself on the whole log. Note that we do not intend to investigate which cluster achieves better performance, so the identification of individual curves is irrelevant. The results are reported in Figures 4 and 5 where the test is done for all of the 10 detected user interests, with the dark square curve representing the recommender over the whole log. Coverage for QueRIE is not depicted because QueRIE achieves perfect coverage in all cases.

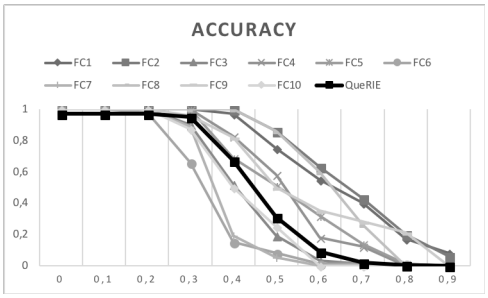

Figure 4: Accuracy for QueRIE on the entire log and on each user interest

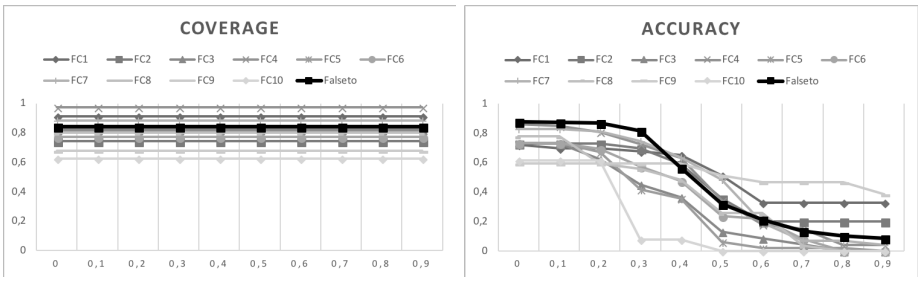

Figure 5: Coverage and accuracy for Falseto on the entire log and on each user interest

The results show that detecting user interest may be useful for already existing recommendation strategies. This is particularly clear for QueRIE, which performs better in the majority of cases when the interest is leveraged. QueRIE benefits from targeting its recommendation in a specific context, which increases the similarity between recommendations and the actual future that is mostly contained in a single user interest. Notably, Clusters 6 and 10 are those for which QueRIE obtains its worst results. These are less homogeneous clusters (see Table 9), leading to the observed decrease in accuracy.

This is more contrasted for Falseto, where in three cases only leveraging user interests makes the recommendations more accurate, interestingly, for high similarity thresholds. Due to its fitting phase, Falseto is more likely to be sensitive to cluster overlapping, and thus more likely to deviate to other neighboring

interests. Therefore, when only one cluster is available, Falseto will miss those sessions spanning different clusters. Similarly, restricting the past history to a single user interest can decrease the coverage of Falseto because of the size of the clusters representing each user interest and because the intra-cluster similarity is not on par with the session-based similarity used by Falseto.

8. Conclusion

This article presents a collaborative recommendation approach that leverages user interests in modern BI systems to relieve the user from tedious explorations. This system combines state-of-the-art techniques from literature in Web Search and BI query recommendation. At the heart of it is an approach for identifying coherent interests of BI users with various expertise querying data sources 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. Once user interests are identified, they are treated as first-class citizens in a collaborative BI query recommender system, that suggest next moves in an exploration based on the probability for a user to switch from one interest to another.

We have shown through user tests that our approach is effective in practice and can be beneficial to analysts whose interests match those of expert users, or whose interests change during the analysis. Overall, our results show that keyword-based interaction systems provide semantically rich user traces well adapted to the detection of coherent BI user interest and that such interests can also be exploited successfully by state-of-the-art recommendation strategies.

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 data sources 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] R. Guha, V. Gupta, V. Raghunathan, R. Srikant, User modeling for a personal assistant, in: WSDM, Shanghai, China., 2015, pp. 275–284.

[2] Y. Song, Q. Guo, Query-less: Predicting task repetition for nextgen proactive search and recommendation engines, in: WWW, 2016, pp. 543–553.

[3] L. Yang, Q. Guo, Y. Song, S. Meng, M. Shokouhi, K. McDonald, W. B. Croft, Modeling user interests for zero-query ranking, in: ECIR, 2016, pp. 171–184.

[4] K. Drushku, J. Aligon, N. Labroche, P. Marcel, V. Peralta, B. Dumant, User interests clustering in business intelligence interactions, in: Advanced

Information Systems Engineering - 29th International Conference, CAiSE 2017, 2017, pp. 144–158.

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

[6] M. Eirinaki, S. Abraham, N. Polyzotis, N. Shaikh, Querie: Collaborative database exploration, IEEE Trans. Knowl. Data Eng. 26 (7) (2014) 1778–1790.

[7] C. Sapia, PROMISE: predicting query behavior to enable predictive caching strategies for OLAP systems, in: DaWaK, 2000, pp. 224–233.

[8] M. Aufaure, N. Kuchmann-Beauger, P. Marcel, S. Rizzi, Y. Vanrompay, Predicting your next OLAP query based on recent analytical sessions, in: DaWaK, 2013, pp. 134–145.

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

[10] B. Mobasher, Data mining for personalization, in: The Adaptive Web: Methods and Strategies of Web Personalization, Vol. 4321 of LNCS, 2006, pp. 90–135.

[11] Y. Sun, N. J. Yuan, Y. Wang, X. Xie, K. McDonald, R. Zhang, Contextual intent tracking for personal assistants, in: SIGKDD, 2016, pp. 273–282.

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

[13] N. Khoussainova, Y. Kwon, M. Balazinska, D. Suciu, Snipsuggest: Context-aware autocompletion for SQL, PVLDB 4 (1) (2010) 22–33.

[14] P. Jayachandran, K. Tunga, N. Kamat, A. Nandi, Combining user interaction, speculative query execution and sampling in the DICE system, PVLDB 7 (13) (2014) 1697–1700.

[15] H. V. Nguyen, al., Identifying user interests within the data space - a case study with skyserver, in: EDBT, 2015, pp. 641–652.

[16] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 734–749. doi:10.1109/TKDE.2005.99.
URL https://doi.org/10.1109/TKDE.2005.99

[17] R. D. Burke, A. Felfernig, M. H. Göker, Recommender systems: An overview, AI Magazine 32 (3) (2011) 13–18.
URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/2361

[18] I. Cantador, A. Bellogín, P. Castells, Ontology-based personalised and context-aware recommendations of news items, in: 2008 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2008, 9-12 December 2008, Sydney, NSW, Australia, Main Conference Proceedings, 2008, pp. 562–565.
URL https://doi.org/10.1109/WIIAT.2008.204

[19] R. Ghani, A. E. Fano, Using text mining to infer semantic attributes for retail data mining, in: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan, 2002, pp. 195–202.

[20] C. Palmisano, A. Tuzhilin, M. Gorgoglione, Using context to improve predictive modeling of customers in personalization applications, IEEE Trans. Knowl. Data Eng. 20 (11) (2008) 1535–1549. doi:10.1109/TKDE.2008.110.
URL https://doi.org/10.1109/TKDE.2008.110

[21] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill, 2009.

[22] A. A. Vaisman, E. Zimányi, Data Warehouse Systems - Design and Implementation, Data-Centric Systems and Applications, Springer, 2014.

[23] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
URL http://webdam.inria.fr/Alice/

[24] J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, E. Turricchia, Mining preferences from OLAP query logs for proactive personalization, in: ADBIS, 2011, pp. 84–97.

[25] H. Wang, Y. Song, M.-W. Chang, X. He, R. W. White, W. Chu, Learning to extract cross-session search tasks, in: In WWW, 2013.

[26] B. Desgraupes, Clustering indices, Tech. rep., University Paris Ouest - Lab Modal'X (April 2013).

[27] L. Kaufman, P. Rousseeuw, Clustering by means of medoids, in: Statistical Data Analysis based on the L1 Norm, Elsevier, 1987, pp. 405–416.

[28] L. Kaufman, P. Rousseeuw, Finding groups in Data: An introduction to Cluster Analysis, Wiley, 1990.

[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, AAAI Press, USA, 1996, pp. 226–231.

[30] F. Pedregosa, al., Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[31] P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Computational and Applied Mathematics 20 (1987) 5365.

[32] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53. doi:10.1145/963770.963772.
URL http://doi.acm.org/10.1145/963770.963772

[33] R. A. Baeza-Yates, B. A. Ribeiro-Neto, Modern Information Retrieval - the concepts and technology behind search, Second edition, Pearson Education Ltd., Harlow, England, 2011.
URL http://www.mir2ed.org/

[34] A. Gunawardana, G. Shani, Evaluating recommender systems, in: Recommender Systems Handbook, 2015, pp. 265–308.

[35] J. Aligon, K. Boulil, P. Marcel, V. Peralta, A holistic approach to OLAP sessions composition: The falseto experience, in: DOLAP, 2014, pp. 37–46.

Appendix A. Business needs

Hereafter, we list the business needs that were proposed to SAP analysts during our experiments.

1. Plan what kinds of films should Spain media shops aim to sell this year and in which media format? According to your prediction, calculate a target sales revenue.
2. For each European country, detect which genres of films did not reach the expecting sales.
3. For which Medias and Genres Iceland has not reached the total sales target? What should they change in their selling politics, knowing their preferences and top retailers?
4. For each region, and for each month (from January to June), which should be the priority country for Decathlon to be supplied with goods?
5. Which retailer has the best performance (sales revenue/target sales revenue) in 2013 and what is its growth for 2014 and 2015? Which were the most successful colours in these three last years?
6. Which are the three most problematic countries running out of stocks? Which are the retailers and products they should be supplied?
7. Which airports have the most increasing number of passengers from 1990-2010? Analyze the main airports with the greatest number of passengers (2010) for each region to find out if this is related to the increasing number of destination/population.
8. Which is the country with the greatest number of airports? How many passengers have flown from this country in 2005 and in 2010? Is this in proportional with the number of destinations from each airport?
9. For at least one country of low / lower middle / upper middle income countries, analyze how the GDP has evaluated.
10. In which Income Group would you classify a candidate country with a GDP of $6 billion?