Alma Mater Studiorum - Università di Bologna

Dottorato di ricerca in Computer Science and Engineering

Ciclo XXIX

Settore concorsuale di afferenza: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
Settore scientifico disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Business Intelligence on Non-Conventional Data

Enrico Gallinucci

Coordinatore Dottorato	Relatore
Prof. Paolo Ciaccia	Prof. Stefano Rizzi

Esame finale 2017

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof. Stefano Rizzi and to my advisor Prof. Matteo Golfarelli for their continuous support, patience and motivation during these three research years. Their mentoring has provided an invaluable contribution to both my professional and personal growth.

My sincere thanks also goes to Prof. Alberto Abelló and Prof. Oscar Romero for their collaboration and support during my research period in Spain at Universitat Politècnica de Catalunya, Barcelona.

I would like to thank Prof. Enrico Denti, Prof. Claudio Sartori, Prof. Robert Wrembel and Prof. Esteban Zimanyi for reviewing this thesis.

I wish to thank my dear colleagues in Cesena and Barcelona for the effective teamwork, the stimulating discussions and the strong camaraderie.

Finally, I would like to thank my family and my closest friends for making this journey easier with their unquestioned love and support.

Abstract

The revolution in digital communications witnessed over the last decade had a significant impact on the world of Business Intelligence (BI). In the big data era, the amount and diversity of data that can be collected and analyzed for the decision-making process transcends the restricted and structured set of internal data that BI systems are conventionally limited to. This thesis investigates the unique challenges imposed by three specific categories of non-conventional data: social data, linked data and schemaless data. Social data comprises the user-generated contents published through websites and social media, which can provide a fresh and timely perception about people's tastes and opinions. In Social BI (SBI), the analysis focuses on topics, meant as specific concepts of interest within the subject area. In this context, this thesis proposes meta-star, an alternative strategy to the traditional star-schema for modeling hierarchies of topics to enable OLAP analyses. The thesis also presents an architectural framework of a real SBI project and a cross-disciplinary benchmark for SBI. Linked data employ the Resource Description Framework (RDF) to provide a public network of interlinked, structured, cross-domain knowledge. In this context, this thesis proposes an interactive and collaborative approach to build aggregation hierarchies from linked data. Schemaless data refers to the storage of data in NoSQL databases that do not force a predefined schema, but let database instances embed their own local schemata. In this context, this thesis proposes an approach to determine the schema profile of a document-based database; the goal is to facilitate users in a schema-on-read analysis process by understanding the rules that drove the usage of the different schemata. A final and complementary contribution of this thesis is an innovative technique in the field of recommendation systems to overcome user disorientation in the analysis of a large and heterogeneous wealth of data.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Business Intelligence

The term Business Intelligence (BI) indicates a wide range of IT applications whose goal is to help managers in the decision making process by collecting, managing and analyzing data. In particular, the goal in BI is to provide corporate decision makers with software solutions that help them identify and understand the key business factors, so as to make the best decisions for the situation at the time [63]. The *BI pyramid* shown in Figure 1.1 describes the progressive extraction of knowledge from the raw data of a company, as well as the roles played by different technologies for the decision-making process. The enabling components at the core of a BI platform are a data warehouse (DW), i.e., a repository that stores information according to a multidimensional schema, and OLAP (On Line Analytical Processing) interfaces, which allow users with limited IT skills to explore the data and gain valuable information. OLAP analyses are traditionally carried out on the data owned by the companies, which are typically well-structured in accordance with the relational model.

The revolution in digital communications witnessed over the last decade had a significant impact on the world of BI. For starters, the term *BI 2.0* has been coined to refer to the newest wave of tools and software for BI that propose a dynamic, collaborative and web-based approach to BI [125]. Most importantly, the amount and diversity of data that can be collected and analyzed for the decision-making process has increased exponentially. It is widely common to use the term *big data* to refer to those data that impose important challenges from (at least) one the following four perspectives: volume (when the required storage capacity surpasses the one of common machines), velocity (when data is generated continuously and at a higher rate than it can be collected), variety (when contents are either unstructured or structured in

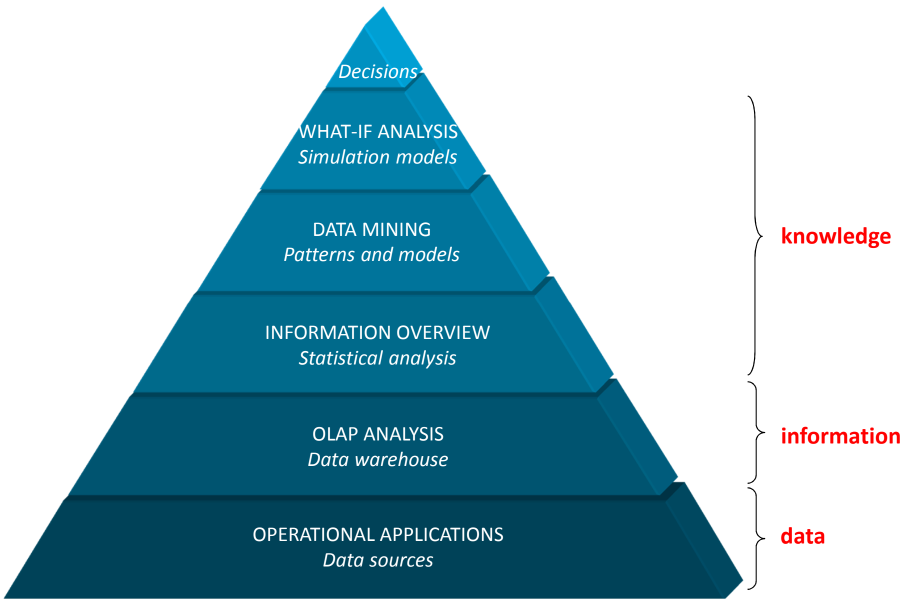


Fig. 1.1 The business intelligence pyramid.

vary different ways), veracity (when the quality and trustworthiness of data can be questioned). This is also commonly known as the *rule of the four Vs* [77]. Nonetheless, big data have a rather generic definition and the term has been overly used (if not abused) in the literature. For the purpose of this thesis, we introduce a more detailed classification —based on the characteristics of those data — that helps us to outline the unique challenges imposed by each category and the roles respectively played in the BI process.

- **Social data**. The planetary success of social networks and the widespread diffusion of portable devices has enabled simplified and ubiquitous forms of communication and has contributed, during the last decade, to a significant shift in human communication patterns towards the *voluntary sharing of personal information*. Everyone is able to connect to the Internet anywhere and anytime, and to continuously send messages to a virtual community centered around blogs, forums, social networks, and the like. This has resulted in the accumulation of enormous amounts of *user-generated content* (UGC), that include geolocation, preferences, opinions, news, etc. This huge wealth of information about people's tastes, thoughts, and actions is obviously raising an increasing interest from decision makers because it can give them a fresh and timely perception of the market mood; besides, the diffusion of UGC is so widespread that it can directly influence the phenomena of business and society in a decisive way [21, 146, 181].

- **Linked data**. The World Wide Web has radically altered the way we share knowledge by lowering the barrier to publishing and accessing documents as part of a global *information* space, where hypertext links allow users to traverse this information space using web browsers. *Linked data* have extended this concept from documents to data: by employing the Resource Description Framework (RDF) and the Hypertext Transfer Protocol (HTTP), structured data are published on the web and linked to other data in different data sources. [13] The adoption of linked data has created a valuable network of structured and interlinked data that bears knowledge from the most diverse domains (from books, music and movies, to scientific publications, clinical trials and statistical data) and that has enabled a wide set of applications. For instance — besides simple browsing and searching activities — linked data can be a powerful tool in the hands of the decision makers to integrate and extend the corporate knowledge on a specific domain [1].

- **Schemaless data**. Recent years have witnessed the progressive erosion of the relational DBMS predominance to the benefit of DBMSs based on different representation models (e.g., document-oriented and key-value). Most new models adopt a *schemaless* representation for data, although this does not mean that data are stored *without* a schema: it rather means that schema is a soft concept and that the instances referring to the same concept in the same collection can be stored using different "local" schemata, in order to better fit the specific features of each instance. As they allow designers to easily change and handle different data structures, schemaless databases (often referred to as *NoSQL* databases) have become the preferred way to store heterogeneous data with variant schemata and structural forms — eventually leading to the creation of so-called *data lakes* [176]. From the perspective of a decision maker, data lakes conveniently fit the innovative data analysis strategy called *schema-on-read*: unlike the *schema-on-write* strategy, which forces a schema on the data when it is written to the database (i.e., the typical strategy in a relation database), schema-on-read applies a schema when data is read from the database. This grants a high flexibility to decision makers, as they can read and interpret the data according to their specific requirements at the moment of the analysis [31].

Noticeably, this kind of information can provide insights of crucial — if not fundamental — importance for decision makers. We call them *non-conventional data*, as opposed to the restricted and structured set of internal data that BI systems are conventionally limited to.

1.2 Motivations and Contributions

The contribution of non-conventional data to the decision-making process is as much valuable as it is diverse, according to the nature of the considered data. Social data provide a fresh and timely perception of the market mood and can be used to better understand the phenomena of business and society. Linked data provide a structure and interlinked knowledge that can be exploited to integrate and extend the available corporate knowledge. Schemaless data are a large and heterogeneous source of information whose structure and requirements can be determined on-the-fly to accommodate different users and different business scenarios.

It is clear that no standard approach can be adopted to cope with all kinds of non-conventional data. Nonetheless, we identify the following branches of BI 2.0 that (also) address the issue of integrating different kinds of non-conventional data into the BI process.

- *Social Business Intelligence* (SBI) is the discipline of effectively and efficiently combining corporate data with social data to let decision-makers analyze and improve their business based on the trends and moods perceived from the environment [47]. In the industrial world, the analysis of social data often relies on commercial tools known as *social media monitoring tools*: they provide a fixed set of dashboards to analyze the data and rely on some ad-hoc KPIs, so they lack in providing flexible user-driven analyses. Conversely, SBI is more focused on the "big picture", so as to give decision makers an unprecedentedly comprehensive picture of the ongoing events and of their motivation. As in traditional BI, the goal of SBI is to allow users with a limited expertise in databases and ICT to carry out powerful and flexible analyses; this goal is typically achieved by storing information into a data warehouse, in the form of multidimensional cubes to be accessed through OLAP techniques.

- *Exploratory Business Intelligence* is a trending approach that indicates the enrichment of the decisional process by including, besides data extracted from the corporate sources (such as ERPs and CRMs), also external data (coming, for instance, from the web). The focus is on the fact that the recognition and acquisition of these external data is carried out by the user in an exploratory way, i.e., she navigates the information at her disposal and chooses what should be acquired and what should not (a process also known as *surf and save*). The data to be acquired is often referred to as *situational data*, to emphasize the fact that they have a narrow focus on a specific business problem and, typically, a short lifespan for a small group of decision makers with a unique set of needs [111]. A further push towards exploratory BI is now coming from the wide success of *data science*, meant as the cross-discipline of extracting knowledge and insights from large volumes of data in various forms, either structured or unstructured, either internal or external to the company.

- *Pervasive Business Intelligence* has emerged with the rise of BI 2.0 and highlights the key aspects that such tools should cope with in order to meet the new and ambitious requirements of business users. In particular, the pervasiveness of BI tools refers to a series of characteristics such as *personalization* (i.e., to customize the result according to the user who takes advantage of it), *timeliness* (i.e., to promptly provide the business information for decision-making) and *integration* (i.e., to enable the access to information from any available source) [166]. The aspect of integration gains considerable importance in presence of corporate data lakes, where the data can be interpreted differently according to the specific requirements and vision of the user. Besides, the raising popularity of *data enthusiasts* (i.e., users that need the ability to analyze data even without formal training in data science) heightens the need of BI tools that answer the issues of pervasiveness [120].

Whereas the integration of non-conventional data in the BI yields great value for the business, we also recognize that this large and heterogeneous wealth of data can become overwhelming for the user — who may end up struggling to garner the most valuable information for decision making. On the one hand, decision makers heavily rely on OLAP tools, as they have a universally recognized key role in supporting flexible and effective exploration of multidimensional cubes in data warehouses; on the other hand, it is commonly agreed that the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating [85]. Providing users with tools and techniques that help them in getting the most out of the available data becomes an even more important task in this context. For this reason, a complementary part of this thesis is dedicated to the study and enhancement of *recommendation systems*, which supervise the user's activity by providing suggestions for the "next step" in the analysis process. Such work falls within the Pervasive BI discipline as well, since recommendation systems constitute one of the answers to the personalization aspect of pervasiveness.

1.2.1 Social BI

In the analysis of social data, a key role is played by *topics*, meant as specific concepts of interest within the subject area [53]. Users are interested in knowing how much people talk about a topic, which words are related to it, if it has a good or bad reputation, etc. Thus, topics are obvious candidates to become a dimension of the multidimensional cubes for SBI. In this context, we give the following contributions:

- We formally introduce the *meta-star* as an alternative to the traditional star-schema for modeling hierarchies of topics, which combines meta-modeling, navigation tables and traditional dimension tables.
- We discuss how meta-stars can be exploited to formulate meaningful queries in the SBI context, by providing examples and highlighting the differences from queries on the star schema.
- We provide the execution plans and a cost model for meta-stars.
- We evaluate meta-stars in terms of efficiency and space occupation.

Our experience in the field of SBI has grown also thanks to our involvement in different real-world projects. Most importantly, we participated in *WebPolEU*[1], a project aimed at analyzing the correlation between social media and the European elections of 2014. We make use of the experience gained in such project to provide the following contributions:

- We introduce the architectural and methodological framework that we propose to carry out an SBI project.
- We discuss the architectural choices adopted in WebPolEU and evaluate them in terms of effectiveness, efficiency and sustainability.
- We describe a cross-disciplinary benchmark for SBI, called SABINE, obtained through a series of enrichments on the data collected with WebPolEU and released to the public.

[1]http://webpoleu.net

The meta-star modeling technique is discussed in Chapter 3; the experience on the WebPolEU project is discussed in Chapter 4; finally, the SABINE benchmark is discussed in Chapter 5.

1.2.2 Exploratory BI

In the context of Exploratory BI, we focus on linked data as a valuable source to gather the knowledge required to integrate and extend the corporate multidimensional cubes. Since linked data are often chaotic and badly organized, especially from the schema point of view , data scientists are often prevented from taking full advantage of the informative wealth lying within them. The contributions that we provide in this context are summarized as follows:

- We propose *iMOLD* (Interactive Multidimensional Modeling of Linked Data) as an interactive and collaborative approach to build aggregation hierarchies from linked data.
- We formally define five aggregation patterns found in ontologies and provide the algorithms for detecting them in RDF linked data and translating them into multidimensional hierarchies.
- We provide a case study and an extensive user evaluation to demonstrate the applicability and the advantages of the proposed approach.

The discussion of the iMOLD approach is provided in Chapter 6.

1.2.3 Pervasive BI

As anticipated, the discipline of Pervasive BI is address in terms of integration (i.e., to enable access and analysis on schemaless data) and personalization (i.e., to address the issue of disorientation in the user's experience).

In the context of the integration, we recall that corporate data lakes have the main advantage of increasing the flexibility in storing the data, due to the absence of a unique, well-defined schema. However, this turns to a disadvantage when moving from operational applications to analytical applications and business intelligence. As BI analyses typically involve large sets of data, a single analysis may involve instances with different — possibly conflicting — schemata. It is then crucial for the user to understand the rules that drove the usage of different schemata, so as to start an integration activity that identifies a common schema to be adopted. To this end, we provide the following contributions:

- We propose an approach to perform the activity of *schema profiling* in a document-based database, which first identifies the distinct schemata in a collection of documents and then adopts a decision tree algorithm to determine the usage of each schema.
- We implement a classification algorithm (based on the well-known C4.5) that builds on two original features: the coupling of value-based and schema-based conditions and the introduction of a novel measure of entropy.
- We evaluate our approach in term of effectiveness and efficiency against both synthetic and real-world dataset.

In the context of personalization, we first state that different approaches have been taken in the literature to address the issue of user disorientation; in particular both preference-based (e.g., [64, 86]) and recommendation techniques (e.g., [157, 57]) were specifically devised for OLAP systems. Our work is specifically focused on recommendations and propose an innovative approach stemming from collaborative filtering. In particular, we provide the following contributions:

- We propose a collaborative approach inspired by clickstream analysis which provides recommendations in the form of sequences of OLAP queries (i.e., OLAP *sessions*) instead of individual OLAP queries.
- We devise a sophisticated algorithm that not only identifies the most suitable recommendation for the user, but also adapts it to fit the specific user's preferences.
- We perform an extensive evaluation of the approach in terms of quality, effectiveness and efficiency assessment.
- We propose *CubeLoad*, a parametric benchmark generator of OLAP session, which was used to generate the dataset for the experimental evaluation of the recommendation approach.

The work in the area of schema profiling is discussed in Chapter 7; the benchmark generator CubeLoad is discussed in Chapter 8; the work concerning the recommendation system is discussed in Chapter 9.

Chapter 2

Background

In this chapter we provide an introduction to the fundamental concepts that constitute the basis of BI. First, we describe the features of a DW, which is at the core of any BI solution. We continue by providing an overview of the traditional architectures for data warehousing and introducing the paradigm behind OLAP analysis. Then, we provide a formal definition of the key elements of the multidimensional model, which will be recalled in the following chapters. We conclude by introducing the basic logical modeling technique of a DW, i.e., the star schema.

2.1 Data Warehouse

The data warehouse (DW) is the main component in a BI platform and is defined as "a collection of data that supports the decision-making process" [63]. The goal of the DW is to provide data in a format that is most suitable for the analysis by the decision makers. The ability to rapidly access all the available information is of crucial importance for any company. Such information is mainly extracted from the wealth of data stored in operational databases by means of a progressive selection and aggregation process, as shown in Fig. 2.1.

The role of a DW is fundamentally different from the one of operational databases, which are typically used in all kinds of applications. Operational database are used to store operational data, i.e., data created by business operations involved in daily management processes (e.g., purchase management, sales management, invoicing). The process of managing operational data is typically defined as *OLTP* (online transactional processing) and is characterized by a mainly transactional workload: a high number of users continuously interact with the database to carry out simple read and write operations (e.g., add a product to the basket). Conversely, the process of data

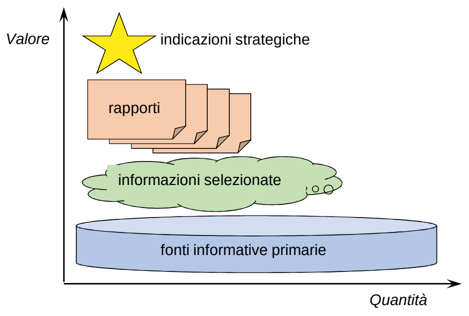


Fig. 2.1 The progressive selection and aggregation process to extract valuable information from operational data sources

analysis by decision makers is defined as *OLAP* (online analytical processing) and is characterized by a low number of users that periodically interact with the database to carry out complex read operations (e.g., identify the products that maximize the profit). Since directing such a workload on an operational database would be highly inconvenient, a DW is typically used for OLAP purposes.

The complexity of OLAP queries lies not only in the large amounts of data to process, but also in the involvement of historic data and in the collection of data from different data sources. Data warehouses must then comply with the problem of data historicization on the one hand (as they need to keep track of the evolution of the data, which is typically lost by overwrites in operational databases), and the problem of data integration on the other. In particular, [63] identifies the following requirements for a data warehouse:

- *accessibility* to users not very familiar with IT and data structure;
- *integration* of data on the basis of a standard enterprise model;
- *query flexibility* to maximize the advantages obtained from the existing information;
- *information conciseness* allowing for target-oriented and effective analyses;
- *multidimensional representation* giving users an intuitive and manageable view of information;
- *correctness* and *completeness* of integrated data.

Data warehouses fulfill these requirements by providing the following features [82]:

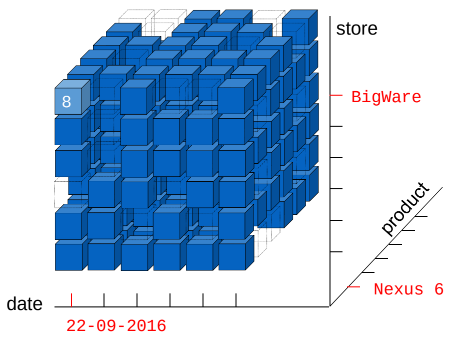


Fig. 2.2 A sample multidimensional cube.

- *Subject-orientation*: the focus of the data warehouse is on enterprise-specific concepts, such as customers, products, sales, and orders. On the contrary, operational databases hinge on many different enterprise-specific applications.

- *Integration and consistency*: data warehouse provide a unified view of the data gathered from multiple data sources, such as data extracted from production and then stored to enterprise databases, or even data from a third party's information systems. In general, data warehouses do not add new information, but they rearrange the existing information that was already available.

- *Persistence*: data warehouses periodically capture a snapshot of the operational data and integrate it with the previously collected data, in order to provide a complete picture that covers years of transactions.

To facilitate OLAP analyses, the DW is typically broken up into different *data marts*, each representing a subset or an aggregation of the data stored in the primary DW. A data mart includes a set of information pieces relevant to either a specific business area, a corporate department, or a category of users. The data mart is composed by different *facts* (e.g., orders and sales) that are the basic concepts of the multidimensional schema. Each fact is analyzed by different perspectives, called *dimensions* (e.g., products and stores). Each dimension is associated to a *hierarchy* characterized by different levels of aggregation, called *attributes*. The actual values of each attribute (e.g., a specific product or store) are referred to as *members*. Each instance of a fact is called an *event* (e.g., a particular order or a specific sale); it is described by the values of a set of relevant *measures* (e.g., the quantity sold) that provide a quantitative description of the event, and it is identified by the combination of members on each dimension that is relates to (e.g., the sale of a specific product in a specific store on a specific date). Starting from these concepts, the multidimensional

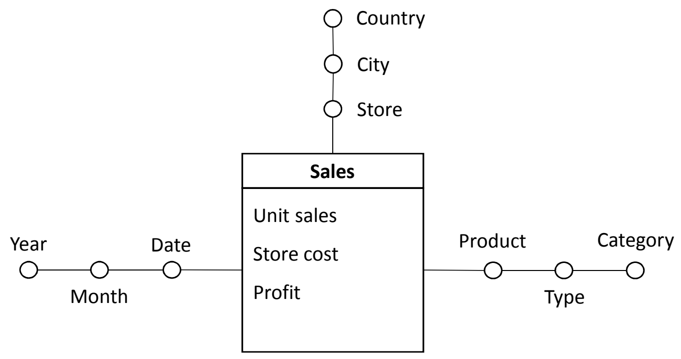


Fig. 2.3 The conceptual DFM representation of the multidimensional cube in Fig. 2.2.

data can be represented by an n-dimensional *cube*, where n is the number of dimensions. Each cell of the cube includes a value for each measure. To represent the conceptual design of a data warehouse, the *dimensional fact model* (DFM) is typically used [61].

Example 1 *Figure 2.2 shows a three-dimensional cube representing the sales in a store chain. The dimensions are products, stores, and dates. Each cell represents the sales of a given product in a given store and in a given date; the highlighted cell shows that, on the 22nd of September, 2016, 8 units of Nexus 6 (member of level Product) have been sold in BigWare (member of level Store). Note that the cube is sparse, as it is (naturally) not guaranteed that each product is sold every day and in every store. Figure 2.3 shows the conceptual representation of the same cube according to the DFM. The central box represents the fact (Sales) with its measures (Unit sales, Store cost and Profit); each dimension is associated to a hierarchy of levels, where arches between levels represent many-to-one relationship (in outward direction from the cube). For instance, the products (e.g., Nexus 6) can be aggregated into types (e.g., smartphones) and into categories (e.g., mobile technology).*

2.2 DW Architecture

A common classification divides DW architectures in three main classes, depending on the number of levels they involve. Figure 2.4 shows the three kinds of architectures.

- In a *single tier* architecture, the user (i.e., the decision maker) directly performs her analyses on the operational data. This is typically achieved by means of a *middleware* software that shows to the user a virtual representation of the data marts without relying of a supporting physical tier: the only physical tier is composed of the operational data sources.

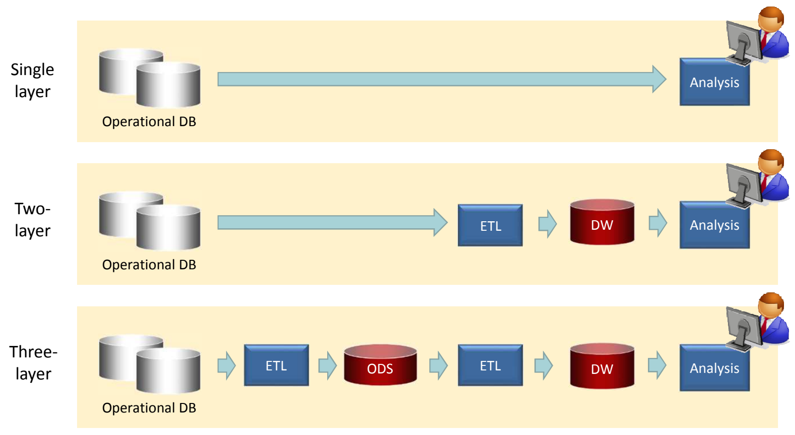


Fig. 2.4 The three kinds of DW architectures according to the number of involved levels.

- In a *two-tier* architecture, a data warehouse is physically implemented to store the data marts. This architecture introduces an ETL process (*extraction, transformation and loading* that extracts the data from the operational sources, filters and cleanses the data (e.g., to remove inconsistencies and fill the gaps) and integrates them into a common schema. The DW tier provides a single, centralized repository that allows the separation of the OLAP workload from the OLTP one.

- In a *three-tier* architecture, an additional physical tier is inserted between the operational sources and the DW. The goal of the *operational data store* is to act as a reconciled tier, which materializes the operational data obtained after integrating and cleansing source data. This kind of architecture physically separate the ETL process of data filtering and cleansing from the ETL process that populates the DW. Although this solution leads to more redundancy of the source data, it provides a common reference data model for a whole enterprise and allows additional operations to be accomplished on the ODS (e.g., execution of daily reports or enrichment of the reconciled data).

An important remark is that this thesis is focused on non-conventional data, which — by definition — do not correspond to the operational sources mentioned in the architectures. As mentioned in Chapter 1, we will investigate the use of social and linked data (which can be summarized as *external data*, i.e., data coming from the web) as well as schemaless data (which are typically stored in *data lakes*, i.e., large repositories of heterogeneous data). The incorporation of external data and data lakes in a company information system can take different shapes and there are no standard reference architectures. As a result, the architectures adopted in the different specific scenarios will be individually presented in the following chapters of this thesis.

2.3 OLAP Analysis

The interactive navigation of DW information by the user is known as OLAP analysis. Typically, the data are analyzed at different levels of aggregation by applying subsequent OLAP operators, each yielding one or more different queries. The user can scout the multidimensional model and choose the next operator based on the outcome of the previous ones. In this way, the user creates a navigation path that corresponds to an analysis process for facts according to different points and at different detail levels. This is also informally called an *OLAP session*. In the following we describe the most common OLAP operators, referring to the cube of sales of Figure 2.3:

- *Roll-up* causes an increase in data aggregation and removes a detail level from a hierarchy (e.g., from product to type).
- *Drill-down* is the complement to the roll-up operator; it reduces data aggregation and adds a new detail level to a hierarchy (e.g., from category to type).
- *Slice-and-dice* reduces the number of cube dimensions after setting one of the dimensions to a specific member (e.g., product="Nexus 6"); the dicing operation reduces the set of data being analyzed by a selection criterion.
- *Pivot* implies a change in layouts, aiming at analyzing a group of data from a different viewpoint.
- *Drill-across* allows to create a link between concepts in interrelated cubes, to compare them.
- *Drill-through* switches from multidimensional aggregate data to operational data in sources or in the reconciled tier.

2.4 Multidimensional Model

In this section we provide a formal definition of the main concepts in the multidimensional model that will be used in the following chapters of the thesis.

Definition 1 (Multidimensional Schema) *A* multidimensional schema *is a couple* $\mathcal{M} = \langle Hier, Meas \rangle$ *where Hier is a finite set of* hierarchy schemata*, and Meas is a finite set of* measures*, i.e., numerical attributes.*

Given a measure $m \in Meas$, we will use $Dom(m)$ to define the numerical domain of m.

Definition 2 (Hierarchy Schema) *A* hierarchy schema $\mathcal{H} \in Hier$ *is a couple* $\mathcal{H} = \langle L, \succ \rangle$ *where L is a set of levels, and $\succ$ is a* roll-up *partial order of L. We will write $l_k \dot{\succ} l_j$ to emphasize that l_k is an immediate predecessor of l_j in $\succ$, and $Lev(\mathcal{H})$ to refer to L.*

Let use defined $H = \{h_1, ..., h_n\}$ as the set of hierarchies that respectively conform to the the hierarchy schemata in $Hier$. Given a hierarchy $h_i \in H$, the top level of the order typically defines the name of the corresponding dimension and determines the finest aggregation for the hierarchy. Conversely, the bottom level is denoted by ALL_i, has a single possible member and determines the coarsest aggregation for the hierarchy. Given a level $l_k \in L$, we define $Dom(l_k)$ the set of members of the level.

Example 2 *The multidimensional schema in Figure 2.3 has three dimensions, namely* DATE*,* STORE *and* PRODUCT*, and measures* Unit sales*,* Store cost *and* Profit*. The roll-up order in the product dimension is such that* Product $\succ_{PRODUCT}$ Type $\succ_{PRODUCT}$ Category*. The domain of level* Product *includes member "Nexus 6", while the domain of level* Type *includes member "smartphone".*

To define a multidimensional cube (or simply cube) we must first introduce the notion of *group-by set*. A group-by set includes one attribute for each hierarchy, and defines a possible way to aggregate data. A coordinate of a group-by is a point in the n-dimensional space defined by the attributes in that group-by set.

Definition 3 (Group-by Set) *Given a multidimensional schema $\mathcal{M}$, let $Dom(Hier) = Lev(\mathcal{H}_1) \times ... \times Lev(\mathcal{H}_n)$ where $\mathcal{H}_i \in Hier$, $1 \leq i \leq n$; each $B \in Dom(Hier)$ is a group-by set in $\mathcal{M}$. Let $B = \langle l_{k_1}, ..., l_{k_n} \rangle$ and $Dom(B) = Dom(l_{k_1}) \times ... \times Dom(l_{k_n})$; each $b \in Dom(B)$ is called a* coordinate *of B.*

Example 3 *Example group-by sets in the multidimensional schema in Figure 2.3 are:*

- $B_1 = \langle$ Month, Store, Category $\rangle$
- $B_2 = \langle$ Year, Country, Type $\rangle$

A cube is populated with facts, each recording a useful information for the decision-making process. A fact is characterized by a group-by set B that defines its aggregation level, by a coordinate of $b \in Dom(B)$, and by a value for one measure.

Definition 4 (Fact) *Given a multidimensional schema $\mathcal{M}$, a group-by set B and a measure $m \in Meas$, a fact (or event) is a couple $e_{B,m} = \langle b, v \rangle$, where $b \in Dom(B)$ and $v \in Dom(m)$. The space of all facts for $\mathcal{M}$ is:*

$$\mathcal{E}_{\mathcal{M}} = \bigcup_{B,m} (Dom(B) \times Dom(m)) \tag{2.1}$$

Finally, we define a cube as follows.

Definition 5 (Cube) *A* cube *is a set of facts $C \subseteq \mathcal{E}_{\mathcal{M}}$ such that no two facts characterized by the same coordinate exist in C.*

2.5 Logical modeling: the star schema

The multidimensional structure can be represented in accordance to three different logical models.

- *MOLAP* (Multidimensional OLAP) systems use multidimensional structures to represent cubes; for instance, multidimensional vectors can be used, where each vector is associated with a set of coordinates in the space of all facts. MOLAP system provide the simplest representation of data warehouse data and deliver top performance because of their perfect suitability for OLAP tasks. The main problem, however, is the one of data sparsity, as multidimensional DBMSs have to store every cell in the fact space, even if only a small percent of them actually includes any information [25].

- *ROLAP* (Relational OLAP) systems adopt the well-known relational model, i.e., the database industry standard which every professional database designer is familiar with. ROLAP systems rely on decades of evolution of relational DBMSs, which have received much more research and industry attention than MOLAP systems. One of the main advantages of ROLAP systems is its propensity to scalability, as there is no sparsity in relational systems when storing multidimensional data.

- *HOLAP* (Hybrid OLAP) systems adopt a hybrid model, which relies on both the multidimensional and the relational model to exploit the respective unique advantages. Different strategies can be adopted to achieve such goal; for instance, a cube can subdivided into *chunks*, where the dense ones are stored in MOLAP mode and the sparse ones are stored in ROLAP mode.

In this thesis we will work on ROLAP systems, where the reference multidimensional modeling technique is the so-called *star schema*. A star schema consists of the following:

- A set of relations $(DT_1, ..., DT_n)$ called *dimension tables*, each of them corresponding to a dimension. Every DT_i features a primary (typically surrogate) key and a set of attributes describing its dimension at different aggregation levels.
- A *fact table* (FT) referencing all the dimension tables. The primary key of the FT is the composition of the set of foreign keys referencing dimension tables; an FT also includes an attribute for every measure.

Dimension tables essentially rely on denormalization to store all the levels of the hierarchy in the same relation; whereas this causes redundancy and requires more disk space to store the data, it also reduces the number of joins needed to retrieve information. The absence of normalization does not typically raise concerns as hierarchies are mostly static.

Example 4 *Given the cube in Figure 2.3, the query to obtain the sum of the profit for each store in each product category would be the following.*

```
SELECT    DT_STORE.Store, DT_PRODUCT.Category, SUM(FT.Profit)
FROM      DT_STORE, DT_PRODUCT, FT
WHERE     FT.keyS = DT_STORE.keyS AND FT.keyP = DT_PRODUCT.keyP
GROUP BY  DT_STORE.Store, DT_PRODUCT.Category;
```

The star schema also has a series of variants that accommodate different specific requirements. For example, the *snowflake schema* is obtained from the star schema by reintroducing some degree of normalization in dimension tables, for instance to allow a portion of hierarchy to be shared by multiple dimension tables. Also, *slowly-changing dimensions* [95] can be used to handle historicization of the data by introducing additional attributes in the dimension tables. In Chapter 3 we will introduce a brand new modeling technique (called meta-star) to store dimensional data in a dynamic and heterogeneous scenario as the one of Social BI.

Chapter 3

Meta-star: a modeling approach for social data aggregation

In this chapter we propose meta-stars, a modeling approach for multidimensional hierarchies that is alternative to the traditional star schema and that effectively supports the analysis of social data. Its basic idea is to use meta-modeling coupled with navigation tables and with dimension tables: navigation tables support hierarchy instances with different lengths and with non-leaf facts, and allow different roll-up semantics to be explicitly annotated; meta-modeling enables hierarchy heterogeneity and dynamics to be accommodated; dimension tables are easily integrated with standard business hierarchies. After describing the meta-star approach, we formalize its querying expressiveness and give a cost model for the main query execution plans. Then, we evaluate meta-stars by presenting experimental results for query performances and disk space.

3.1 Introduction

As mentioned in Chapter 1, social data come in the form of user-generated content (UGC), i.e., any form of content created and published by a user, from simple blog posts to audio and video files [99]. The issue of extracting most information out of the UGC (and using it) is a hot research theme in different areas, such as information retrieval, text mining, and natural language processing; each community contributes to this common goal by employing different techniques. In the perspective of SBI, the data to be combined have very different features: while corporate data are structured, reliable, and accurate, UGC is unstructured or poorly structured, possibly fake, often vague and imprecise; however, both types of data are crucial for an effective decision-making process.

The category of UGC that most significantly contributes to the decision-making process in the broadest variety of application domains is the one coming in the form of textual *clips* [146, 181]. Clips can either be messages posted on social media (such as Twitter, Facebook, blogs, and forums) or articles taken from on-line newspapers and magazines. Digging information useful for decision-makers out of textual UGC requires first crawling the web to extract the clips related to a *subject area*, then enriching them in order to let as much information as possible emerge from the raw text. The subject area defines the project scope and extent, and can be for instance related to a brand or a specific market. Enrichment activities range from the simple identification of relevant parts (e.g., author, title, language) if the clip is semi-structured, to the use of either natural language processing or text analysis techniques to interpret each sentence and if possible assign a polarity to it (i.e., *sentiment analysis* or *opinion mining* [107]).

In the analysis of textual UGC, a key role is played by *topics*, meant as specific concepts of interest within the subject area [53]: users are interested in knowing how much people talk about a topic, which words are related to it, if it has a good or bad reputation, etc. A topic could be a word having a specific role in the users' business glossary (e.g., a product, a product type, or a brand), or it could be a common word that at some time becomes relevant to the subject area. Depending on the tool or technique adopted for UGC analysis, each topic is normally coupled with some measures taken either at the clip/sentence level (e.g., number of occurrences of that topic in each clip) or for each single occurrence (e.g., sentiment of each occurrence of that topic in each clip). Such a detailed information is useful, e.g., for early-alerting applications [59] in which users need to timely react to some specific message; however, to effectively summarize the mood raised by a topic, topic measures must be aggregated using clip metadata, e.g., by author, media type, or language, which can be easily done through OLAP-style analyses.

On the other hand, OLAP analyses of single, specific topics are often not sufficient to give users a clear and comprehensive picture of the social mood. Like for any other dimension, users are also very interested in grouping topics together in different ways to carry out more general and effective analyses —which requires the definition of a topic hierarchy that specifies inter-topic roll-up (i.e., grouping) relationships so as to enable aggregations of topic measures at different levels. Once a topic hierarchy is available, it can be used to compute aggregated measures.

3.1.1 Motivation

Topic hierarchies are quite different from traditional hierarchies (like the temporal and the geographical one) for several reasons [29]:

♯1 From the point of view of their instances, topic hierarchies are *irregular* in several ways. According to the terminology introduced in [134] they are *non-onto*, which means that hierarchy instances can have different lengths and also non-leaf topics can be related to facts (e.g., clips may talk of smartphones as well as of the Galaxy III). Topic hierarchies are also *non-covering* (some hierarchy levels may be missing in some instances) and *non-strict* (many-to-many relationship between topics may exist)[1]. Note that, in ROLAP (Relational OLAP) contexts, a non-onto hierarchy can be represented either by coupling a classic dimension table with a *navigation table* that explicitly represents the transitive closure of the node relationships, or by creating a parent-child (i.e., recursive) dimension table [95]. Conversely, a non-strict hierarchy can be dealt with using many-to-many *bridge tables* [95].

♯2 Trendy topics are heterogeneous (e.g., they could include names of famous people, products, places, brands, etc.) and change quickly over time (e.g., if at some time it were announced that using smartphones can cause finger pathologies, a brand new set of hot unpredicted topics would emerge during the following days), so a comprehensive schema for topics cannot be anticipated at design time and must be dynamically defined. For some topics a classification could even be hard due to their fuzzy nature, or unnecessary due to their transitoriness. So, from the point of view of their schemata, we can intuitively say that topic hierarchies are *fluid*.

♯3 Some topics (e.g., products) are normally part of the hierarchies that store business data. Thus, modeling those topics in such a way as to enable direct *integration with the EDW* (Enterprise Data Warehouse) is highly desirable.

♯4 Relationships between topics can have different *roll-up semantics*: for instance, the relationship semantics in "Galaxy III has brand Samsung" and "Galaxy III has type smartphone" is quite different. In the multidimensional model this distinction can only be (implicitly) enforced by leaning on the semantics of aggregation levels, which is possible for a regular hierarchy ("Smartphone" is a member of level Type, "Samsung" is a member of level Brand) but not for a non-onto hierarchy because all topics are members of the same Topic level.

[1]Non-onto hierarchies are also called *unbalanced* [129] or *recursive* [63], while non-covering hierarchies are also called *ragged* [129] or *incomplete* [63]

Example 5 *In our motivating example, a marketing analyst wants to analyze people's feelings about mobile devices. A basic cube she could use to this purpose is the one counting, within the textual UGC, the number of occurrences of each topic related to subject area "mobile technologies", distinguishing between those expressing positive/negative/neutral sentiment as labeled by an opinion mining algorithm (e.g., the one in [164]). Figure 3.1 shows a sample set of topics for mobile technologies and chain stores and the roll-up relationships the analyst deems interesting (e.g., when analyzing topic "Samsung", the analyst may wish to also include occurrences of topics "Galaxy III" and "Galaxy Tab"), while Table 3.1 gives some sample facts with four measures (the total number of occurrences is higher than the sum of positive and negative ones, because most occurrences are normally unbiased; the average sentiment is computed by averaging the numerical sentiment scores for the occurrences). Note that, since this example is from the marketing area, most topics reasonably correspond to values of attributes (e.g., products) in the EDW; in other settings, topics could also be more common words such as "finger pathologies" in our example, or emerging trends like "hands-free" and "wearable device". Now, let the analyst be specifically interested in three types of analysis of the UGC: (i)* brand reputation*, aimed at assessing the people's perception of each brand; (ii)* talking volume*, whose goal is to count the overall occurrences of mobile tech topics; and (iii)* health rumors*, aimed at capturing the customers' concerns about touchscreens and the possible pathologies they may cause. In the first case, the perception of Samsung will be measured by counting the positive and negative occurrences of topics "Samsung", "Galaxy III", and "Galaxy Tab"; in the second case, all occurrences of all tech-related topics except "Nokia" and "Samsung" will be counted; in the third case, only the occurrences of "Touchscreen" and "Finger Pathologies" will be considered. The results are shown in Table 3.2; it appears that, depending on the analyst's goals, facts can be aggregated in different ways by navigating or not navigating inter-topic relationships with the different semantics shown in grey in Figure 3.1.*

3.1.2 Goal

In light of the above, topic hierarchies in ROLAP contexts must clearly be modeled with more sophisticated solutions than traditional star schemata. Though some attempts have been made in the literature to address some of the mentioned issues (e.g., [181, 29]), no solution to all of them has been found so far. The approach we propose in this chapter is called *meta-stars*, and it couples meta-modeling with navigation tables and with dimension tables to improve flexibility and expressiveness when modeling topic hierarchies [47–49]. Meta-stars are flexible because, differently

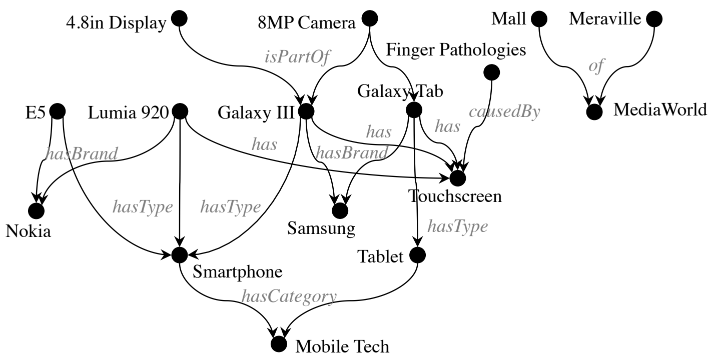


Fig. 3.1 A topic hierarchy for the mobile technology subject area; inter-topic roll-up relationships are represented by arrows and labelled (in grey) with their semantics

Table 3.1 Sample (fake) facts for topics

Topic	positiveOcc	negativeOcc	totalOcc	avgSentiment
4.8in Display	10	2	180	2.1
8MP Camera	0	3	12	-0.2
E5	30	10	100	1.0
Lumia 920	10	10	50	0.0
Galaxy III	20	5	1005	0.1
Galaxy Tab	22	0	102	2.0
Nokia	20	10	35	5.0
Samsung	50	10	400	2.5
Tablet	5	5	30	0.1
Smartphone	60	20	1600	1.8
Mobile Tech	10	20	3000	-0.1
Touchscreen	60	10	100	3.2
Finger Path.	0	25	25	-4.6
Mall	5	5	200	0.1
Meraville	3	0	50	2.3
MediaWorld	10	5	2000	0.5

from traditional solutions that mainly model regular hierarchies with a fixed and pre-defined structure, they use a single schema to cope with hierarchies with very different features. Indeed, navigation tables support non-onto, non-covering, and non-strict hierarchies (requirement ♯1); meta-modeling enables heterogeneity and dynamics of topic classification to be accommodated without requiring changes to the underlying schema (requirement ♯2); finally, dimension tables are easily integrated with standard business hierarchies (requirement ♯3). On the other hand, meta-stars are more expressive than traditional solutions because they can model unclassified topics (requirement ♯2) and because different roll-up semantics can be explicitly annotated (requirement ♯4), which in turn enables a brand new class of semantics-aware OLAP queries.

As discussed in Section 3.6, an obvious consequence of the adoption of navigation tables is that the total size of the solution increases exponentially with the size of the

Table 3.2 Brand reputation, talking volume, and health rumors analyses

Topic	positiveOcc	negativeOcc	totalOcc
Nokia	60	30	
Samsung	92	15	
Mobile Tech			6079
Touchscreen		35	

topic hierarchy. This clearly limits the applicability of the meta-star approach to topic hierarchies of small-medium size; however, we argue that this limitation is not really penalizing because topic hierarchies are normally created and maintained manually by domain experts, which suggests that their size can hardly become too large.

The remainder of the chapter is organized as follows:

- In Section 3.2 we discuss the related literature.
- In Sections 3.3 and 3.4 we present our approach and the types of queries it supports.
- In Section 3.5 we show query execution plans and propose a cost model.
- In Section 3.6 we propose a set of experimental tests.
- In Section 3.7 we draw the conclusions.

3.2 Related Literature

OLAP techniques are normally applied to multidimensional cubes storing structured business data. However, several research paper focus on the possibility of enhancing OLAP analyses and broadening its scope to unstructured content. Overall, as sketched in Table 3.3, the literature related to our approach can be classified into three partially overlapping areas: *OLAP analyses on textual documents*, *advanced data warehouse modeling*, and *analysis of social contents*. In the following we briefly comment on each cited paper, with more emphasis on those that come closer to our work. Table 3.3 also summarizes a comparison between previous approaches and ours in terms of supported features.

In [51] cubes are exploited to compute multidimensional aggregations on classified documents, using measures such as keyword frequency, document count, document class frequency; the hierarchies used for analyses are based on a given ontology, which limits the approach flexibility. A cube for analyzing term occurrences in documents

Table 3.3 Related literature classification and comparison

Reference	*OLAP on text doc.*	*Advanced DW modeling*	*Anal. of social cont.*	irregularity	fluidity	EDW integration	roll-up semantics
[146]	✓		✓	no	no	yes	no
[181]	✓	✓	✓	no	part.	yes	no
[53]	✓	✓	✓	no	no	yes	no
[29]	✓	✓	✓	part.	part.	yes	no
[134]		✓		yes	no	yes	part.
[129]		✓		part.	no	yes	no
[51]	✓			part.	no	no	no
[101]	✓			no	no	yes	no
[144, 145]	✓			part.	no	yes	no
[10]	✓			no	yes	no	no
[110]		✓		yes	no	yes	part.
[115]			✓	n.a.	n.a.	n.a.	n.a.
[113]			✓	n.a.	n.a.	n.a.	n.a.
[138]	✓			no	no	yes	no
Our approach		✓	✓	yes	yes	yes	yes

belonging to a corpus is proposed in [101]; the categorization of terms is obtained from a thesaurus or from a concept hierarchy such as Wordnet. A measureless cube for OLAP analysis of semi-structured documents is presented by [144]; a novel OLAP operation called *focus* is introduced to specify a subject of analysis and aggregate data accordingly. Similarly, [10] proposes a multidimensional model for OLAP analysis of XML documents based on a measureless fact where a semantic dimension is coupled with standard dimensions like date and place. In a related paper [145], a *top keyword* aggregation function is defined to represent a set of documents by their most significant terms using the well-known tf-idf weighing function. Finally, [138] shows how OLAP and information retrieval functionalities can be integrated to access both structured data stored in a data warehouse and unstructured data in form of documents; a global ontology models the business domain and provides the mappings to connect OLAP and information retrieval.

A work sharing some similarities with ours is the one in [146], that presents an architecture to extract tweets from Twitter and load them to a data warehouse. Conceptual models for Twitter streams from both OLTP and OLAP points of view are also proposed. However, both models are focused on the inter-relationships between tweets and between users (the *influencer/followers* mechanism), and little attention is paid to classifying and analyzing tweet topics. An approach for disambiguating and categorizing the entities in the tweets aimed at discovering topics is described in [115];

Wikipedia is used as a knowledge base to this end. The results obtained are used for determining users' topic profiles, and the possibility of analyzing them using OLAP techniques is not considered. The real-time identification of emerging topics in tweets is studied in [113]. Bursty keywords are extracted first, then grouped to identify trends; however, trends are analyzed using a front-end with limited flexibility.

In [53] a multidimensional model is proposed that shares with meta-stars the goals of integrating sentiment data in a traditional EDW and that of supporting non-onto hierarchies. However, neither facts for non-leaf topics, nor multiple semantics of aggregation, nor non-strict hierarchies are allowed. Besides, schema dynamics is not supported. Finally, the model is defined on a specific subject area (i.e., analysis of market data) and is not formalized in the general case. Topic modeling is also the goal of the approach in [181], that extends traditional cubes to cope with a topic hierarchy and to store probabilistic content measures of text documents learned through a probabilistic model. The topic hierarchy is a tree that models parent-child relationships between topics of interest.

In [29] the authors model non-onto and non-strict topic hierarchies as DAGs of topics. On the one hand, the proposed solution has higher expressivity with respect to traditional hierarchies due to the presence of topic-oriented OLAP operators; on the other hand, it lacks in providing a semantics for the topics in the DAG, that are organized and aggregated only according to their position in the DAG. In other words, with reference to Example 5, the user cannot ask for the average sentiment of each single smartphone since there is no evidence of which instances have type "Smartphone".

Apart from the specific social context, advanced modeling of multidimensional hierarchies has been studied by several authors [110, 134]. However, none of the proposed solutions completely match the topic hierarchy requirements.

To the best of our knowledge, in the commercial world no solution is offered to run OLAP analyses on UGC. The platform whose functionalities are closest to those achieved by our approach is SAS, that exploits its in-memory engine (called SAS In-Memory Analytics Server) to store facts and dimensions in a single, flat in-memory table. The SAS solution supports manual definition of topic hierarchies and their navigation; however, it has inherent limits due to memory availability and does not allow the UGC to be integrated with the information stored in the EDW.

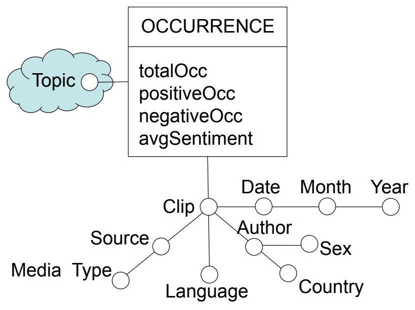


Fig. 3.2 A conceptual representation of a cube for analyzing textual UGC

3.3 Meta-Stars

Different multidimensional cubes can be stored in a corporate data warehouse, focused for instance on the perceived sentiment for the topics in the subject area, on the correlations between topics, on the trending topics, and so on as determined by the semantic enrichment process (Figure 3.2 shows a simple cube for Example 5, that can also be used for any other subject area). Typical indicators associated to these cubes are the *topic share* (ratio between the number of occurrences of a topic and the total number of occurrences of all topics in a given time interval), the *topic awareness* (ratio between the number of clips mentioning a topic and the total number of clips), the *market beat* (percentage of positive/negative opinions on a topic), the *average sentiment* (average of biased opinions on a topic). Clearly, topics are first-class citizens for most analyses that decision-makers find interesting in this field. Thus, expressive and flexible solutions are required to model topics in DM cubes.

It is almost impossible to devise a fixed schema for a subject area at design time and force all newly-discovered topics to fit that schema. However, a large part of topics can be effectively classified into levels, such as Product and Brand in our motivating example, that mostly correspond to aggregation levels in traditional business hierarchies. Like in traditional multidimensional modeling, the relationships between these topics can be captured by a hierarchy schema, to be expressed via roll-up partial orders. With the following definitions, we lay the foundations for meta-stars by formally modeling hierarchy schemata and topic hierarchies (i.e., instances of hierarchy schemata) in an implementation-independent manner.

Recalling Definition 2 of a hierarchy schema, note that:

- There is no explicit semantics attached to topic inter-relationships: the semantics of the relationship between two topics is implicitly coded by the couple of levels these topics belong to (e.g., the relationship between Product and Brand has semantics *has*).

- Inter-topic relationships are not constrained to be many-to-one.
- Disjoint hierarchies (like the ones rooted in Mobile Tech and MediaWorld in our motivating example) are supported.

Example 6 *In our motivating example it is* $L = \{\mathsf{Product}, \mathsf{Type}, \mathsf{Category}, \mathsf{Brand}, \mathsf{Component}, \mathsf{Store}, \mathsf{Chain}\}$ *and* $\mathsf{Component} \dot{\succ} \mathsf{Product} \dot{\succ} \mathsf{Type} \dot{\succ} \mathsf{Category}$*,* $\mathsf{Product} \dot{\succ} \mathsf{Brand}$*,* $\mathsf{Store} \dot{\succ} \mathsf{Chain}$ *(see also Figure 3.3, gray boxes).*

The connection between hierarchy schemata (intension) and topic hierarchies (extension) is captured by Definition 6, that also annotates roll-up relationships with their semantics.

Definition 6 (Topic Hierarchy) *A* topic hierarchy *conformed to hierarchy schema* $\mathcal{H} = (L, \succ_{\mathcal{H}})$ *is a triple of (i) an acyclic directed graph* $G = (T, U)$*, where* T *is a set of topics and* U *is a set of inter-topic roll-up relationships; (ii) a partial function* $Lev : T \rightarrow L$ *that associates some topics to levels; and (iii) a partial function* $Sem : U \rightarrow \rho$ *that associates some roll-up relationships to their semantics (with* ρ *being a list of user-defined roll-up semantics). A topic is said to be* classified *if it is associated to a level via function Lev,* unclassified *otherwise. Graph* G *must be such that*

1. *for each arc* $(t_1, t_2) \in U$ *such that* $Lev(t_1) = l_1$ *and* $Lev(t_2) = l_2$*, it is either (i)* $l_1 \dot{\succ} l_2$ *or (ii)* $l_1 \succ l_2$ *provided that* (t_1, t_2) *is not transitively implied by the other arcs in* U*;*
2. *for each couple of arcs* $(t_1, t_2), (t_3, t_4) \in U$ *such that* $Lev(t_1) = Lev(t_3)$ *and* $Lev(t_2) = Lev(t_4)$*, it is* $Sem((t_1, t_2)) = Sem((t_3, t_4))$*.*

The constraints on G are aimed at ensuring consistency between the topic hierarchy and the hierarchy schema. In particular, the first constraint states that the relationships between classified topics must not contradict the roll-up partial order; the distinction between conditions (i) and (ii) aims at avoiding transitively implied arcs while supporting missing levels (non-covering hierarchies). For instance, the arc from "Galaxy III" to "Smartphone" is allowed because $\mathsf{Product} \dot{\succ} \mathsf{Type}$, while an arc from "Galaxy III" to "Mobile Tech" is forbidden because transitively implied. The arc from "Galaxy III" to "Touchscreen" is allowed because the latter is unclassified. Multiple arcs between

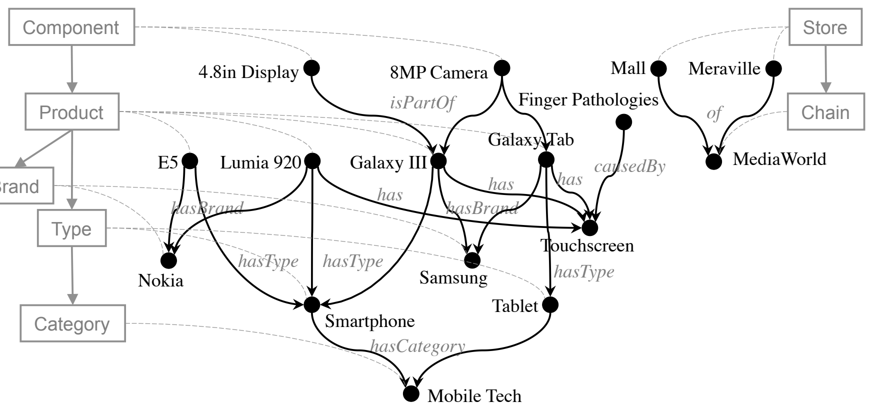


Fig. 3.3 The annotated topic hierarchy for the mobile technology subject area

topics of the same levels (non-strict hierarchies, e.g., the one between components and products) and directed paths of any length (non-onto hierarchies) are also supported. As to the second constraint, it states that all the roll-up relationships between topics of the same two levels must share the same semantics.

Finally, Definition 7 provides a compact representation for the semantics involved in any path of a topic hierarchy.

Definition 7 (Roll-up Signature) *Let G^+ be the transitive closure of G, and let t_1 and t_2 be two classified topics such that $(t_1, t_2) \in G^+$ (i.e., $Lev(t_1) \succ Lev(t_2)$). The* roll-up signature *of (t_1, t_2), denoted $RUS(t_1, t_2)$, is a binary string of $|\rho|$ bits, where each bit corresponds to one roll-up semantics and is set to 1 if at least one roll-up relationship with that semantics is part of any directed path from t_1 to t_2 in G, is set to 0 otherwise. Conventionally, $RUS(t, t)$ is a string of 0's for each t.*

Example 7 *In Figure 3.3 the topic hierarchy of Figure 3.1 is reconsidered and annotated with levels and roll-up semantics. Note that topics "Touchscreen" and "Finger Pathologies" are unclassified, and that the relationship between components and products is many-to-many. If $\rho =$* (isPartOf, hasType, hasBrand, hasCategory, has, causedBy, of)*, then $RUS($*8MP Camera, Samsung*$) =$* 1010000 *(because the two paths from "8MP Camera" to "Samsung" include roll-up relationships with semantics* isPartOf *and* hasBrand*), while $RUS($*8MP Camera, Smartphone*$) =$* 1100000*.*

The approach we propose to model on ROLAP platforms a multidimensional cube including a topic hierarchy (like the one in the example of Figure 3.2) extends a classical star schema. The fact (e.g., OCCURRENCE) can be represented by a standard fact table, and each hierarchy other than the topic one (e.g., Clip) can be represented by a standard dimension table. On the other hand, to model the topic hierarchy we use a new data structure called meta-star. Meta-stars directly support irregular hierarchies (requirement ♯1), allow different roll-up semantics to be explicitly annotated (requirement ♯4), and enable schema fluidity to be accommodated (requirement ♯2). This is achieved by coupling two tables: a *topic table* where each tuple models a topic (essentially, a dimension table extended with a column Level for storing topic levels), and a *roll-up table* that extensively represents inter-topic relationships (essentially, a navigation table extended with a column for storing roll-up signatures). De facto, with this solution we meta-model hierarchy schemata because, by including column Level in the topic table, we represent an intensional information at the extensional level.

To make meta-stars more easily integrable with standard business hierarchies (requirement ♯3) and provide better performance for queries that do not require advanced topic aggregation, the designer can fine-tune the meta-star solution by deciding which levels are to be modeled also in a classic way, i.e., by adding dedicated columns to the topic table. We will call *static* these levels (L^{stat} in Definition 8) because, like in a star schema, a schema change that involves them requires a change in the relational schema —while any change to the other levels can be accommodated at the tuple level thanks to meta-modeling.

Definition 8 (Meta-Stars) *The* meta-star *for topic hierarchy* (G, Lev, Sem) *with* $G = (T, U)$ *and static levels* $L^{stat} \subseteq L$ *includes:*

1. *A* topic table *storing one tuple for each distinct topic* $t \in T$*. The schema of this table includes a primary surrogate (i.e., DBMS-generated) key* IdT*, a* Topic *column, and a* Level *column. The tuple associated to topic* t *has* Topic $= t$ *and* Level $= Lev(t)$ *if* t *is classified,* Level $= NULL$ *otherwise. Besides, the topic table has an additional column for each static level* $l \in L^{stat}$*. If* $Lev(t) \in L^{stat}$*, the tuple associated to topic* t *has value* t *in column* $Lev(t)$*, value* $Anc^{l}(t)$ *in each column* l *such that* $l \in L^{stat}$ *and* $Lev(t) \succ l$*, and NULL elsewhere. If* $Lev(t) \notin L^{stat}$*, all additional columns in the tuple associated to* t *are NULL.*

2. *A* roll-up table *storing one tuple for each topic in* T *and one for each arc in* G^{+}*. The tuple corresponding to topic* t *has two foreign keys,* ChildId *and* FatherId*, both referencing the topic table and storing the surrogate of topic* t*, and a column* RollUpSignature *that stores* $RUS(t, t)$*, i.e., a string of 0's. The tuple corresponding to arc* (t_1, t_2) *stores in* ChildId *and* FatherId *the two surrogates of topics* t_1 *and* t_2*, while column* RollUpSignature *stores* $RUS(t_1, t_2)$*.*

TOPIC_T

IdT	Topic	Level
1	8MPCamera	Component
2	GalaxyIII	Product
3	GalaxyTab	Product
4	Smartphone	Type
5	Tablet	Type
6	MobileTech	Category
7	Samsung	Brand
8	Finger Path.	–
9	Touchscreen	–
...	...	...

ROLLUP_T

ChildId	FatherId	RollUpSignature
1	1	0000000
2	2	0000000
...	...	0000000
1	2	1000000
1	3	1000000
2	4	0100000
2	7	0010000
4	6	0001000
8	9	0000010
2	9	0000100
...	...	...
1	4	1100000
1	5	1100000
1	7	1010000
1	9	1000100
2	6	0101000
3	6	0101000
...	...	...
1	6	1101000
...	...	...

Fig. 3.4 Meta-star modeling for the mobile technology subject area when no levels are static

To enable an easier reading, we will first consider the case in which $L^{stat} = \emptyset$, i.e., the designer has chosen to make no levels static.

Example 8 *The topic and the roll-up tables for our motivating example when* $L^{stat} = \emptyset$ *are shown in Figure 3.4. The 12-th tuple of the roll-up table states that the roll-up signature of couple* (8MP Camera, Smartphone) *is* 1100000*, i.e., that the path from one topic to the other includes semantics* isPartOf *and* hasType.

As shown in Section 3.4, meta-stars yield higher querying expressiveness thanks to the presence of roll-up signatures by enabling a brand new class of semantics-aware OLAP queries. Meta-stars also better support changes in the schema of topic hierarchies, through the combined use of meta-modeling and of the roll-up table. A whole new set of emerging topics, possibly structured in a hierarchy with different levels, can be accommodated —without changing the underlying relational schema— by:

- adding new values to the domain of the Level column;
- adding tuples to the topic and roll-up tables to represent the new topics and their relationships
- extending roll-up signatures with new bits for the new roll-up semantics.

TOPIC_T

IdT	Topic	Level	Product	Type	Category
1	8MPCamera	Component	–	–	–
2	GalaxyIII	Product	GalaxyIII	Smartph.	MobTech
3	GalaxyTab	Product	GalaxyTab	Tablet	MobTech
4	Smartphone	Type	–	Smartph.	MobTech
5	Tablet	Type	–	Tablet	MobTech
6	MobileTech	Category	–	–	MobTech
7	Samsung	Brand	–	–	–
8	Finger Path.	–	–	–	–
9	Touchscreen	–	–	–	–
...	...	...	...	...	...

Fig. 3.5 Meta-star modeling for the mobile technology subject area when $L^{stat} = \{\mathsf{Product}, \mathsf{Type}, \mathsf{Category}\}$

The newly-added levels will immediately become available for querying and aggregation, with no DML operations nor modifications to the application logic needed. Similarly, any other revision of the topic hierarchy (including updates and/or removals of topics, levels or roll-up relationships) can be accommodated by operating at the tuple level.

We observe that, in the general case, the summarizability property as defined in [106] (i.e., the correct computation of aggregate values with a coarser level of detail from aggregate values with a finer level of detail) may not hold for topic hierarchies due to their inherent nature: indeed, summarization leads to incomplete results in a non-onto or non-covering hierarchy, while it leads to double counting in a non-strict hierarchy [134].

As stated by Definition 8, the designer can fine-tune the meta-star solution by deciding which levels $L^{stat} \subseteq L$ are made static. For the sake of simplicity, in the remainder we only consider the case in which the levels in L^{stat} are related by many-to-one relationships, so that they can be modeled like in a classic dimension table. More precisely, we assume that for each couple of levels $l_1, l_2 \in L^{stat}$ such that $l_1 \succ l_2$, each topic t in l_1 is connected in G to at most one topic of l_2, which we denote with $Anc^{l_2}(t)$.

Example 9 *The topic table for our motivating example when, for instance,* $L^{stat} = \{\mathsf{Product}, \mathsf{Type}, \mathsf{Category}\}$ *is shown in Figure 3.5. Note that levels* Component *and* Product *are related by a many-to-many relationship, so the presence of* Component *in* L^{stat} *would be incompatible with the presence of* Product*,* Type*, and* Category.

Choosing static levels is done at design time, based on a trade-off between efficiency and effectiveness. In particular, the designer should consider that:

- levels that are also part of the EDW should be made static in order to enable the integration with enterprise data;

- queries that do not require advanced topic aggregation would benefit (in terms of performance) from making the involved levels static;
- portions of the topic hierarchy that are likely to be involved into relevant changes are discouraged from being made static.

Remarkably, query formulation in presence of static levels is simplified, because it may be no longer necessary to use the roll-up table (see Section 3.4.4).

3.3.1 Slowly-Changing Topics and Levels

Historicization of hierarchies is a relevant issue in data warehouse design since it allows users to better focus their analyses and enables the execution of queries that use different hierarchy versions. Hierarchies subject to changes in their data are normally referred to as *slowly-changing dimensions* [95], and different techniques can be adopted to cope with them. In particular, in a star schema implementation of a cube, a *Type-2* solution is one where data versions are tracked by creating multiple tuples in the dimension table for the same natural key (e.g., several tuples in the product dimension table corresponding to different classifications into types of the same product at different times); each fact in the fact table is then referred to the specific tuple that was valid when the fact took place, so that the historical truth can easily be reconstructed by a simple star join. A more powerful solution is so-called *full logging* [63], that adds a couple of timestamps to dimension tables to explicitly model the temporal validity of each version so as to enable more expressive queries. While handling different data versions is essentially a technical problem, dealing with changes in the *schema* of hierarchies is still a research issue, with only a few proposed solutions in the literature (e.g., [60]).

Although meta-stars natively support data and schema changes, keeping track of the different versions requires some further expedient. First of all we observe that, thanks to meta-modeling and differently from traditional star schemata, meta-stars can track also schema changes using the same solutions devised for slowly-changing dimensions. This means that not only data changes (i.e., creation of a new member, deletion of a member, and inter-member relationship update), but even schema changes (i.e., creation of a new level, deletion of a level)[2] can be tracked in a meta-star without affecting the schema of topic and roll-up tables.

Both Type-2 and full-logging solutions can be applied to meta-stars. As in star schemata, a Type-2 solution does not impact on the meta-star schema and is implemented by properly setting the ETL process only. Conversely, full logging impacts on the meta-star schema; more precisely, tracking changes in the roll-up partial order requires timestamps in the roll-up table only, while all the other operations also involve the topic table since a change in a topic/level must be reflected in all the related arcs of G^+.

Example 10 *A full-logging solution for our motivating example is shown in Figure 3.6. On Jan. 31, 2014 a change in the hierarchy schema occurred: level* SubCategory *was introduced and topic "Smartphone" was moved from* Type *to* SubCategory*. A new tuple with* IdT *10 was added to* TOPIC_T*, while the previous version (the tuple with* IdT *4) run out of validity; the related arcs in* ROLLUP_T *were updated accordingly.*

Though Type-2 and full-logging solutions are more powerful when applied to meta-stars, they should be used carefully because their impact on cardinality of roll-up tables is very strong. Indeed, a roll-up table explicitly stores the transitive closure of inter-topic relationships, so any change in a topic, a level, or an arc may affect several tuples.

3.4 Querying Meta-Stars

In this section we show how meta-stars support OLAP queries with increasing expressiveness and complexity, starting from queries using only static levels to end-up with semantics-aware queries.

OLAP queries normally return the values of one or more measures aggregated according to a group-by set including a few hierarchy levels, optionally filtered according to a selection. For instance, with reference to the conceptual schema depicted in Figure 3.2, a possible query could return the total number of positive occurrences of topics "Smartphone" and "Mediaworld" in clips written in Italian during each month of 2014. In this case the measure returned is positiveOcc, the query group-by is {Topic,Month}, and

[2]Though the roll-up partial order between levels is part of the hierarchy schema, its historicization is handled at the instance level in both stars and meta-stars; while from the extensional point of view inter-level relationships can be reconstructed from the relationships between level members, from the intensional point of view they are explicitly stored only in meta-data repositories, not in dimension table schemata.

TOPIC_T

IdT	Topic	Level	From	To	Master
1	8MPCamera	Component	Jan 01 2014	-	1
2	GalaxyIII	Product	Jan 01 2014	-	2
3	GalaxyTab	Product	Jan 01 2014	-	3
4	Smartphone	Type	Jan 01 2014	Jan 31 2014	4
5	Tablet	Type	Jan 01 2014	-	5
6	MobileTech	Category	Jan 01 2014	-	6
7	Samsung	Brand	Jan 01 2014	-	7
8	Finger Path.	–	Jan 01 2014	-	8
9	Touchscreen	–	Jan 01 2014	-	9
10	Smartphone	SubCat	Feb 1 2014	-	4
...	...	......	...	...	...

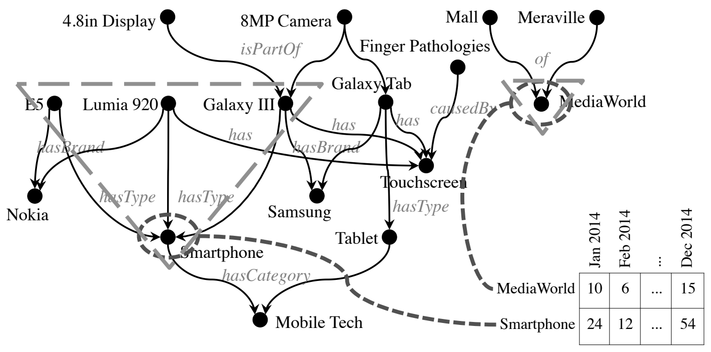
ROLLUP_T

ChildId	FatherId	RollUpSignature	From	To
1	1	0000000	Jan 01 2014	-
...	...	0000000	...	...
1	2	1000000	Jan 01 2014	-
2	4	0100000	Jan 01 2014	Jan 31 2014
2	10	0100000	Feb 01 2014	-
4	6	0001000	Jan 01 2014	Jan 31 2014
10	6	0001000	Feb 01 2014	-
...	...	...	...	...
1	4	1100000	Jan 01 2014	Jan 31 2014
1	10	1100000	Feb 01 2014	-
...	...	...	...	...

Fig. 3.6 Full-logging solution for the mobile technology subject area

the selection predicate is Topic IN {'Smarthpone','MediaWorld'} AND Year = '2014' AND Language = 'Italian'. From a conceptual point of view, answering an OLAP query of this type amounts to properly building groups of members for each dimension of the underlying multidimensional schema, then aggregating measure values for each combination of these groups. For instance, the query above is answered by building two groups of topics (the first including only member "Smartphone", the second including only member "Mediaworld" as shown in Figure 3.7, dark gray circles) and 12 groups of clips, each including all the Italian clips written during a single month of 2014; then, the values of measure positiveOcc for the facts related to each of the 2×12 couples of groups are summed together and returned, possibly in the form of a pivot table (see Figure 3.7, bottom-right).

While the semantics explained above for OLAP queries is commonly understood and shared, in presence of irregular, fluid, and semantically-rich hierarchies —such as the topic one— some further possibilities arise. In particular, since facts can also be associated to non-leaf topics, multiple semantics of aggregation can be made available to users. For instance, computing the number of occurrences of "Smartphone" may either mean considering only the UGC mentioning the word "Smartphone", or also considering the UGC mentioning products of type smartphones (such as Galaxy III),



	Jan 2014	Feb 2014	...	Dec 2014
MediaWorld	10	6	...	15
Smartphone	24	12	...	54

Fig. 3.7 Topic groups in a query without topic aggregation (dark gray circles) and with semantic topic aggregation (light gray triangles)

or also considering the UGC mentioning a component of a product of type smartphone (such as 8MP Camera). To deal with these alternative semantics, we need to extend the definition of OLAP query, which is done in the following. To keep the formalism simple, we will formalize only the part of a query that determines how groups of topics are created (which, with a slight abuse of terminology, we will call *topic query*), while the building of groups of members for the other dimensions (in our example, the groups of clips) and the subsequent aggregation of facts will still be the same of a standard OLAP query as outlined above.

Definition 9 (Schema-Aware Topic Query) *Given topic hierarchy* (G, Lev, Sem) *with* $G = (T, U)$*, a* schema-aware topic query q *is a triple of (i) a* group-by component*, that is a level* $l \in L$*; (ii) an optional* selection*, that takes the form of a conjunction of Boolean predicates of type* $l' = \bar{t}$ *(where* $l \succ l'$ *and* $Lev(\bar{t}) = l'$*); and (iii) a* semantic filter θ *consisting of a subset of allowed roll-up semantics, coded as a binary string of* $|\rho|$ *bits (where each bit corresponds to one roll-up semantics and is set to 1 if the corresponding roll-up semantics is allowed, to 0 otherwise).*

A schema-aware topic query works much like a classical OLAP query, except for the semantic filter. Its interpretation is that of building a set Φ of groups of topics including a group $\phi_{t_i} = \{t \in T : (RUS(t, t_i) \ \& \ \theta) = RUS(t, t_i)\}$ [3] for each topic t_i that has level l and satisfies the selection (a topic t_i satisfies predicate $l' = \bar{t}$ if there is a directed path in G from t_i to $\bar{t}$). While the group-by component and the selection determine which groups will be part of Φ, the semantic filter determines the composition of each group. In particular:

[3]Symbol "&" denotes the bitwise AND operator.

- Queries with $\theta = 00\ldots0$ are called *queries without topic aggregation*, because the group for topic t_i only includes t_i.
- Queries with $\theta = 11\ldots1$ are called *queries with full topic aggregation* because, for each group, they require navigating the topic hierarchy to aggregate the occurrences of a set of related topics; in this case there is no filter on roll-up semantics, i.e., all topics t from which t_i can be reached in G are included in the group for t_i regardless of the semantics of the roll-up relationship from t to t_i.
- In all the other cases, we will talk of *queries with semantic topic aggregation* because only some roll-up semantics will be considered. In particular, condition $(RUS(t, t_i)\ \&\ \theta) = RUS(t, t_i)$ allows for using the semantic filter θ as a mask, because it only returns true for the couples (t, t_i) whose roll-up semantics is 0 in (at least) all the positions where θ is 0, i.e., it includes in aggregation the paths in G^+ that include only roll-up relationships whose semantics is allowed in θ.

However, not all topics in T belong to a level, so there is a need for a further class of queries that work independently of the hierarchy schema. In these queries, the topics of interest are explicitly listed in the group-by component:

Definition 10 (Schema-Free Topic Query) *Given topic hierarchy* (G, Lev, Sem) *with* $G = (T, U)$*, a* schema-free topic query q *is a couple of (i) a* group-by component*, that is a set of topics* $T' \subseteq T$*, and (ii) a* semantic filter θ *consisting of a subset of allowed roll-up semantics, coded as a binary string of* $|\rho|$ *bits (where each bit corresponds to one roll-up semantics and is set to 1 if the corresponding roll-up semantics is allowed, to 0 otherwise).*

The interpretation of a schema-free query is that of building a set Φ of groups of topics including a group $\phi_{t_i} = \{t \in T : (RUS(t, t_i)\ \&\ \theta) = RUS(t, t_i)\}$ for each topic $t_i \in T'$. The composition of groups is determined like for schema-aware queries, so the same distinction based on topic aggregation can be made.

Example 11 *A schema-free query without topic aggregation for our motivating example is the one with group-by component* $T' = \{\text{Smartphone, MediaWorld}\}$ *and semantic filter* 0000000 *(no roll-up semantics allowed) that returns* $\Phi = \{\{\text{Smartphone}\}, \{\text{Mediaworld}\}\}$ *(see Figure 3.7, dark gray circles). Using the same group-by component and semantic filter* 0100000 *(only* hasType *semantics is allowed), the query is with semantic topic aggregation and returns* $\Phi = \{\{\text{Smartphone}, \text{E5, Lumia 920}, \text{Galaxy III}\}, \{\text{Mediaworld}\}\}$ *(see Figure 3.7, light gray triangles). On the other hand, a possible schema-aware query with semantic aggregation is the one with group-by* Type*,* *selection* Category = 'Mobile Tech'*, and semantic filter* 1100000*, that returns* $\Phi = \{\{$Smartphone, E5, Lumia 920, Galaxy III, 4.8in Display, 8MP Camera$\}$, $\{$Tablet, GalaxyTab, 8MP Camera$\}\}$.

Examples of the SQL formulation on meta-stars for different types of queries will be given in the following three subsections. The last subsection will discuss SQL formulation when some levels are static.

3.4.1 Translating Group-by Components into SQL

The group-by component T' of a schema-free query is translated into SQL by including the Topic column into the GROUP BY clause and adding a predicate in the WHERE clause to select the topics in T'. For instance,

```
SELECT     TOPIC_T.Topic, SUM(FT.totalOcc)
FROM       TOPIC_T, DTCLIP, FT
WHERE      FT.IdT = TOPIC_T.IdT AND FT.IdC = DTCLIP.IdC AND
           TOPIC_T.Topic IN ('Touchscreen', 'Finger Pathologies') AND DTCLIP.Date = '06/22/2013'
GROUP BY   TOPIC_T.Topic;
```

is a query without topic aggregation that returns the total number of occurrences for two topics on June 22, 2013 (DTCLIP is a separate dimension table storing clips, see Figure 3.2).

In a schema-aware query, the predicate on Topic is replaced by one on Level. For instance, the query with group-by component Brand (number of total occurrences for each brand) is formulated as follows:

```
SELECT     TOPIC_T.Topic, SUM(FT.totalOcc)
FROM       TOPIC_T, DTCLIP, FT
WHERE      FT.IdT = TOPIC_T.IdT AND FT.IdC = DTCLIP.IdC AND
           TOPIC_T.Level = 'Brand' AND DTCLIP.Date = '06/22/2013'
GROUP BY   TOPIC_T.Topic;
```

3.4.2 Translating Semantic Filters into SQL

In both queries shown in Section 3.4.1 the topic hierarchy is not navigated, i.e., only occurrences of the very topics of interest are counted ($\theta = 00\ldots0$). Hence, those queries can be formulated on the topic table without involving the roll-up table. Conversely, in a query with full topic aggregation ($\theta = 11\ldots1$) the topic hierarchy is extensively navigated, i.e., each topic of interest is considered together with its descendants when computing the number of occurrences, so the roll-up table must be joined with the topic table. For instance, this is the case for the talking volume analysis of Example 5, that can be expressed as a schema-free query with $T' = \{\text{Mobile Tech}\}$ and returns the total number of occurrences for "Mobile Tech" and all its descendants:

```
SELECT  SUM(FT.totalOcc)
FROM    TOPIC_T, ROLLUP_T, DTCLIP, FT
WHERE   FT.IdT = ROLLUP_T.ChildId AND ROLLUP_T.FatherId = TOPIC_T.IdT AND
        FT.IdC = DTCLIP.IdC AND
        TOPIC_T.Topic = 'Mobile Tech' AND DTCLIP.Date = '06/22/2013';
```

(no GROUP BY clause is necessary here because a single topic is selected).

While the query above could also be formulated on a classic star schema extended with either a navigation table or a parent-child dimension table to model recursion, a query with semantic topic aggregation uses the semantic filter to determine the way the topic hierarchy is navigated so as to produce custom aggregations. For instance, this is the case with the brand reputation analysis of Example 5, that is expressed as a schema-aware query and returns the number of positive and negative occurrences of each brand and of its products:

```
SELECT     TOPIC_T.Topic, SUM(FT.positiveOcc), SUM(FT.negativeOcc)
FROM       TOPIC_T, ROLLUP_T, FT
WHERE      FT.IdT = ROLLUP_T.ChildId AND ROLLUP_T.FatherId = TOPIC_T.IdT AND
           TOPIC_T.Level = 'Brand' AND
           BITAND(ROLLUP_T.RollUpSignature,001000) = ROLLUP_T.RollUpSignature
GROUP BY   TOPIC_T.Topic;
```

A similar (but schema-free) query is the one for health rumors analysis, that returns the negative occurrences for touchscreens and the related pathologies:

```
SELECT  SUM(FT.negativeOcc)
FROM    TOPIC_T, ROLLUP_T, FT
WHERE   FT.IdT = ROLLUP_T.ChildId AND ROLLUP_T.FatherId = TOPIC_T.IdT AND
        TOPIC_T.Topic = 'Touchscreen' AND
        BITAND(ROLLUP_T.RollUpSignature,000001) = ROLLUP_T.RollUpSignature;
```

3.4.3 Translating Selections into SQL

For a schema-aware query with selection, aliases must be introduced to use different "versions" of the topic table and the roll-up table must be used to establish a relationship between the topics in the group-by level and those in the selection. For instance, the query below computes the average sentiment for each type of category "Mobile Tech":

```
SELECT     T2.Topic, AVG(FT.avgSentiment)
FROM       TOPIC_T T1, TOPIC_T T2, ROLLUP_T R, FT
WHERE      FT.IdT = R.ChildId AND T1.IdT = R.FatherId AND R.ChildId = T2.IdT AND
           T1.Topic = 'Mobile Tech' AND T2.Level = 'Type'
GROUP BY   T2.Topic;
```

In a query with semantic topic aggregation, also the roll-up table must be aliased to support navigations in both directions. For instance, the query below computes the average sentiment for each type of category "Mobile Tech" and all its subtopics:

```
SELECT    T2.Topic, AVG(FT.avgSentiment)
FROM      TOPIC_T T1, ROLLUP_T R1,
          TOPIC_T T2, ROLLUP_T R2, FT
WHERE     FT.IdT = R2.ChildId AND R2.FatherId = T2.IdT AND
          T1.IdT = R1.FatherId AND R1.ChildId = T2.IdT AND
          T1.Topic = 'Mobile Tech' AND T2.Level = 'Type'
GROUP BY  T2.Topic;
```

3.4.4 Impact of Static Levels

Clearly, static modeling only impacts on the SQL formulation of schema-aware queries that use levels in L^{stat}. So, for instance, if $L^{stat} = \{\mathsf{Product}, \mathsf{Type}, \mathsf{Category}\}$ like in Example 9, the total number of negative occurrences for each mobile tech type in Italian clips during the last month can be simply obtained by including level Type in the GROUP BY clause and adding a selection on Category in the WHERE clause:

```
SELECT    TOPIC_T.Type, SUM(FT.negativeOcc)
FROM      TOPIC_T, DTCLIP, FT
WHERE     FT.IdT = TOPIC_T.IdT AND FT.IdC = DTCLIP.IdC AND
          TOPIC_T.Level = 'Type' AND TOPIC_T.Category = 'Mobile Tech' AND
          DTCLIP.Month = ' Jan 2014' AND DTCLIP.Language = 'Italian'
GROUP BY  TOPIC_T.Type;
```

Predicate Level = 'Type' is used to discard the group of tuples where Type IS NULL and to avoid topic aggregation. Some control on topic aggregation can be applied, limited to static levels, using the WHERE clause. So, for instance, the following query has an implicit semantic filter $\theta = 0100000$ because it also counts the occurrences of product topics:

```
SELECT    TOPIC_T.Type, SUM(FT.negativeOcc)
FROM      TOPIC_T, DTCLIP, FT
WHERE     FT.IdT = TOPIC_T.IdT AND FT.IdC = DTCLIP.IdC AND
          TOPIC_T.Level IN ('Type', 'Product') AND TOPIC_T.Category = 'Mobile Tech' AND
          DTCLIP.Month = ' Jan 2014'
GROUP BY  TOPIC_T.Type;
```

How to automatically and transparently translate into SQL a query expressed by a user through a graphical OLAP-like interface is itself a research problem. In our prototypical implementation, query translation is simplified by the adoption of a drastic rule that formulates in static terms all schema-aware queries with full topic aggregation and that only involve levels in L^{stat}, as seen in the example above. For

Table 3.4 Features of meta-stars for testing

Topic hier.	♯ *topics* = |TOPIC_T|	*Fan-out*	*Hier. height*	|ROLLUP_T|
$G1$	106	4	4	562
$G2$	658	8	4	4002
$G3$	27306	4	8	318578

future implementations we envision a component (to be plugged into the OLAP engine) capable of writing an optimized SQL formulation of each query based on data statistics and on the cost model described in Section 3.5.

3.5 Query Execution Plans and Cost Model for Meta-Stars

As made clear in the previous sections, meta-stars entail higher flexibility and expressiveness than classical star schemata both from the point of view of the hierarchy structures supported and from that of the queries enabled. However, this also leads to extra costs in terms of storage space and, in some cases, query execution time. To quantitatively evaluate the querying efficiency of meta-stars vs. the one of traditional star schemata (see Section 3.6), in this section we discuss the main execution plans for topic queries and present a cost model for these plans.

From the point of view of hierarchy structures, we recall that meta-stars natively support irregular, fluid, and semantically-rich hierarchies. To enable a meaningful meta-star vs. star comparison, we restrict our attention to covering and strict topic hierarchies (i.e., no levels missing and no many-to-many relationships) and we use no static levels. We implemented both meta-stars and stars on the Oracle 11g RDBMS and we populated them with three different topic hierarchies as shown in Table 3.4 (in the star implementation, topics are stored into a traditional dimension table DT). All topic hierarchies are height-balanced but have different height and fan-out (i.e., number of subtopics for each topic); clearly, the number of topics and the size of the roll-up table increase exponentially with the hierarchy height. Two fact tables, storing 1 and 10 millions of tuples respectively, were also created. B$^+$-tree indexes were created on all foreign keys, on the Level and Topic columns, and on all columns of DT used in group-by's or selection; no materialized views were created.

From the point of view of the queries supported, we summarize in Table 3.5 the query types that can be expressed on star schemata. Plans for queries without aggregation are the same for stars and meta-stars, so they will not be further discussed; we will

Table 3.5 Query types supported by star schemata

	without topic aggreg.	with full topic aggreg.	with semantic topic aggreg.
schema-free	partially	partially	no
schema-aware	yes	yes	no

focus our evaluation on queries with (semantic or full) topic aggregation, that represent the worst case for meta-stars because they require access to the roll-up table. On the topic hierarchies described above we ran 60 queries with different features and analyzed the plans followed by Oracle. It turned out that the set of execution plans adopted is quite restricted, so we focused on the 6 most representative ones, that cover 80% of queries and are depicted in Figure 3.8. Plans for meta-stars and stars are shown on the left and on the right, respectively; interestingly, the optimizer often chooses the same type of access in both cases. The two top plans are those followed for schema-free queries; the middle ones for schema-aware queries without selection; the bottom ones for schema-aware queries with selection. Importantly, the top-right plan refers to the subclass of schema-free queries that can be expressed on a star schema, i.e., those where the topic list includes one or more classified topics belonging to the same level. Some specific comments follow:

- Hierarchy data are always accessed before fact data. Using index access vs. full scan obviously depends on selectivity (more precisely: on the number of topic groups for `ROLLUP_T`, on the total number of topics aggregated for `FT`); nested loops join is always associated to indexed access, hash join to full scans.
- Index access is adopted only when selectivity is very strict (roughly, when less than 1% of tuples is selected).
- Differently from a query with full topic aggregation, in a query with semantic topic aggregation the access to the roll-up table also entails a selection of tuples based on the semantic filter (not shown in the figure for simplicity).
- Due to the presence of the roll-up table, cardinality estimations of join results made by the DBMS are less accurate on meta-stars than on stars, hence, when using meta-stars there is higher probability that a non-optimal plan is picked.

Based on the plans previously described we devised a cost model to simulate the DBMS behavior. Tables 3.6 and 3.7 show the cost formulae (derived from [52]) for the basic operations used in the plans; the cost of a complete plan is estimated by choosing the cheapest alternative for each step of the plan and properly assembling

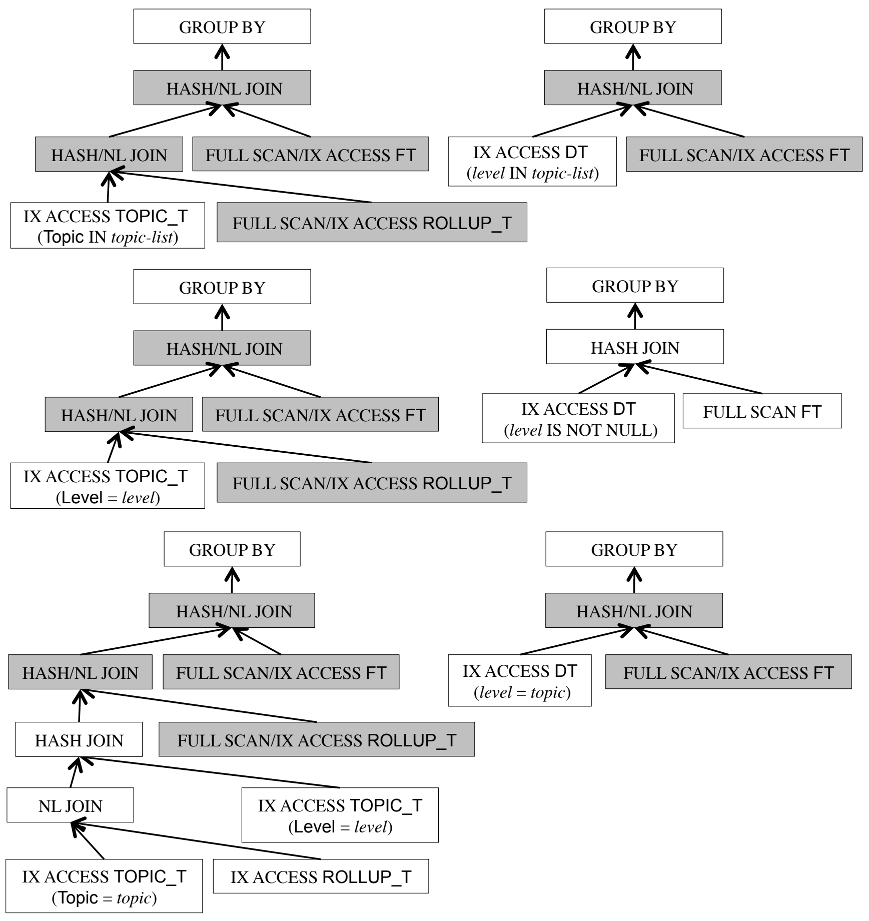

GROUP BY
HASH/NL JOIN
HASH/NL JOIN
FULL SCAN/IX ACCESS FT
IX ACCESS TOPIC_T (Topic IN topic-list)
FULL SCAN/IX ACCESS ROLLUP_T
GROUP BY
HASH/NL JOIN
IX ACCESS DT (level IN topic-list)
FULL SCAN/IX ACCESS FT
GROUP BY
HASH/NL JOIN
HASH/NL JOIN
FULL SCAN/IX ACCESS FT
IX ACCESS TOPIC_T (Level = level)
FULL SCAN/IX ACCESS ROLLUP_T
GROUP BY
HASH JOIN
IX ACCESS DT (level IS NOT NULL)
FULL SCAN FT
GROUP BY
HASH/NL JOIN
HASH/NL JOIN
FULL SCAN/IX ACCESS FT
HASH JOIN
FULL SCAN/IX ACCESS ROLLUP_T
NL JOIN
IX ACCESS TOPIC_T (Level = level)
IX ACCESS TOPIC_T (Topic = topic)
IX ACCESS ROLLUP_T
GROUP BY
HASH/NL JOIN
IX ACCESS DT (level = topic)
FULL SCAN/IX ACCESS FT

Fig. 3.8 Execution plans for schema-free queries (top), schema-aware queries (middle), and schema-aware queries with selection (bottom), considering a meta-star (left) and a star (right) implementation. Gray boxes represent alternative operations.

the formulae. Costs are expressed in numbers of page reads/writes. We tested our cost model on the set of 60 queries used to determine the execution plans; the average gap between the costs estimated with the model and the actual ones is lower than 10%, which is a surprisingly good result.

3.6 Evaluation

In this section we discuss how meta-stars perform both in absolute terms and with respect to traditional star schema implementations. Execution times are computed using the cost model described in Section 3.5 with a fact table of 10 millions of tuples, and considering an average read/write time of 2.27×10^{-4} seconds per disk page (as measured on the Oracle 11g RDBMS with disk page size set to 8 KB, on a 64-bits

Table 3.6 Building blocks of our cost model; for an explanation of the notation see Table 3.7

Operation	*Cost*
FULLSCAN(T)	NP_T
IXACCESS-CLUST(T,a,$pred$)	$h - 1 + \left\lceil Sel(pred) \cdot NL_{T,a} \right\rceil + \lceil Sel(pred) \cdot NP_T \rceil$
IXACCESS-UNCLUST(T,a,$pred$)	$h - 1 + \left\lceil Sel(pred) \cdot NL_{T,a} \right\rceil + \lceil NK_a \cdot Sel(pred) \cdot$ $\cdot \Phi\left(NR_T / NK_a, NP_T\right)\rceil$
NLJOIN(T_1,T_2,$pred_{T_1}$)	$Cost(T_1) + \lceil Sel(pred_{T_1}) \cdot NR_{T_1} \cdot Cost(T_2) \rceil$
HASHJOIN(T_1,T_2,$pred_{T_1}$,$pred_{T_2}$)	$NP_{T_1} + NP_{T_2} + 2(Sel(pred_{T_1}) \cdot NP_{T_1} + Sel(pred_{T_2}) \cdot NP_{T_2})$
GROUPBY(T)	$2NP_T(\lceil \log_{NB-1} NP_T \rceil + 1)$

Table 3.7 Explanation of the notation used in Table 3.6

Notation	*Meaning*
NR_T	Number of tuples in table T
NP_T	Number of disk pages for table T
NK_a	Number of distinct values of column a
NB	Main memory sort area size (in disk pages)
$NL_{T,a}$	B^+-tree index size
h	B^+-tree index height
$Sel(pred)$	Selectivity of predicate *pred*
$Cost(T)$	Access cost to table T (either full scan or indexed)

AMD Opteron quad-core 2.09GHz virtual machine with 8GB RAM and RAID 10 disk architecture, running Windows Server 2008 R2 Standard SP1). Note that meta-stars cannot be comprehensively evaluated using any existing benchmark, because all of them (e.g., TPC-DS [136]) include regular hierarchies and standard queries only. For this reason we had to create an ad-hoc benchmark including queries like the ones considered in Section 3.4.

We start by observing that for queries without topic aggregation and queries like those in Section 3.4.4, that take advantage of static levels to avoid accessing the roll-up table, performances of meta-stars are clearly the same of star schemata, so in this section we will only consider queries with topic aggregation operating on the part of meta-stars that does not include static levels. First we will consider queries with full topic aggregation; for clarity we will separately discuss the costs for accessing topics (including those for accessing the roll-up and topic tables in meta-stars, those for accessing the dimension table in stars) and those for accessing facts (including those for accessing the fact table, for joining with topic data, and for grouping the results). Then we will consider queries with semantic topic aggregation, that are not supported by star schemata. Finally, we will evaluate meta-stars from the point of view of disk storage requirements.

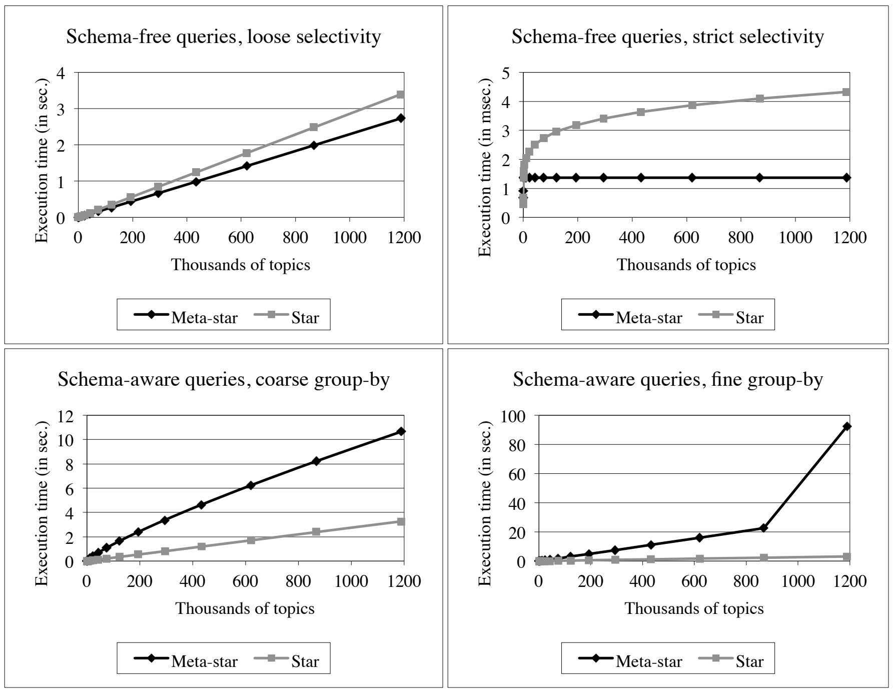


Fig. 3.9 Query execution time (only access to the topic hierarchy) vs. number of topics for different query types

3.6.1 Time (for Accessing Topics)

Figure 3.9 shows the query executions times, on both meta-stars and stars, for a six-levels topic hierarchy with fan-out ranging from 1 to 16, so as to determine an increasing number of topics from 11 to more than one million (roll-up table cardinality, i.e., number of inter-topic relationships, from 41 to more than 12 millions). All queries are with full topic aggregation. For schema-free queries we distinguish two cases: loose selectivity, where the group-by component includes one topic near to the roots of the topic hierarchy (hence with several subtopics to be aggregated), and strict selectivity, where the group-by component includes one topic near to the leafs of the topic hierarchy (hence with few subtopics to be aggregated); in both cases, to enable a star formulation we will restrict to the case where the group-by component is a classified topic. Similarly, for schema-aware queries we distinguish between coarse group-by, where the group-by level is near to the roots of the hierarchy schema (which means few groups with several topics each), and fine group-by, where the group-by level is near to the leaves of the hierarchy schema (several groups with less topics each). Time trends are similar for meta-stars and stars, because as mentioned in Section 3.5 the plan structure adopted is basically the same. Remarkably, for schema-free queries accessing stars requires more time than accessing meta-stars because the dimension table is always accessed using

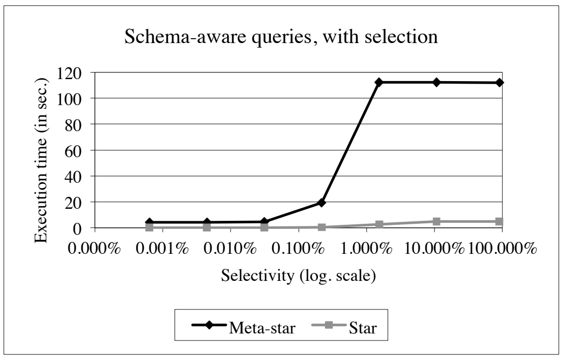


Fig. 3.10 Query execution time (only access to the topic hierarchy) vs. selectivity —worst case

an unclustered index (it is ordered on its surrogate key), while the roll-up table can be ordered on the FatherId column to enable a more convenient access. The worst case for meta-stars takes place for schema-aware queries when the number of topics is very high and the group-by level determines several groups, because in this case the DBMS accesses the (large) roll-up table via full scan.

Figure 3.10 shows the execution times for schema-aware queries with selection. We considered the worst case for meta-stars, i.e., fine group-by and high number of topics (namely, eight hierarchy levels, fan-out 7, about one million topics and 14 millions arcs), and we applied a progressively looser selection on topic groups. The chart shows that the performance gap between the two implementations is smaller for strict selections. The drastic change in the meta-star cost is determined by a change in the execution plan: when the selectivity exceeds 1.5% the optimizer moves from indexed access to full scan of the roll-up table.

3.6.2 Time (for Accessing Facts)

As to fact table access and join as well as group-by operations, the plans (hence, the execution times) are the same both when using stars and meta-stars. Figure 3.11 shows the fact table cost and the group-by cost in function of the percentage of fact table tuples to be accessed (same experimental setting used for Figure 3.10). The group-by cost is not relevant and its increase is obviously determined by the increase in the number of tuples to be grouped. The fact table cost is the predominant one; the steep increase it shows when the selectivity exceeds 0.22%, is due to the fact that the optimizer moves from indexed access to full scan of the fact table.

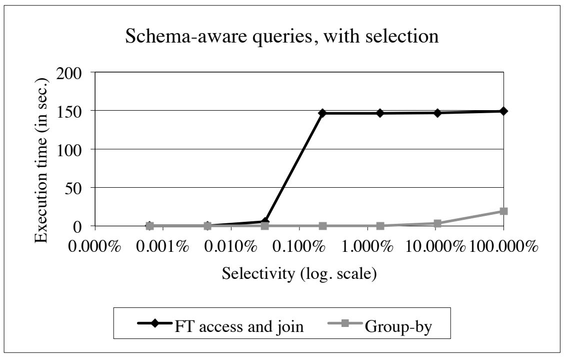


Fig. 3.11 Query execution time (only access to fact table, join, and group-by) vs. selectivity —worst case

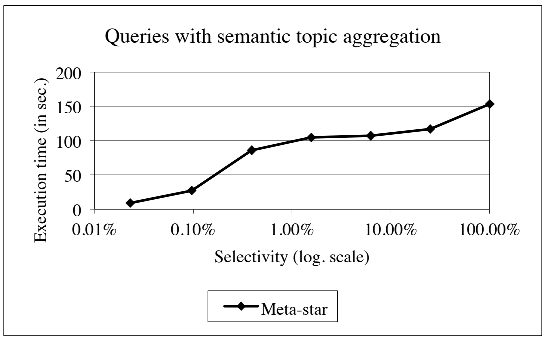


Fig. 3.12 Query execution time vs. selectivity of the semantic filter

3.6.3 Queries with Semantic Topic Aggregation

Here we deal with queries with semantic topic aggregation, that are not supported by traditional star schemata. Figure 3.12 shows the overall execution time in function of the semantic filter selectivity. Remarkably, the times displayed in this chart are the actual ones measured on the Oracle DBMS (on a hierarchy with eight levels and fan-out 4).

The chart has been obtained considering a fact table including 10 millions of facts and the $G3$ topic hierarchy whose features are described in Table 3.4. Expectedly, the execution time increases with selectivity because an increasing number of topics is accessed, however, the slope is less steep than the one in Figure 3.10; this is because semantic filtering is directly carried out on the roll-up table (see Figure 3.8, middle-left plan) without requiring further joins with the topic and roll-up tables as required by a query with selection (see Figure 3.8, bottom-left plan).

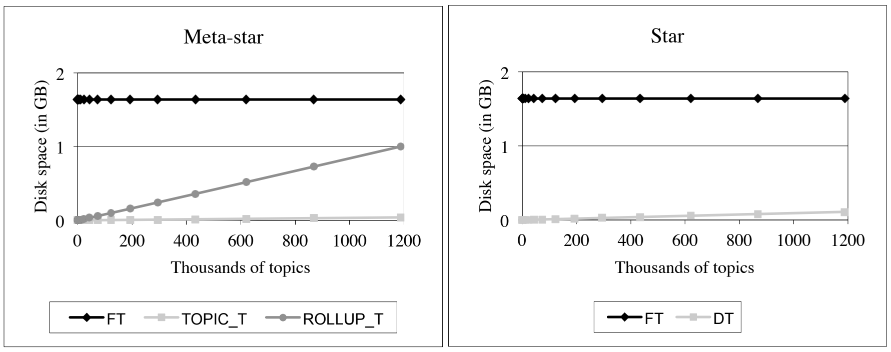


Fig. 3.13 Disk space vs. number of topics

3.6.4 Space

The toll to be paid for the impressive performance of meta-stars even in presence of large topic tables, is clearly the larger disk space they require. Figure 3.13 shows the disk space to hold the data for the test discussed above. It emerges that most of the space is used for the fact table, while the roll-up table severely impacts on the overall space only for very high numbers of topics. Actually, the number of topics monitored in real SBI applications is far from reaching such cardinality, so the right-hand halves of these charts should be considered as stress tests in this context.

The size of the roll-up table and its impact on the query execution times are also related to the topic hierarchy height. We will not include specific charts for this since the trend they show are very similar to the ones already discussed.

3.7 Conclusions

In this chapter we proposed meta-stars as an expressive solution to model topic hierarchies for SBI based on some specific requirements: irregularity and fluidity of hierarchies, integrability with business hierarchies, and semantics-aware aggregation [47–49]. Noticeably, the choice of the subset of levels to be modeled as static rules the trade-off between the fluidity of topic classification and aggregation and the efficiency of integrating UGC-related facts (accessed via topic hierarchies) with business-related facts (accessed via standard hierarchies).

To improve our approach we are currently working on the following issues:

1. *Static levels*: while in this chapter we modeled static portions of the topic hierarchies in a redundant fashion (i.e., by modeling inter-topic relationships both in a denormalized form and within roll-up tables), to improve performances it is

sometimes possible to exclude these portions from roll-up tables so as to reduce their size. Understanding under what circumstances this can be done while preserving full querying expressiveness and correctness (in particular, in presence of many-to-many relationships), so as to enable a performance-aware tuning of static levels, requires some further investigation. Delivering an optimization algorithm for writing efficient SQL formulation of user queries is also an open research problem.

2. *Topic hierarchy generation*: the frequent changes in the set of relevant topics requires that their values and relationships are continuously maintained. Though the basic topics can be automatically derived from the enterprise business hierarchies, in general they will be manually inserted, possibly by the users. Since feeding the topic and the roll-up tables appears to be a cumbersome task, we are working towards modeling the topic hierarchy through an ontology that can be automatically turned into a meta-star.

3. *Coupling SQL and OWL*: in the same direction, we are also considering the possibility of using the OWL language to directly query the topic hierarchy. This can avoid the storing of the roll-up table that, as already said, could become very large and represents the main limitation when adopting the meta-star approach on large topic hierarchies.

Chapter 4

An architectural and methodological framework for Social BI

In this chapter we dig deeper into the analysis of social data by discussing the architectural and methodological aspects of an SBI solution. Since SBI solutions can come in a variety of shapes and with different demands, it may be hard for the designer to find the right cost-benefit compromise depending on the project goals and time horizon and on the available resources. In this context, we discuss the main factors that impact this compromise aimed at providing a guideline to the design team. First we list the main architectural options and their methodological impact. Then we discuss a case study focused on an SBI project in the area of politics, aimed at assessing the effectiveness and efficiency of these options and their methodological sustainability.

4.1 Introduction

With *SBI process* we refer to the one whose phases range from web crawling to users' analyses of the results. In the industrial world, the SBI process is often implemented in the so-called *social media monitoring tools* [161], i.e., commercial tools and platforms available for the analysis of UGC, such as Brandwatch, Tracx, and Clarabridge. Their main feature is the availability of a fixed set of dashboards that analyze the data from some fixed points of view (such as topic usage, topic correlation, and brand reputation) and rely on some ad-hoc KPIs (e.g., topic counting and sentiment), so they lack in providing flexible user-driven analyses.

In the academic world, the SBI "big picture" has not been deeply investigated so far. In [43] we proposed a reference architecture and an iterative methodology for designing SBI applications, showing how its adoption can make the activities for developing and maintaining SBI processes more efficient and the SBI process itself more effective. Nonetheless, SBI projects come in a variety of shapes, characterized by different relevance and sophistication degrees for each design task and architectural component, which results in quite different demands in terms of skills, computing infrastructure, and money. Hence, finding the right cost-benefit compromise depending on the project goals, on its time horizon, and on the available resources may be quite hard for the designer.

During the last few years we have been involved in different SBI projects. In particular, in the context of the WebPolEU project we developed an SBI platform aimed at investigating the connection between politics and social media. The project used UGC written in three languages and was focused on the 2014 European Election. Based on this experience, we discuss in this chapter the main factors that impact the above-mentioned compromise aimed at providing design guidelines to the SBI design team [42]. In particular, we give the following contributions.

- In Section 4.2 we discuss the related work in the area.
- In Section 4.3 a reference architecture for SBI is described.
- In Section 4.4 we introduce WebPolEU, the project in the area of politics used as a case study in this chapter.
- In Section 4.5 we discuss, for each component of the SBI process, the main technical options and their expected impact on the project complexity and effectiveness.
- In Section 4.6 we present the result of our case study and give a quantitative characterization of the above options. and in Section 4.7 we draw the conclusions.

4.2 Related Literature

As stated in Section 4.1, only a few papers have focused on the full picture of SBI so far. Complete architectures for SBI have been proposed in [146] and [53]; in both cases, the basic blocks of the architecture have been identified, but still with a limited expressiveness. In particular, in [146] a comprehensive solution for the extraction of

Twitter streams and the enhancement and analysis of their meta-data is presented; the approach of [53] extracts sentiment data about products and their features from selected opinion websites and builds *opinion facts*. An important step towards increasing the expressiveness of SBI queries has been taken in [29], where a first advanced solution for modeling topic hierarchies has been proposed. Another step in this direction has been made in [47], where topic hierarchies are modeled by handling their dynamics and irregularity so as to enable full OLAP analyses of social data. In terms of OLAP analysis over UGC, a cube for analyzing term occurrences in documents belonging to a corpus is proposed in [101], although term categorization is very simple and does not support analyses at different levels of abstraction. In [145] the authors propose to use textual measures to summarize textual information within a cube.

As to the enabling technologies for the SBI process, a number of academic works have focused on specific issues that find application on strictly correlated fields. First of all, web crawling is a central issue in information retrieval, in whose context powerful languages to automatically and precisely capture the relevant data to be extracted were studied (e.g., [46]). In terms of semantic enrichment of raw clips and text understanding, different techniques have been studied in several areas of computer science. Whereas most of these techniques are typically tuned to perform well on a limited set of selected (web) sources, their accuracy tends to decrease when applied to a heterogeneous collection of documents extracted from multiple kinds of sources. In general, NLP approaches try to obtain a full text understanding [178], while text mining approaches rely on different techniques (e.g., n-grams) either to find interesting patterns in texts (e.g., named entities [147], relationships between topics [151], or clip sentiment [133]) or to classify/cluster them [175]. Also hybrid approaches between classical NLP and statistical techniques have been tried, either user-guided, as in [88], or automated and unsupervised, as in [53].

4.3 Architectural and Methodological Framework

The reference architecture we proposed in [43] to support the SBI process is depicted in Figure 4.1. Its main highlight is the native capability of providing historical information, thus overcoming the limitations of social media monitoring tools in handling the data reprocessing typically required by cleaning and semantic enrichment needs. In the following we briefly comment each component.

Table 4.1 Summary of main architectural options

Component	Option	Pros	Cons
Analysis	dashboard	effective summary of trends	low flexibility
	text search	detailed content analyses	increased storage
	OLAP	high flexibility	increased storage; extra ETL
	text mining	enables advanced analyses	complexity; expert analyst required
ODS	relational	clip buffering, reprocessing, and cleaning; structured	increased storage; performances
	NoSQL	clip buffering, reprocessing, and cleaning; scalability	low control of data transformation and quality
Crawling	designer-managed	good control of quality	large effort
	provider-managed	small effort	low control of quality
Sem. Enr.	crawler meta-data	enables clip classification and aggregation	some complexity in collecting
	crawler sentiment	enables analysis of sentiment; no tuning	unreliable for non-neutral clips
	inf. retrieval	enables topic occurrence analysis	low text understanding
	NLP analysis	enables analysis of sentiment; also reliable for non-neutral clips	complex tuning; affected by clipping quality
	domain expert	enables analysis of sentiment; fully reliable	costly; subjective

- The *Operational Data Store* (ODS) stores all the relevant data about clips, their authors, and their source channels; the ODS also represents all the topics within the subject area and their relationships.
- The *Data Mart* (DM) stores integrated data in the form of a set of multidimensional cubes which support the decision making process.
- The *Document-Base* stores the clips in textual form and the related meta-data to be used for text search.
- *Crawling* carries out a set of keyword-based queries aimed at retrieving the clips (and the available meta-data) that are in the scope of the subject area. The target of the crawler search could be either the whole web or a set of user-defined web sources (e.g., blogs, forums, web sites, social networks).
- *Semantic Enrichment* works on the ODS to extract the semantic information hidden in the clip texts. Such information can include its topic(s), the syntactic and semantic relationships between words, or the sentiment related to a whole sentence or to each single topic it contains.
- The *ETL* process turns the semi-structured output taken from either the crawler or the CRM into a structured form and loads it onto the ODS. Then it integrates

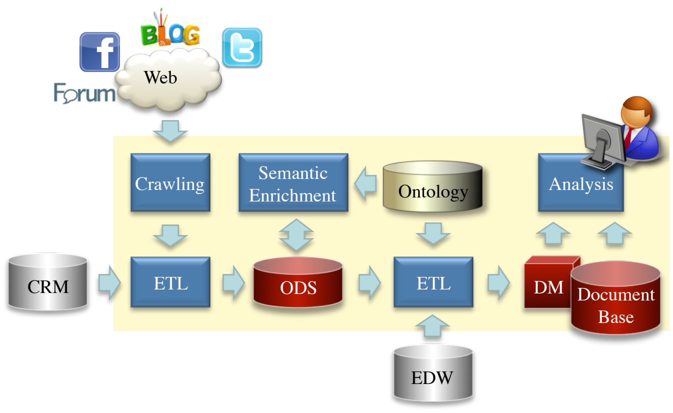


Fig. 4.1 A reference architecture for the SBI process

data about clips and topics with the business data extracted from the EDW (Enterprise Data Warehouse), and loads them onto the DM.

- *Analysis* enables users to explore the UGC from different perspectives and control the overall social mood.

From the methodological point of view, we observe that the roles in charge of designing, tuning, and maintaining each component of the SBI process may vary from project to project, and so may vary the complexity of each design activity and the control the designer and the user have over it. Specifically, as claimed in [43], SBI projects can be classified into:

- *Best-of-Breed*. A best-of-breed policy is followed to acquire tools specialized in one of the parts of the SBI process. In this case, the designer has full control of the SBI process by finely tuning all its critical parameters.
- *End-to-End*. Here, an end-to-end software/service is acquired and tuned. Designers only need to carry out a limited set of tuning activities that are typically related to the subject area, while a service provider or a system integrator ensures the effectiveness of the technical phases of the SBI process.
- *Off-the-Shelf*. This type of projects consists in adopting, typically in a *as-a-service* manner, an off-the-shelf solution supporting a set of standard reports and dashboards. The designer has little or no chance of impacting on activities that are not directly related to the analysis of the final results.

Moving from level best-of-breed to off-the-shelf, projects require less technical capabilities from designers and users and ensure a shorter set-up time, but they also allow less control of the overall effectiveness and less flexibility.

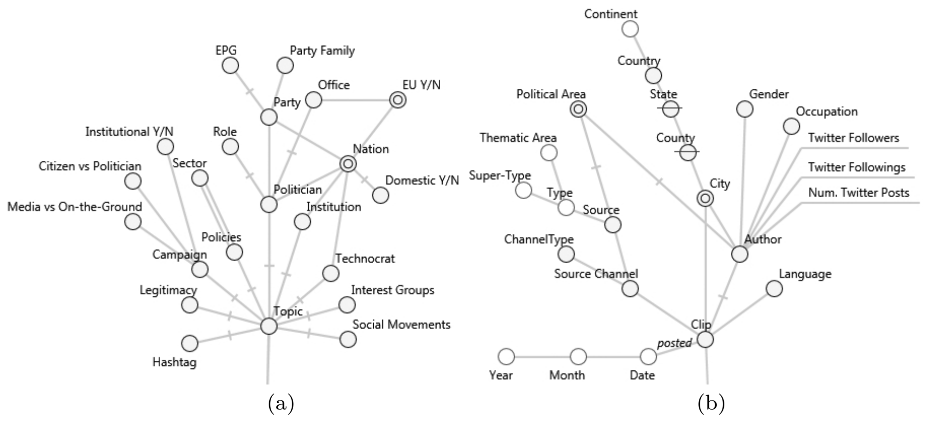


Fig. 4.2 A DFM representation of the topic (a) and clip (b) hierarchies for WebPolEU

4.4 A Case Study on EU Politics

The WebPolEU Project[1] aims at studying the connection between politics and social media. By analyzing digital literacy and online political participation, the research evaluates the inclusiveness, representativeness, and quality of online political discussion.

SBI is used in the project as an enabling technology for analyzing the UGC generated in Germany, Italy, and UK during a timespan ranging from March, 2014 to May, 2014 (the 2014 European Parliament Election was held on May 22-25, 2014). In the architecture we adopted, topics and related taxonomies are defined through Protégé; we use Brandwatch as a service for keyword-based crawling, Talend for ETL, SyN Semantic Center by SyNTHEMA for semantic enrichment (specifically, for labeling each clip with its sentiment), Oracle to store the ODS and the DM, MongoDB to store the document database for full-text search, and Mondrian as the multidimensional engine. Given the nature of the subject area, no EDW and no CRM are present in the architecture. We used the Indyco CASE tool to design the DM, and we developed an ad-hoc OLAP & dashboard interface using JavaScript, D3, and Saiku.

To enable topic-based aggregations of clips in the OLAP front-end, the classes in the domain ontology describing the subject area (that was designed together with the domain experts by classifying the topics emerged during macro-analysis) have been arranged into a *topic hierarchy* (see Figure 4.2(a)). To effectively model the topic hierarchy, taking into account its specificities (it is heterogeneous, dynamic, non-onto, non-covering, and non-strict), the meta-star approach has been used [47].

[1]http://webpoleu.net

4.5 Architectural Options

The techniques to be used to support the processes appearing in Figure 4.1 may change depending on the context of each specific project, resulting in heavier or lighter architectures. In the light of our experience with SBI projects of different types, in the following subsections we discuss the main options available to the design team, as well as their methodological impact.

4.5.1 Analysis

A component for analyzing the UGC is always present in SBI architectures, and it can take a variety of shapes characterized by quite different capabilities:

- **Dashboards** effectively summarize the trends and behaviors within the subject area, but only support a small number of predefined views and navigations (e.g., by topic or by geography).
- **Text search** enables very detailed analyses of the UGC up to the single-clip level, by supporting searches on both the clip text and its related meta-data.
- **OLAP** provides very flexible analyses based on the multidimensional metaphor, which enables users to understand in depth the market mood by slicing and drilling according to different dimensions such as time, topic, geography, UGC source, and the related hierarchies.
- **Text mining** enables advanced analyses on textual data such as clip clustering and new topic discovery [50].

Standard commercial SBI systems normally provide only dashboards and text search, and only a few of them support text mining (e.g., SAS Text Miner and Temis). Providing OLAP capabilities requires an additional layer of multidimensional data to be added to the architecture, as well as additional ETL processes that obviously increase the overall complexity. In the WebPolEU implementation, a set of cubes (see Figure 4.3) are provided; noticeably, their schemata are largely project-independent, except for the topic hierarchy whose content and structure strictly depends on the domain ontology. Besides, to enable text search functionalities, the relational ODS is coupled with a document-oriented database.

4.5.2 ODS

In principle, the ODS component could even be dropped (in which case, the two ETL processes in Figure 4.1 could be unified) since the users do not access it directly. However, the presence of the ODS —in compliance with three-tier data warehouse architectures— is warmly recommended in SBI for several reasons:

- **Buffering and early analysis**. Crawling and semantic enrichment activities have a very different timing due to the complexity of enrichment. The ODS can be seen as the *buffer* that makes the two phases independent of each other, so as to give users the possibility of timely accessing a subset of information that (i) enables some relevant early analyses; (ii) has a key methodological role for tuning the crawling and enrichment processes at the next iteration. Such information ranges from the clip meta-data returned by the crawler (e.g., source, author, and clip count) to some *quick-and-dirty* semantic enrichment.

- **Clip reprocessing**. Semantic enrichment is inherently an iterative process, due to changes in topics and in the domain ontology which may occur even months after the clips were retrieved. Storing clips in an ODS, where they can be easily queried at any time, makes reprocessing feasible.

- **Data cleaning**. It is well known that data cleaning techniques are more effective when applied to materialized data rather than when they are applied on-the-fly to a data flow. In the specific case of SBI, cleaning is necessary, for instance, to correct wrong character sequences, to repair enrichment/crawling errors which may produce wrong or incomplete results, and to filter off-topic clips based on relevance measures computed on both text and meta-data.

In our prototypical implementation, a relational ODS is used to store clips and their meta-data together with topics and their relationships. However, other alternatives could be explored. Choosing a NoSQL repository is mainly a matter of scalability, strictly related to the quantity of data to be stored and processed. In WebPolEU, about 10 millions of raw clips were retrieved and about 1.3 billions of entity occurrences were produced by semantic enrichment. Although this size is still manageable with traditional RDBMS, larger projects may make NoSQL solutions more attractive. In our experience, the main advantages of using an RDBMS are:

- The ODS plays the role of a hub for ETL data flows, and its tuples are subject to several updates to trace the process steps. This determines a transactional workload which is better handled if the ACID properties are preserved.

- The presence of a well-defined, structured, and normalized schema is very useful to process the clip meta-data.

4.5.3 Crawling

The crawling component is the main entry point to the SBI system for all the data that will be analyzed. From a technical point of view, the problem with crawling is to ensure that a satisfactory compromise is achieved between retrieving too much content (which adds harmful noise and leads to useless efforts during semantic enrichment and analysis, as well as during all test activities) and retrieving too little content (which may dramatically reduce the reliability of analysis results). The two drivers that can be used to tune this compromise are *clipping* and *querying*.

Clipping is the process through which an indexed web page is parsed and every building section of the page itself is identified in order to exclude from the information extraction process all those contents that are not relevant and do not contain any useful information [179, 180]. Bad clipping implies that the crawler will introduce into the system UGC filled with useless text such as hyperlinks, which will make the information almost incomprehensible for the semantic enrichment engine and often also for a human being —and also negatively affect the performance and quality of semantic enrichment activities.

Besides an accurate page clipping, the other ingredient for an effective crawling is a proper set of crawling queries. The standard way to identify relevant UGC from the web is by using Boolean keyword-based queries, where keywords considered as relevant or descriptive for the project scope are combined using different operators to instruct the crawler on the topics we are interested in and the ones that are out of scope. The operators typically provided by crawlers can be roughly classified into *Boolean* (e.g., AND, OR, NOT), *proximity* (e.g., NEAR/n), *meta* (e.g., country, site, author); wildcards are supported.

In the light of the above, it is apparent that managing and tuning the specific features of crawling to ensure its effectiveness is a burdensome and very time-consuming task. Noticeably, the roles in charge of these activity drastically depend on the project type as defined in Section 4.3: (i) in best-of-breed projects, all technical activities are in charge of the designer; (ii) in end-to-end projects, crawling templates are created and maintained by a service provider who is responsible of the clipping quality, but crawling queries are managed by the designer; (iii) in off-the-shelf projects, designers and users jointly carry out macro-analysis, but all other activities are largely in the hands of the service provider —which means that the designer can control the crawling effectiveness only to a limited extent [43]. So, from a project management point of view, the main trade-off involved in crawling is between (i) *do it yourself —but it will take a lot of time and effort* and (ii) *let the provider do it for you —but then you will have little control on the overall quality.*

4.5.4 Semantic Enrichment

The semantic enrichment process is maybe the one showing the widest spectrum of possible technological alternatives, with a very relevant impact on the expressiveness of the supported OLAP queries and on the accuracy of the results. Basic semantic enrichment techniques may be sufficient if users are only interested in analyzing raw data (e.g., counting the number of occurrences of each topic in the UGC); in some cases (for instance, for languages —like German— whose inherent complexity discourages automated analysis and interpretation of sentences), semantic enrichment is done by manually tagging each sentence with its sentiment. In our WebPolEU project, semantic enrichment is achieved as the combination of different (and possibly alternative) techniques:

- **Crawler meta-data**: each clip is equipped with several meta-data, which are mainly related to the web source (e.g., http address and web site nation), to the author (e.g., name, sex, and nationality), and to the clip itself (e.g., its language). As shown in Figure 4.2(b), in WebPolEU these meta-data are used to build the clip hierarchy.

- **Information retrieval**: the content of the clips can be analyzed by searching the raw text for user-defined topics (or their aliases). Although this type of analysis is not based on an in-depth comprehension of clip semantics, it returns a quick and valuable first level of analysis of the texts. In particular it allows to count the number of occurrences of a given topic and the number of co-occurrences of a pair of topics in a clip. Figures 4.3(a,b) show the IR Clip and IR Topic Occurrence cubes of the DM; each event of IR Clip represents a clip and its topics, while each event of IR Topic Occurrence represents the occurrence of a single topic within a clip.

- **Crawler sentiment**: the crawler often provides its own sentiment score. In WebPolEU we use Brandwatch, whose sentiment analysis module is based on mining rules developed for each supported language and assigns a single sentiment

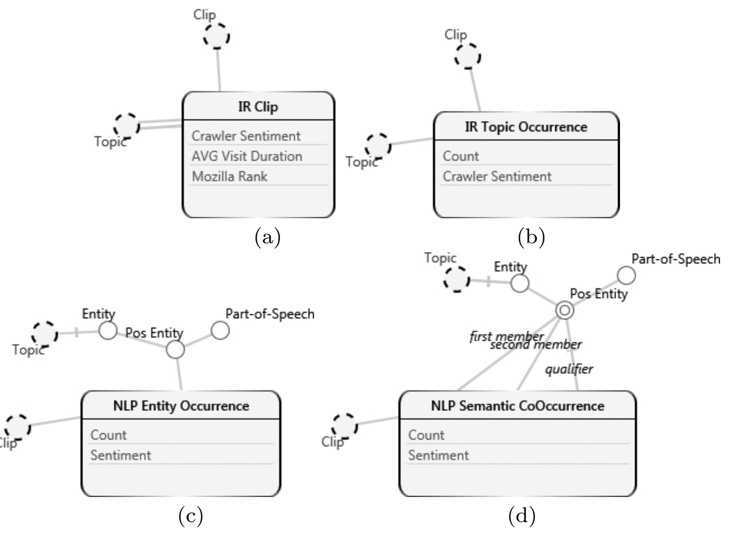


Fig. 4.3 A DFM representation of IR and NLP cubes. Topic and clip hierarchies have been hidden to simplify the picture

to each clip. In both the IR Clip and IR Topic Occurrence cubes, the crawler sentiment for each clip is modeled as a measure.

- **NLP analysis**: it is the deepest analysis raw texts undergo. As shown in Section 4.3, the commercial system SyN Semantic Center is in charge of extracting the single entities, their part-of-speech, and their semantic relationships from the raw data. Two cubes are derived through NLP analysis. The first one, NLP Entity Occurrence (Figure 4.3(c)), differs from IR Topic Occurrence since it also stores all the entities (i.e., lemmas, annotated with their part-of-speech) discovered in the text. The second one, NLP Semantic CoOccurrence (Figure 4.3(d)), stores semantic relationships and explicitly models couples of topics/entities in the same sentence together with an optional qualifier (e.g., Angela Merkel had lunch with Matteo Renzi).
- **Domain expert**: differently from social media monitoring solutions, SBI projects allow additional meta-data to be provided by domain experts by means of the domain ontology coded in the topic hierarchy (see Figure 4.2(a)) and by additional meta-data to be added to the other hierarchies.

4.6 Case Study Analysis

Carrying out an SBI project requires to find the right trade-off between its effectiveness, efficiency, and sustainability, respectively expressed in terms of correctness of the

Table 4.2 Number of topic occurrences detected by IR and NLP (a) and number of positive, neutral, and negative clips detected by NLP, by IR, and agreed upon by NLP and IR (b)

	ITA	ENG
# Topic Occ. NLP	14 215 K	23 399 K
# Topic Occ. IR	15 401 K	25 006 K
# Shared Occ.	12 922 K	21 497 K

(a)

	ITA			ENG		
Sentiment	NLP	IR	Agreed	NLP	IR	Agreed
Pos.	566 K	36 K	19 K	1090 K	142 K	107 K
Neu.	893 K	2340 K	888 K	1368 K	2973 K	1337 K
Neg.	934 K	17 K	14 K	817 K	159 K	112 K

(b)

results obtained, appropriateness of the response time, and time/money required to run the project. In this section we provide a quantitative evaluation of these aspects with reference to our case study.

Overall, the number of collected clips in WebPolEU was around ten millions, which decreased to six millions after dropping non-relevant sources and duplicate clips. Noticeably, the quantity of information generated by the semantic enrichment process is much larger ($|\textsf{NLP Entity Occurrence}| \approx 500M$ for each language) and places the project on the edge of big data. The topics were provided by the team of socio-political researchers involved in WebPolEU; the number of topics is about the same (around 500) for Germany, Italy, and UK, since the same issues were discussed in the three nations. Although the number of clips collected for Germany (933 K) is quite lower than that for Italy and UK (about 3 M each), the number of occurrences generated is not so different; this is because the lower number of clips for Germany is counterbalanced by their greater average length.

4.6.1 Effectiveness

Our first goal is to evaluate different semantic enrichment techniques in terms of the trade-off they offer between added value on the one side, and resource demand/effort on the other. In particular, we will compare the approach based on crawler meta-data, crawler sentiment, and information retrieval (called IR in the following) against the approach based on NLP analysis (called NLP). We will focus on the Italian and English clips since they were both enriched using the same tools (Brandwatch for IR and SyN Semantic Center for NLP). As shown in Table 4.2(a), the two techniques find the same topic occurrences in a clip in most cases. This shows that the KPIs based on topic counting, which are widely adopted for UGC analysis, does not necessarily require the adoption of sophisticated ontology-based techniques and a full comprehension of sentence syntax and semantic. Conversely, these techniques are required when analyzing semantic co-occurrences is one of the users' goals.

Table 4.3 IR and NLP sentiment accuracy for each sub-sample

Clipping Quality	Text Complexity	Negative		Neutral		Positive	
		IR	NLP	IR	NLP	IR	NLP
High	Standard	16.7%	62.7%	85.1%	39.9%	21.7%	68.3%
	Hard	15.2%	36.4%	100.0%	44.4%	0.0%	100.0%
	Overall	15.9%	49.5%	92.5%	42.2.%	10.8%	84.2%
Low	Standard	20.0%	55.0%	87.8%	54.9%	28.6%	57.1%
	Hard	0.0%	0.0%	100.0%	0.0%	–	–
	Overall	10.0%	27.5%	93.9%	27.4%	28.6%	57.1%

Text Complexity	Negative		Neutral		Positive		IR	NLP
	IR	NLP	IR	NLP	IR	NLP		
Standard	18.3%	58.8%	86.4%	47.4%	25.1%	62.7%	43.3%	56.3%
Hard	7.6%	18.2%	100.0%	22.2%	0.0%	100.0%	43.0%	36.2%
Overall	13.0%	38.5%	93.2%	34.8%	16.7%	75.2%	43.2%	47.2%

The real power of NLP comes into play when analyzing sentiment. Table 4.2(b) shows that Brandwatch, which adopts a rule-based technique for sentiment analysis, hardly assigns a non-neutral sentiment to a clip: most of the clips that Brandwatch labels as positive/negative are positive/negative for SyN too, while the two systems often disagree on neutral clips.

There is not much point in discussing the differences in IR and NLP sentiment without knowing which is the correct one. For this reason we evaluated the accuracy of the returned sentiment by asking five domain experts to manually tag a sample of the clips. The sample includes about 600 clips from the English corpus, equally divided by media type and NLP sentiment (as computed by Syn). Besides defining the clip sentiment as either negative, neutral, or positive, the domain experts were also asked to rate, for each clip, its *clipping quality* (i.e., the amount of non-relevant text present in the clip), which could impact on the difficulty of assigning the right sentiment, and its intrinsic *text complexity* (i.e., the effort of a human expert in assigning the sentiment due to irony, incorrect syntax, abbreviations, etc.). Table 4.3 shows the IR and NLP sentiment accuracy (i.e., percentage agreement with the consensus sentiment) for each sub-sample; a correct interpretation of the results requires some further explanation due to the different cardinalities of the sub-samples. It is apparent that the experts rated most of the clips as neutral —thus, a dummy classifier always stating *neutral* would most probably be very successful! Before commenting the tables, we recall that the lower bound on accuracy is 33%, which is the percentage of success of a random classifier.

- The high accuracy achieved by IR on neutral clips is not actually due to its real capability of discerning between negative, neutral and positive clips, but rather to its inability/caution in assigning a non-neutral sentiment. Indeed, its accuracy on negative and positive clips is below that of a dummy classifier.
- When using NLP, detecting positive sentiments turns out to be much easier than identifying negative ones. This happens because positive opinions are normally characterized by enthusiastic words, while negative ones are often blurred by irony, which can hardly be detected. This is confirmed by the experts, that mostly label positive clips as having standard complexity.
- For clips whose texts complexity has been classified as hard, both IR and NLP often fail in assigning the right sentiment.
- The clipping quality impacts more on NLP than on IR accuracy. It would be interesting to investigate if this is related to the deeper level of text understanding NLP tries to achieve.

As to analysis, the last phase of the SBI process, we can only give some qualitative assessment. Moving from standard dashboards to user-driven OLAP analysis has been recognized as truly valuable by the WebPolEU users since it enables them to flexibly and autonomously navigate data to get a deeper insight on the ongoing trends, leaning on hierarchies to better analyze data.

4.6.2 Efficiency

We start this section by mentioning how the architecture in Figure 4.1 has been implemented in the WebPolEU project. ETL and analysis run on an 8-cores server with 64 GB of RAM; the text search engine runs on a 7-nodes cluster (each node equipped with a 4-cores processor and 32 GB of RAM); the semantic enrichment component runs on a 6-nodes virtual cluster (each node equipped with a 12-cores processor and 10 GB of RAM). As to the data volume, the raw crawler files take 79 GB, the ODS 481 GB, the DM 116 GB, and the documents for text search 65 GB. Noticeably, since the OLAP cubes in the DM mainly store numerical data, their required storage is lower than that of the ODS.

Table 4.4 shows the time required for running the main ETL flows with reference to all clips (a 20× parallelization was adopted to maximize the throughput) and the time for the bi-directional ETL flow between the ODS and NLP semantic enrichment as a function of the clip length (here times were measured on a single-process basis). These

Table 4.4 Average processing time in seconds for 10 000 clips; to the right, average time for NLP semantic enrichment of one clip

ETL Flow	Time per 10 K Clips
Crawling → ODS	2868
ODS ↔ IR Sem. Enrich.	180
ODS ↔ NLP Sem. Enrich.	23 035
ODS → DM (IR)	13
ODS → DM (NLP)	68
ODS → Document-Base	16

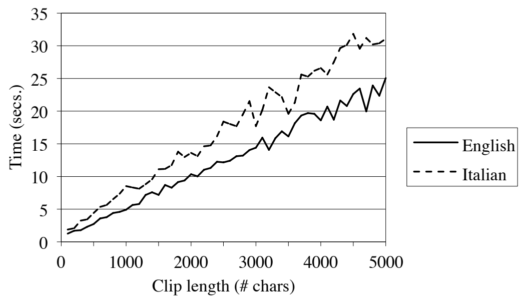


Table 4.5 Execution time for chart, OLAP and free-text queries

Query Type	Exec. Time (sec.)			Query Example
	Min	Avg	Max	
IR Charts	1.2	7.4	25.5	Daily trend of UK topic occurrences for each channel type and party
NLP Charts	0.8	62.2	288.7	Top 5 entities related to the “Cameron” topic
IR OLAP	0.3	7.7	50.1	Average crawler sentiment for each party and country
NLP OLAP	0.4	14.7	79.4	Average sentiment for each topic sector and clip type
Free-text	0.2	1.1	2.9	“Europe” AND “Politics” (filter on Clip.Source = “telegraph.co.uk”)

results confirm that NLP semantic enrichment deeply impacts on the time and space required to feed the DM, so its adoption should be carefully evaluated. Interestingly, both processing time and data size are higher for Italian clips due to the greater complexity of the Italian language.

We close our efficiency analysis by showing, in Table 4.5, the execution time for an analysis workload including 33 queries, which can be classified into three groups corresponding to the main functions of a typical SBI platform: charts, OLAP analysis, and free-text search. The first group includes the queries whose output is used to draw the charts available in the WebPolEU interface (e.g., tag cloud, trends, etc.), while the other two groups were created by auditing and sampling the queries actually issued by WebPolEU users. Although the average query time is higher for NLP queries (because the corresponding cubes have higher cardinalities), all the groups are compatible with interactive analyses.

4.6.3 Sustainability

The first design iteration for WebPolEU took 84 person-days overall; of these, 18 were for designing the domain ontology (including topic definition), 21 for designing and testing semantic enrichment (in particular for tuning the dictionary), and 26 for designing and testing crawling queries. The second iteration was mostly used for tuning the ETL (20 person-days out of 30).

The main critical issues related to each activity are listed below:

- Ontology design: the correctness of the results is deeply affected by the number of topics and aliases defined. For example, with reference to Figure 4.2, the number of occurrences for each topic sector depends on the topics and aliases summarizing that sector, hence, including an unbalanced number of topics for the different sectors may lead to an unfair analysis. Keeping a proper level of detail for different sectors requires a deep knowledge of the domain and related vocabulary.
- Crawling design: commercial solutions (like Brandwatch) normally limit the length of the crawling queries; this makes it harder to properly define the subject area, which is necessary to filter off-topic clips. Finding the proper formulation of queries with constraints on their length and number may become a real nightmare.
- ETL & OLAP design: although parsing a JSON file is a trivial task, handling all the possible unexpected character sequences is more tricky and requires continuous tuning along the whole project. On the other hand, unexpected character sequences often determine a failure of semantic enrichment.

4.7 Conclusions

In this chapter we have analyzed the main factors that impact on the costs and benefits of the main architectural options for SBI [42]. A summary of the pros and cons of the different options, as emerging from our case study, is shown in Table 4.1. Remarkably, it turned out that crawling and semantic enrichment are the components that impact the most on the overall cost-benefit compromise. Here we summarize a few rules of thumb for making a good choice:

- The accuracy of both NLP and IR sentiment can be high on very specific sources and closed domains (such as the CRM of a bank or the movie reviews [76]), but it easily drops as soon as the domain becomes wider. Since a relevant effort is required to properly handle sentiment, the design team should carefully evaluate the use of sentiment analysis techniques by trading-off the accuracy achievable with the related costs.

- Although Twitter provides a partial analysis of the social environment, the shortness of tweets and the high percentage of non-neutral clips make it a good candidate to be the main source for an effective sentiment analysis. Indeed, experimental data show that Twitter clips yield the highest accuracy for NLP sentiment (56.6%, vs. 51.5% of forums and 42.4% of news).

- Dashboards are the standard way for visualizing and analyzing data in SBI projects since they yield an immediate, easy-to-understand, and well-focused representation of results. However, as the role of SBI systems becomes more important in companies, full-OLAP capabilities will increasingly be provided because they clearly enable more flexible and accurate analyses of the UGC.

- Off-the-shelf projects provide *quick-and-dirty* answers but preclude the possibility of carrying out in-depth analysis, tuning, reprocessing, and integration with enterprise data. They should be pursued either at an early stage of adoption of SBI solutions to assess the real value of social data for the company, or if the available resources are very limited.

Chapter 5

SABINE: a modular benchmark for Social BI

In this chapter we conclude the study on social data by presenting SABINE, a modular benchmark for SBI in the domain of European politics. SABINE is meant to fill the gap left by the lack of publicly-available, real-world data for experimenting approaches, which often restrains research in the SBI area. The benchmark includes 6 millions bilingual clips crawled from 50 000 web sources, each associated with metadata and sentiment scores; an ontology with 400 topics, their occurrences in the clips, and their mapping to DBpedia; two multidimensional cubes for analyzing and aggregating sentiment and semantic occurrences. We also propose a set of research challenges that can be addressed using SABINE; remarkably, the presence of an expert-validated ground truth ensures the possibility of testing approaches to the whole SBI process as well as to each single task.

5.1 Introduction

From a scientific point of view, SBI stands at the crossroads of several areas in Computer Science such as Database Systems, Information Retrieval, Data Mining, Natural Language Processing, and Human-Computer Interaction. Figure 5.1 sketches a functional overview of the overall SBI process and highlights it cross-disciplinarity. Though the ongoing research in these single fields has made available a bunch of results and enabling technologies for SBI, an overall view of the related problems and solutions is still missing. Besides, the peculiarities of SBI systems open new research problems in all the previous areas. On the other hand, research developments in SBI are often restrained by the lack of publicly-available, real-world data for experimenting and

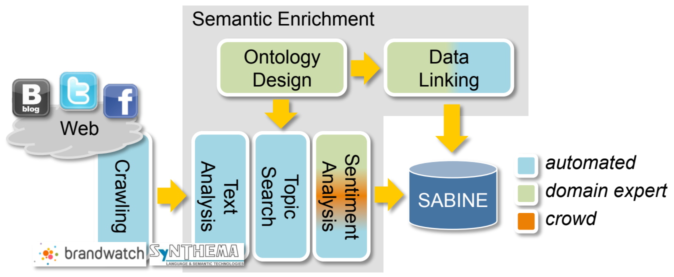


Fig. 5.1 The functional architecture of the SBI process which created SABINE

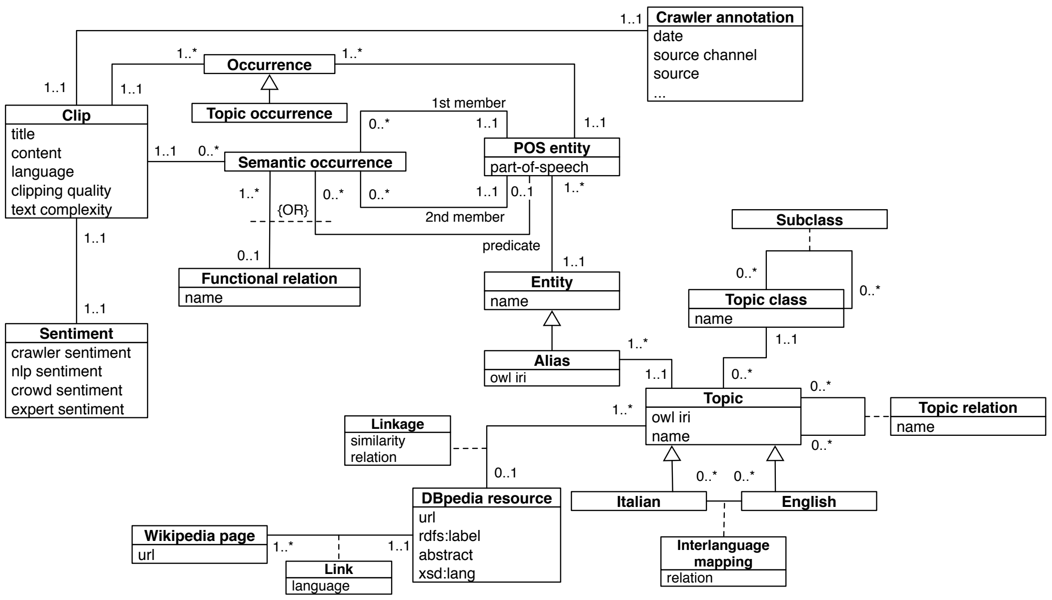


Fig. 5.2 UML model of SABINE

comparing approaches, and by the inherent difficulties in determining a ground truth for assessing the effectiveness of an approach.

To fill this gap, in this chapter we present *SABINE* (SociAl Business INtelligence bEnchmark), a modular benchmark in the domain of European politics with specific reference to the 2014 European elections. SABINE includes: 6 millions bilingual clips crawled from 50 000 web sources, each one associated with metadata and sentiment scores; an ontology with 400 topics, their occurrences in the clips, and their mapping to DBpedia; and two multidimensional cubes for analyzing and aggregating sentiment and semantic occurrences for SBI analytics purposes. Remarkably, the presence of a manually-validated ground truth for each phase of the SBI process ensures the possibility of testing approaches to the whole process as well as to each single task. In this direction, our proposal is completed by a set of research challenges that can be addressed using SABINE; the task selection we propose is large and diverse enough to

Table 5.1 SABINE figures

Figure	*ENG*	*ITA*
# Web Sources	23 K	25 K
# Topics	409	434
# Topic Aliases	709	798
# Entities	2868 K	1242 K
# Clips	3275 K	2394 K
Avg Clip Length (# Chars)	2026	1677
# Entity Occurrences	511 M	218 M
# Topic Occurrences	23 M	14 M
# Semantic Occurrences	48 M	35 M

be sufficiently representative of a wide range of research tasks, ranging from content analysis, to semantic analysis, up to the more comprehensive SBI analytics.

The outline of the chapter is as follows.

- In Section 5.2 we describe the benchmark content.
- In Section 5.3 we discuss the techniques adopted for building SABINE.
- In Section 5.4 we propose a set of SBI-related research tasks for SABINE.
- In Section 5.5 we draw the conclusions.

5.2 The Content of SABINE

SABINE has been built as one of the results of the WebPolEU project, already presented in Section 4.4. The UML model of the SABINE content (except for the multidimensional part, whose content is described by Figure 5.4) is shown in Figure 5.2, while its quantitative features are summarized in Table 5.1 (see [42] for a more detailed profiling of the clips). The main content components of SABINE can be described as follows.

5.2.1 Topics and Mappings

SABINE provides about 400 relevant topics organized in a topic ontology built by domain experts (a team of five socio-political researchers). The topic ontology (modeled by classes Topic, Topic class, Subclass, and Topic relation in Figure 5.2) represents the

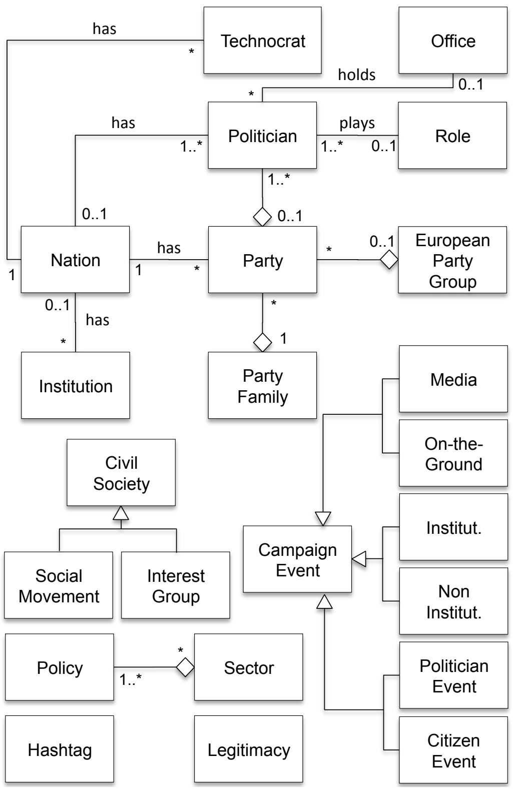


Fig. 5.3 The topic ontology represented as a UML class diagram

set of concepts and relationships that, on the domain experts' judgement, are relevant to the subject area; its role in the SBI process is twofold: to act as a starting point for designing effective crawling queries on the one hand, and to support analyses based on relevant concepts (e.g., how often the public debt policy is mentioned) and on their aggregations (e.g., how often the sector of economics and its policies are discussed) on the other. The class diagram for the topic ontology of the socio-political subject area of SABINE benchmark is shown in Figure 5.3; for instance, topics "public debt" and "austerity" are instances of topic class Policy and are related to topic "economic policy" (Sector). To enable more accurate analyses, a large set of topic aliases (class Alias in Figure 5.2) has been identified and is available for topics (e.g., "tory" is an alias for "conservative").

Inter-Language mappings (class Interlanguage mapping) between corresponding topics in the two benchmark languages have been manually created by the domain experts. In most cases these mappings simply express an exact translation (e.g., "immigration" is mapped onto "immigrazione" with semantics owl:sameAs), whereas they are based on weaker semantic relationships when a concept is differently expressed in the two languages (e.g., "immigration" is mapped onto "migrante irregolare" —which means illegal migrant— with semantics sabine:related). A mapping has been found only in 86% of cases since, according to the experts' judgement, some topics are specific of either UK or Italy (e.g., "Scottish National Party" and "Quirinale"). Furthermore, topics have been linked to their corresponding DBpedia resources (classes DBpedia resource, Wikipedia page, Link, and Linkage). Linkage has been carried out automatically as described in Section 5.3.6 and then validated by domain experts.

5.2.2 Clips and Annotations

The benchmark provides a large corpus (around 6 millions) of raw clips (class Clip) extracted by the Brandwatch crawler from a broad set (almost 50 000) of web sources including social networks, blogs, and web sites. The most frequent clip sources are Facebook (53.8% of the clips) and Twitter (27.5% of the clips). The corpus is bilingual and *comparable*, i.e., it includes text in two languages (English and Italian) regarding similar topics [23, 132]. Each clip is associated with a set of metadata (class Crawler annotation); 40 attributes overall are provided, partly returned by the crawler (e.g., title, date, source MozRank, author information, and geo-localization) and partly manually annotated by the domain experts (e.g., source type).

Clips are enriched with other relevant information resulting from clip text analysis. In particular, each clip is associated with all its occurring entities and their parts-of-speech or POSs (classes POS entity, Entity, and Occurrence). An *entity* is a concept that emerged from text analysis but is not necessarily a topic; parts-of-speech (POSs) are the roles taken by entities within a clip sentence (e.g., *noun*, *verb*, *preposition*). Among the set of entity occurrences, a relevant role is taken by the occurrences of topics and their aliases (class Topic occurrence). Finally, text analysis also led to the detection of more complex linguistic patterns involving multiple entities in the same sentence (classes Semantic occurrence and Functional relation). In particular, each semantic occurrence relates two entities (first and second member, respectively) by means of either a functional relation (e.g., *agent* or *qualifier*) or a predicate corresponding to an entity.

All clips are also annotated with two sentiment values (class Sentiment). The first one (*crawler sentiment*) is categorical (i.e., negative, neutral, positive); it has been determined for each clip by the Brandwatch crawler through rule-based techniques. The second one (*NLP sentiment*) is numerical; it has been determined by the SyN semantic engine for each clip sentence, then averaged for each clip (see Section 5.3.5). Finally, a subset of 2400 clips have also been labeled with a *crowd sentiment*, and half of these 2400 clips have further been labeled by domain experts (*expert sentiment*). This subset of manually-labelled clips has been created using a stratified sampling strategy based on the type of clip source (e.g., social network and blog) and on the clip sentiment.

Example 12 *Here is an example of a SABINE clip: "Another compassionate conservative. Making fun of a parkinson's victim. Michael J Fox has more courage than you will ever hope to have". Some metadata for this clip are* source*="facebook.com",* channel_type*="facebook",* source_type*="Social network / Social media",* country*="US", and* fb_role*="audience". The only occurring topic is "conservative"; among the occurring entities we mention "compassionate", "victim", and "courage" (with POSs* adjective*,* noun*, and* noun *respectively). Text analysis led to find different semantic occurrences of entities with their POSs, for instance the one between "compassionate" and "conservative" (with POS* adjective *and* noun*, respectively, and functional relation* qualifier*) and the one between "have" and "courage" (with POS* verb *and* noun*, respectively, and functional relation* object*). The expert sentiment for this clip is* -1 *(i.e., negative), while both the crawler and the NLP sentiment are positive (because neither of the approaches was able to detect the irony in the sentence "Another compassionate conservative"). Another example of clip is "US President Barack Obama criticized Russia in a telephone call [...]", which shows a semantic occurrence between entities "Barack Obama" (POS* proper noun*) and "Russia" (POS* proper noun*) involving entity "criticize" as a predicate.*

5.2.3 Multidimensional Cubes

These are ROLAP cubes providing an easy-to-query representation of the clip content and of the outcome of the clip enrichment process. The first cube, Sentiment, is centered on clips, and represents the set of topics appearing in each clip as well as the sentiment values computed for that clip. The second cube, Semantic Occurrence, is centered on the semantic occurrence of POS entities within clips and explicitly models couples of entities in the same sentence together with an optional predicate. The conceptual schemas of these cubes are depicted in Figure 5.4 using the DFM notation [63], where cube measures are listed inside the box, dimensions are circles directly attached to the box, and hierarchies are shown as DAGs of dimension levels. In particular, the hierarchy built on dimension Clip includes the crawler annotations, while the one built on Topic (called *topic hierarchy* from now on) derives from the topic ontology of Figure 5.3 and enables topic-based aggregations of clips in the OLAP front-end. For instance, a roll-up from Politician to Party and Party Family on the Sentiment cube allows to obtain the opinions about a wing as an average of the opinions about all the politicians belonging to the parties of that wing.

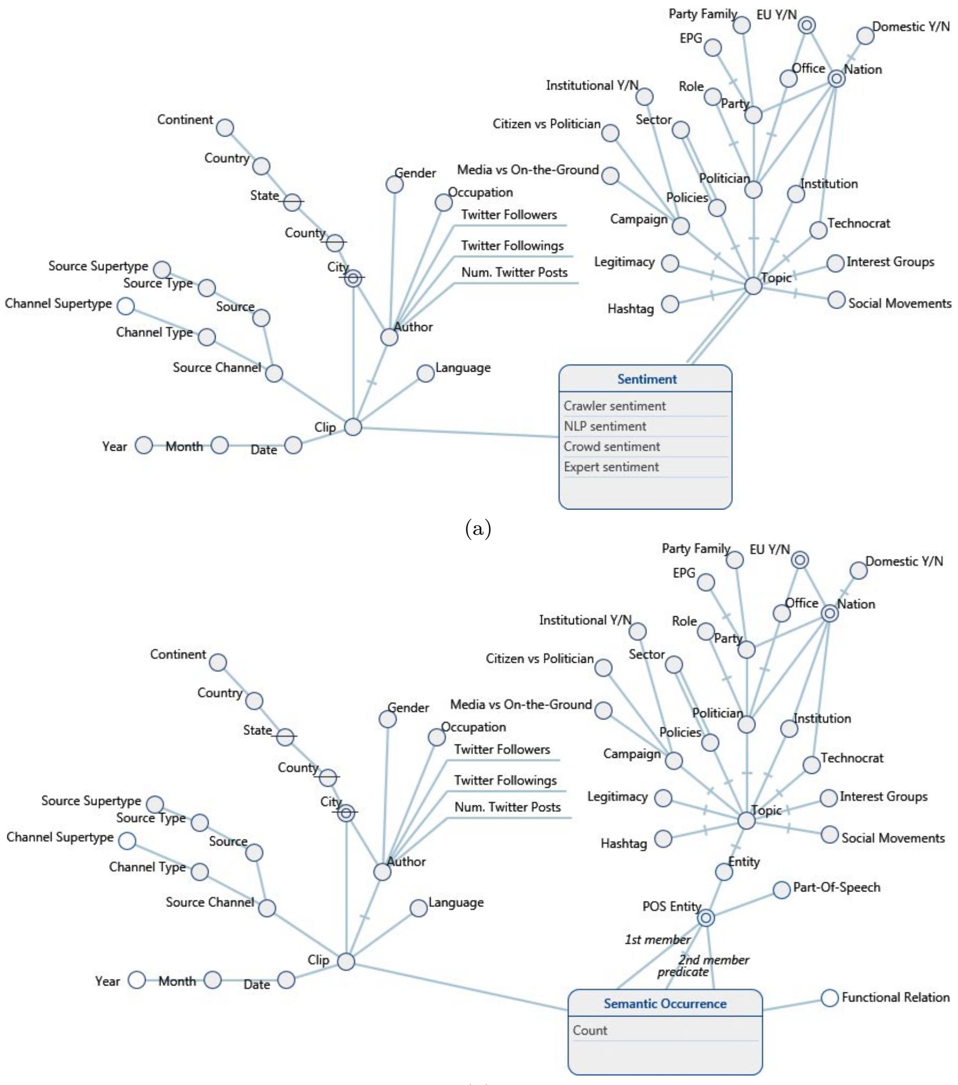


Fig. 5.4 A DFM representation of the Sentiment (a) and Semantic Occurrence (b) cubes (for drawing simplicity, some levels of the topic hierarchy are hidden)

5.3 SABINE Construction Techniques

To develop SABINE we followed the ad-hoc methodology described in [43], which has been conceived to support and speed up the initial design of an SBI process on the one hand and to maximize the effectiveness of the experts' analyses by continuously optimizing and refining all the process tasks on the other hand. Quick tuning iterations are probably the most distinctive feature of this type of projects, and are necessary to cope with the high fickleness of the topics covered in social conversations. In the following subsections we give further details on the techniques used for each task of the SBI process, using Figure 5.1 as a reference.

5.3.1 Ontology Design

Designing the topic ontology of the European politics subject area was mainly a methodological issue. Consistently with the methodology we followed [43], we initially carried out a *macro-analysis* to identify the themes relevant to the subject area (e.g., “culture”) and a first set of topics (e.g., “school”). Then, the *ontology design* phase was specifically dedicated to collecting, for each theme, a comprehensive set of topics and to arrange them within an ontology (using Protégé) by expressing inter-topic relationships. Along the whole project lifetime, the topic ontology was weekly tuned and refined (in collaboration with domain experts) to accommodate new topics, topic classes, and relationships; the final result is shown in Figure 5.3.

As already mentioned, the topic hierarchy depicted in Figure 5.4 was derived from the topic ontology by applying standard techniques for multidimensional modeling of operational data sources, so as to enable topic-based aggregation of clips in an OLAP fashion.

5.3.2 Crawling

For keyword-based crawling we adopted the Brandwatch service (www.brandwatch.com), a commercial solution to ensure a satisfactory coverage of web sources during the project duration (three months). Brandwatch adopts a template-based engine, that is, it extracts only the informative UGC by detecting and discarding advertisements and banners (a process called *clipping*); it also drops duplicate clips using content aggregators.

The two main design activities related to crawling are *source selection* and *crawling query design*. As to source selection, Brandwatch already comes with its source base; however, our domain experts provided a set of about 100 additional domain-specific web sites (e.g., www.davidcameron.com) to be added to the source base and crawled. Similarly, a large number of non-relevant sources (e.g., www.tripadvisor.co.uk) had to be progressively eliminated from the source base because they were generating too many off-topic clips. Overall, in WebPolEU, dropping non-relevant sources and duplicate clips led to cut the number of collected clips from ten to six millions.

Keyword-based queries in Brandwatch rely on a set of 23 Boolean operators that allows to express filters on both textual features (e.g., the distance between two words) and metadata (e.g., the author's country and the web site name). We initially created a set of queries based on the topics in the domain ontology and on a few additional ones we discovered during source selection. To reduce the number of off-topic clips, a *content relevance analysis* was weekly performed by first asking the domain experts to manually label off-topic clips within a sample, then tuning the crawling queries to exclude those clips so as to increase the percentage of in-topic clips.

Example 13 *As an example of how a query was transformed to exclude off-topic clips, we show the one aimed at retrieving the clips mentioning George Osborne (a conservative politician), which had to be changed from*

(raw:"Osborne") OR "george osborne"

to

(raw:"Osborne" NOT (ozzie OR ozzy)) OR "george osborne"

5.3.3 Text Analysis

For text analysis we used the *SyN-Semantic Center* (www.synthema.it) commercial engine. SyN was used for splitting clips in sentences and for extracting the single entities, their part-of-speech, and their semantic occurrences. The linguistic and semantic text analysis made by SyN is based on morpho-syntactic, semantic, semantic role, and statistical criteria. At the heart of the lexical system lies McCord's theory of slot grammars [114]. The system analyzes each sentence, cycling through all its possible constructions and trying to assign the context-appropriate meaning to each word by establishing its context. Each slot structure can be partially or fully instantiated and it can be filled with representations from one or more statements to incrementally build the meaning of a statement. The core of the system is the SyN ontology, developed through twenty years of experiences and projects.

5.3.4 Topic Search

With this term we refer to the task of indexing all the occurrences of a topic (or one of its aliases) within a clip. We relied on two different techniques for searching topic occurrences in SABINE. The first one is a simple text matching technique that retrieves the exact occurrences of topics and aliases, implemented in-house as a Java algorithm. The second one is based on the results of the text analysis made by SyN, which extracts all the occurring entities in a clip; in this case, topic occurrences are obtained by linking topics and aliases to the corresponding entities.

Clearly, both techniques have pros and cons. By avoiding all kinds of text analysis, the first technique typically trades a better performance in terms of speed with a lower accuracy of the results. In particular, the results tend to include the occurrences where a topic (or an alias) is used in the clip with different semantics from the one originally meant in the topic ontology. This problem arises when topics (or aliases) in the ontology are too generic.

Example 14 *"Osborne" is an alias of topic "George Osborne". By identifying the exact matches of the word "Osborne", the first technique wrongly associates the occurrences of other people with the same surname (e.g., Peter Osborne, father of George) to the topic "George Osborne". Although this problem is similar to the one seen in Example 13 for crawling query design, it poses a different challenge (essentially because Peter Osborne frequently appears in in-topic clips).*

The second technique presents the opposite challenge: by carrying out an in-depth comprehension of the clip semantics, the entities produced by SyN tend to be very specific, possibly leading to the pulverization of the same concept into a wide set of entities. Therefore, this problem arises when topics (or aliases) in the ontology are intentionally generic.

Example 15 *"university" is a topic of class* Policy*. The text analysis made by SyN produces a different entity for each specific university found in the clips (e.g., "Oxford University", "University of Cambridge", etc.). As a consequence, all these entities must then be manually associated with topic "university".*

The adoption of both techniques enabled us to double-check the results and to track down the causes of conflicting results. Eventually, mismatches were manually solved in most cases, yielding a 91% agreement between the two techniques (over 14 millions occurrences).

5.3.5 Sentiment Analysis

Sentiment analysis is probably the hardest task in SBI; for this reason we included in SABINE both system-based and human-based sentiment scores. While system-based scores can be used as a baseline for testing other automatic techniques, human-based scores represent the ground truth.

- **Crawler sentiment**. This score, computed by Brandwatch, tags each clip of SABINE. The sentiment analysis component of Brandwatch is based on mining rules specifically developed for each language supported.
- **NLP sentiment**. SyN includes its own sentiment analysis component [126] whose score takes into account the negative or positive polarization of words and concepts, as well as the syntactical tree of the sentence being analyzed. SyN is sophisticated enough to modify the polarization of words based on the related adverbs, adjectives, conjunctions, or verbs, by taking in account specific functional-logic complements; it even tries to identify idiomatic or colloquial expressions and give an interpretation to negations. Each clip of SABINE is tagged with this score as well.
- **Expert sentiment**. This score was defined for a sample of 1200 clips (600 English + 600 Italian) by asking our domain experts to manually tag them. The clips are equally divided by media type and NLP sentiment (as computed by SyN). Besides defining the clip sentiment as either negative, neutral, or positive, the domain experts were also asked to rate, for each clip, its *clipping quality* (i.e., the amount of non-relevant text present in the clip due to an inadequate template used by the crawler when clipping), which could impact on the difficulty of assigning the right sentiment, and its intrinsic *text complexity* (i.e., the effort of a human expert in assigning the sentiment due to irony, incorrect syntax, abbreviations, etc.).
- **Crowd sentiment**. This score was given to a sample of about 2400 clips (1200 + 1200, including the clips tagged by experts) through a crowdsourcing process. To this end, we selected a crowd of around 900 workers within a class of bachelor-degree students in the field of humanities and political science at the University of Milano (average worker age is 21). Crowdsourcing activities were performed during one month and each worker tagged 46 clips on average. As a support we employed our Argo system (island.ricerca.di.unimi.it/projects/argo/, in Italian), which provides crowdsourcing functionalities based on *multi-worker task assignment* and *consensus evaluation* techniques [18]. In Argo, each clip to evaluate was represented as a *choice task*, meaning that each worker receiving a task to execute was asked to read a clip and select her preferred answer among

three available options, namely positive, negative, and neutral sentiment. Each clip was assigned to a group of 6 different workers. A group member autonomously tagged each clip received and independently produced the answer according to her personal feeling and judgement. Each worker also had the opportunity to refuse a clip in case she recognized to have insufficient expertise for its sentiment evaluation. Given a clip, its sentiment score was defined as an answer agreement (i.e., consensus) among the members of the group that tagged that clip. Two workers agree on a clip when they assigned the same score to that clip. In Argo, consensus evaluation was enforced through a weighted-voting mechanism called *supermajority*, in which the answer of a worker has a weight corresponding to her reliability.[1] Supermajority was used to verify that the sentiment score having the highest degree of consensus within a group was supported by a qualified majority larger than 50% [18]. In this case, such score was assigned the corresponding clip. Conversely, when a qualified majority of workers was not found within the group, the task was uncommitted and scheduled for re-execution by a different group of workers with higher reliability.

5.3.6 Data Linking

The goal of data linking is to link ontology topics to the Linked Data Cloud (linkeddata.org). In SABINE, this has been done by coupling automated techniques with manual validation and revision by domain experts; the single steps of the process are described in the following.

1. Topic aliases were used to retrieve a set of candidate DBpedia resources for each topic t through the DBpedia Lookup Service (wiki.dbpedia.org/projects/dbpedia-lookup).

2. The degree of similarity between t and the retrieved candidates (if any) was evaluated through the HMatch matching algorithm [19]. HMatch takes into account both the linguistic information available for t (i.e., its aliases) and the ontological information (i.e., the topic class of t).

3. Topic t was linked to the DBpedia resource r, among the candidates, yielding the highest degree of similarity. The link between t and r is formally defined as a 4-tuple of the form $\mathcal{L}_{t,r} = \langle t, r, \sigma_{t,r}, \rho_{t,r} \rangle$, where $\sigma_{t,r}$ is a real number in the range $[0, 1]$ representing the degree of similarity between t and r, and $\rho_{t,r}$ represents the semantics of the relation holding between t and r.

[1]The worker's reliability depends on the answers she provides to the assigned tasks; specifically, it is increased when the answer contributes to reach the consensus and it is decreased otherwise [18].

4. Each resulting link $\mathcal{L}_{t,r}$ was submitted to domain experts to specify the most suitable semantics $\rho_{t,r}$, choosing among (i) owl:sameAs (t and r have exactly the same meaning); (ii) sabine:narrower / sabine:broader (the meaning of t is more specific/generic than the one of r); (iii) sabine:related (there is a positive association between the meanings of t and r[2]. If none of the previous options was deemed suitable by the domain experts, the link was marked as *incorrect*.

5. The links with semantics different from owl:sameAs and all the incorrect links were submitted to a second validation round, where domain experts manually found additional DBpedia resources to be associated with the corresponding topic with owl:sameAs semantics.

Example 16 *As an example we propose the following link between topic "school" and DBpedia resource* dbpedia:State_school*:*

$$\langle \textit{"school"}, \textsf{dbpedia:State\_school}, 0.75, \textsf{owl:sameAs}\rangle$$

In the first round of validation, experts confirmed that "school" can be linked to dbpedia:State_school*, but with semantics* sabine:broader *(since* school *is broader than* dbpedia:State_school*). The resulting link was*

$$\langle \textit{"school"}, \textsf{dbpedia:State\_school}, 0.75, \textsf{sabine:broader}\rangle$$

Since the semantics is different from owl:sameAs*, the link was submitted to the second validation round, where we asked experts to manually find a DBpedia resource which actually has an* owl:sameAs *relation with "school". Experts found the DBpedia resource* dbpedia:School*, which leads to the addition of a second link for topic "school". The links resulting from the two validation rounds are then*

$$\langle \textit{"school"}, \textsf{dbpedia:State\_school}, 0.75, \textsf{sabine:broader}\rangle$$

$$\langle \textit{"school"}, \textsf{dbpedia:School}, 1.0, \textsf{owl:sameAs}\rangle$$

In Table 5.2, we show some statistics about validation and refinement of English topics. Most of the automatically retrieved links have been considered correct with owl:sameAs semantics (67%); 17% of the remaining links have been evaluated as correct

[2]Note that we modeled relations similarly on SKOS (www.w3.org/TR/skos-reference/) concept relations. The reason why we redefined relation semantics rather than using SKOS is that topics and DBpedia resources are not defined as SKOS concepts.

Table 5.2 Results of the domain expert validation and revision for the English topics

Relation semantics	*# links*	*Avg. similarity*
owl:sameAs	252	0.812
sabine:narrower	7	0.721
sabine:broader	39	0.756
sabine:related	17	0.781
incorrect	62	0.22
sub-total	377	
expert-provided resource	135	0.433
total	512	

but with semantics different from owl:sameAs. In particular, the automatic linking procedure tends to provide specific (rather than generic) DBpedia resources for topics. The automatic approach was incorrect in 16% of cases. We note also a positive correlation between the average degree of similarity associated with links and the positive evaluation provided by experts. This is important for associating a reliable degree of similarity to the links in the final benchmark. Finally, for 135 topics, domain experts provided a DBpedia resource as an owl:sameAs counterpart for the topic.

5.4 Research Tasks

In this section we describe the main research tasks that can be supported and evaluated using SABINE. Remarkably, to enable partial, ad-hoc downloads for each task, we subdivided SABINE into separate components shown as packages in Figure 5.5. A package models a component of the benchmark that can be downloaded separately; one package depends on another when an object of the former references an object of the latter. For each task, in Table 5.3 we summarize the SABINE component(s) to be taken in input, the expected task output, the ground truth we provide, and possible metrics for evaluating the task results with reference to the ground truth.

5.4.1 Content Analysis Tasks

Sentiment analysis. At the state of the art, the available techniques for sentiment analysis provide a satisfactory level of accuracy in narrow domains with limited dictionary and topics of discussion (e.g., movie reviews [107]). Finding an appropriate sentiment for a clip is still an open problem in wide domains and when sarcasm and

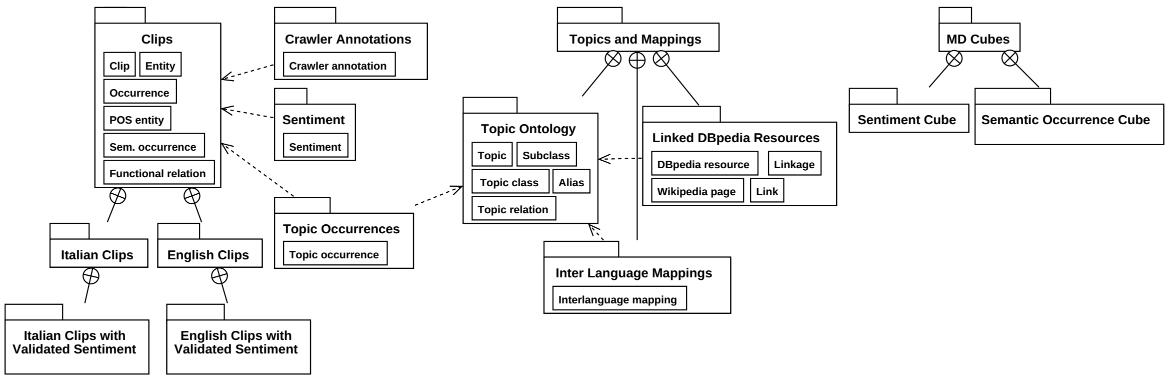


Fig. 5.5 Components of SABINE and their composition and dependency relationships

Table 5.3 Task overview

Task	*Input*	*Output*	*Ground truth*	*Evaluation*
Content Analysis Tasks				
Sentiment analysis	Clips	Clip sentiment	Sentiment	Precision; Recall
Topic search	Clips; Topic Ontology	Topic occurrences in clips	Topic Occurrences	Precision; Recall; Interpolated precision
Document classification	Clips	Classes of clips	Crawler Annotations	Purity; Mutual information; Rank index
Semantic Analysis Tasks				
Cross-language analysis	Topic ontology; Clips	Matches between English and Italian topics	Inter-Language Mappings	Precision; Recall
Topic discovery	Clips	Topics	Topic Ontology	Precision; Recall
Data linking	Topic Ontology	Links between topics and DBpedia resources	Linked DBPedia Resources	Precision; Recall; Interpolated precision
SBI Analytics Tasks				
Multidimensional modeling	Topic Occurrences; Sentiment; Clips; Crawler Annotations; Topic Ontology	Multidimensional cubes	MD cubes	Sum squared error
SBI analytics	Clips	Query answers	MD Cubes	Sum squared error

irony are used —which is often the case for the clips in SABINE. The challenge we propose here is to assign a sentiment score (either *positive*, *negative*, or *neutral*) to each clip. As described in Section 5.3.5, SABINE provides four sentiment scores with different levels of certification. Noticeably, the 1200 clips tagged by domain experts have also been annotated with the level of difficulty (either *normal* or *hard*) the experts encountered in assigning the sentiment and with the presence/absence of undesired and irrelevant text due to crawler errors.

Topic search. Entity search in raw text is a classic information retrieval task. The approaches in this area typically uses lemmatization and stemming to increase their recall. Entity search turns to semantic search when additional information such as synonyms [17], ontologies [40], and context [137] are exploited. The research challenge we propose here is to find the occurrences of the given topics within the corpus of clips. The benchmark can be used to check the efficiency and effectiveness of original techniques. As to effectiveness, the ground truth provided by SABINE consists of the occurrences of the topics in the clips as obtained through two different and independent techniques (see Section 5.3.4). Both traditional and semantic search techniques can be applied since (i) topic aliases provide a set of synonyms for topics, and (ii) the topic ontology provides the complete list of topics and their relationships (e.g., a Sector includes several Policies; a Politician belongs to a Party).

Document classification. Document classification aims at associating a document with one or more document classes. There is a wide literature on this subject and a variety of different approaches have been proposed, including for example probabilistic techniques such as Naive Bayes models [102], techniques based on the vector space model of documents and on support vector machine [103], and techniques based on matrix decomposition [112]. Document classification may lead to different results according to different perspectives, especially when classification is driven by subjective criteria — as in the case of sentiment analysis. Consequently, for document classification we propose to rely on the crawler annotations, i.e., metadata associated to the documents according to objective criteria (see Section 5.2). In particular, annotations include the dimensions of language, author, source, and date. For author, source, and date, metadata also provide further levels of aggregation. Each of these dimensions can be used to group clips in different classes with the desired level of aggregation, from relatively small and specific classes to large and generic ones. Examples of clip classes along the date dimensions are those grouping clips by date of publication (specific) and those grouping clips by year (generic). According to the classification scheme derived from the crawler annotations, it is then possible to evaluate a classification approach either with a supervised strategy, where classes are known a-priori, or with an unsupervised strategy, starting from the clips only. The goal in both cases is to rebuild the same classification that can be derived from crawler annotations.

5.4.2 Semantic Analysis Tasks

Cross-language analysis. This area has several relevant applications. For example, *cross-language information retrieval* addresses the problem of finding information in one language in response to queries expressed in another, while *cross-language text categorization* uses labeled documents in one language to classify documents in another language. A requirement shared by these applications is the availability of a bilingual dictionary. Traditional approaches assume that such dictionaries are either given a priori (and typically obtained through a time-consuming manual effort) or automatically acquired from *parallel corpora* [45]. Since parallel corpora are still a scarce resource in several languages and contexts, recent researches employ *comparable corpora* [23, 132] and unsupervised object matching methods [72, 140]. Both methods are promising in terms of language portability because they do not require external language resource. The research challenge we propose in SABINE is to automatically derive the best match between inter-language topics. The input is the sets of topics in the two languages as well as the set of clips representing comparable corpora. The benchmark includes the correct matches, which have been manually defined by the domain experts.

Topic discovery. Several contributions have been provided in the area of topic discovery, mainly in the research field of topic modeling starting from Latent Semantic Analysis (LSA) approaches [100] back in the 90's to Latent Dirichlet Allocation (LDA) approaches [14] in the 2000's. Besides latent analysis, other important contributions are based on clustering techniques [108, 183]. More recently, combinations of LDA, mainly based on Gibbs sampling, and clustering have been proposed to handle topic modeling with very large text collections [121]. Topic modeling is an important reference in the social sciences domain and it has been applied to several research issues, including the impact and dissemination of research [35], social media [15], and the study of changes in technology & innovation [182]. Judging about the presence of a topic in a document is a highly subjective and domain-dependent task. Thus, evaluating automatic solutions requires a gold standard of high quality, where topics are validated by domain experts. We propose to exploit our topic ontology and topic occurrences in clips, which were both validated by domain experts, in order to evaluate topic discovery strategies and tools. In particular, we envisage both supervised strategies, where the number and categories of topics are given a-priori, and non-supervised strategies, where topics are discovered from the clips only and then the topics discovered are compared with those validated by our experts. The evaluation may be based on the set of topics that are discovered in the corpus, without considering which topic occurs in which clip, or be extended to assess also the topic occurrences in clips.

Data linking. In general terms, data linking is the task of determining whether two object descriptions can be linked one to the other to represent the fact that they refer to the same real-world object in a given domain or the fact that some kind of relation holds between them [41]. Usually, this task is performed based on the evaluation of the degree of similarity among different data instances describing real-world objects across heterogeneous data sources, under the assumption that the higher the similarity between two data descriptions, the higher the probability that the two descriptions actually refer to the same object. In our challenge, we propose to link a subset of the topics with their corresponding DBpedia resources. The ground truth we provide has been manually validated by domain experts for both English and Italian. In the field of data linking, a clear and widely accepted definition of the meaning of links between data is still missing. In general, the results of data linking is a set of links associated with same-as semantics and these links are represented using (or abusing) the owl:sameAs OWL relation. However, in the literature there is a wide spectrum of different techniques, capable of discovering different kinds of possible links between object descriptions, ranging from weak correspondences to strong relations of identity. Aimed at supporting validation also for approaches capable of discovering the specific semantics of data links, we provide not only same-as links between topics and DBpedia resources, but also a set of links with other semantics, including analogy, generic similarity, specialization, and generalization (see Section 5.3.6). The evaluation of automatic techniques for data linking can be performed either by verifying if the correct DBpedia resource has been associated with a topic, disregarding the semantics of the association, or by taking into account also the semantics of the link discovered and checking if it is compatible with the links provided by domain experts.

5.4.3 SBI Analytics Tasks

Multidimensional modeling. The goal here is to design a set of multidimensional cubes that summarize and store the knowledge generated by semantic enrichment, in terms of both efficiency and effectiveness. Possible design solutions range from traditional star schemata to more sophisticated approaches like *meta-stars* [48] or those proposed in [29]. The cubes included in the MD Cubes component can than be used as a baseline for comparison.

SBI Analytics. The research tasks proposed so far can be considered technical tasks aimed at enriching raw clips and enable SBI analyses. To complete the benchmark we also propose a functional task, namely, finding the answers to a set of enquiries proposed by our domain experts:

- $Q1$: *Which is the most discussed sector in relationship with each political party?*
- $Q2$: *Which are the most discussed topics for each source type?*
- $Q3$: *Are there any topics whose volume of discussion significantly changes from UK to Italy?* Note that this is a cross-language enquiry and requires to find out those topics whose number of occurrences are most unbalanced between Italian and English clips.
- $Q4$: *How does the sentiment about each politician and technocrat change along time?*

The ground truth we provide for this task are the answers obtained by directly querying our multidimensional cubes. The domain experts verified that these answers are fully compatible with the real social and political phenomena they directly observed during the election period, thus certifying the validity of the ground truth. Remarkably, this task enables an end-to-end assessment of a whole SBI process starting from clips, possibly enriched with different combinations of the benchmark components.

5.5 Conclusions

In this chapter we have presented SABINE, a benchmark of semantically annotated social content in the domain of European politics. SABINE aims to constitute a publicly-available, real-world data benchmark for experimenting and comparing the most commonly performed SBI tasks, crossing the various involved disciplines ranging from Database Systems, Information Retrieval, Data Mining, up to Natural Language Processing and Human-Computer Interaction. SABINE has been designed and properly packaged for modular download to enable the evaluation of a wide variety of research tasks, either separately or in combination, ranging from those more focused on content analysis, to those related to semantic analysis up to more comprehensive SBI analytics, by ensuring that interesting and technical challenges of the tasks will emerge from benchmark implementation.

A main technical advance of SABINE is the availability of multiple, complementary, and validated enrichments of the social content (i.e., textual clips) in form of metadata, annotations, sentiment scores, and DBpedia mappings. The availability of a user-validated ground truth, either by domain experts or by crowdsourcing or both, for each enrichment phase represents a further technical advance of SABINE, to target the purpose of providing a comprehensive and effective benchmark environment.

To download SABINE: SABINE is available for download at big.csr.unibo.it/?q= sabine; packages are made available as compressed archive files containing JSON files (the Clips package), OWL files (the Topics and Mappings package and all its sub-packages), and CSV files (all other packages).

Chapter 6

iMOLD: a collaborative approach for Exploratory BI on linked data

In this chapter, we outline an approach that enables data scientists to extend and complete the hierarchies in their corporate multidimensional cubes through a user-guided process that explores selected linked data and derives hierarchies from them. This is done in an interactive way, by first letting the user navigate linked data in the light of her view of the business, then by detecting in the linked data the recurring modeling patterns that express roll-up relationships between RDF concepts, and finally by translating these patterns into multidimensional knowledge to extend the corporate cubes. Specifically, five different aggregation patterns are distinguished, and the algorithms for detecting them are described. Finally, a case study based on DBpedia is proposed and the results of an evaluation test made with real users are discussed.

6.1 Introduction

The possibility to access and integrate external data in the corporate knowledge is an effective means to enrich the decision-making process. Semantic web technologies (and ontologies in particular) offer a strong contribution in this direction by explaining the semantics of the data [2]. A relevant role in this context is played by linked data, whose shared, structured, and interlinked nature should make them easily accessible and searchable. Unfortunately, as we anticipated in Chapter 1, linked data are often chaotic and badly organized, especially from the schema point of view: for instance, about 96% of the rdf:Properties in DBpedia are not typed. This often prevents data scientists from taking full advantage of the informative wealth lying with linked data.

The goal of the approach we propose in this chapter, named *iMOLD* (Interactive Multidimensional Modeling of Linked Data), is to enable data scientists to extend and complete the corporate multidimensional cubes by retrieving aggregation hierarchies from linked data. This is done in an interactive way, by first letting the user explore linked data in the light of her view of the business, and then finding aggregation patterns to shape new hierarchies for cubes and feed them with data. To give this process collaborative capacity, the knowledge it creates is accumulated within a shared internal ontology, so that each user can reuse and adapt the work already done by others in the company.

The original contributions of this work can be summarized as follows:

1. We introduce a framework for Exploratory BI based on a user-guided process that explores linked data, builds aggregation hierarchies out of them, populates these hierarchies with data, and integrates them into the corporate cubes.
2. We identify five aggregation patterns found in ontologies.
3. We provide algorithms for detecting these patterns in RDF linked data and translating them into hierarchies.

Remarkably, while several works in the literature address the problem of building multidimensional schemata starting from source data, most of them are meant to be used at design-time in the context of so-called *supply-driven design* and consider well-structured data sources (e.g., Entity/Relationship diagrams and relational schemata) where hierarchies can be easily detected by following functional dependencies (represented, respectively, by many-to-one relationships and by foreign keys). Conversely, (i) our approach operates at exploitation time to integrate the corporate cubes with situational data; (ii) it can be regarded as a *mixed* approach to design, because it combines (pattern-based) access to data —typical of supply-driven design— with user interaction —typical of demand-driven design; (iii) the modeling heterogeneity of linked data and the impossibility of describing the multiplicity of properties in the RDFS vocabulary make hierarchy detection more complex.

The chapter outline is as follows.

- In Section 6.2 we summarize the related work.
- In Section 6.3 we provide the necessary background on multidimensional modeling and linked data modeling.
- In Section 6.4 we give an overview of the iMOLD approach.

- In Section 6.5 we present the different aggregation patterns.
- In Section 6.6 we describe in detail the core techniques we use to find these patterns in linked data and translate them into hierarchies.
- In Section 6.7 we propose a case study for iMOLD and discusses the results of an evaluation test involving real users.
- In Section 6.8 we draw the conclusions.

6.2 Related Literature

There have been several efforts towards automating and semi-automating the discovery of multidimensional schemata from available data. Such approach is known as supply-driven design and consists of exploring the data sources in order to identify potential aggregation patterns that would allow to arrange data in a multidimensional fashion. [149] provides a comprehensive discussion on different techniques used to identify dimension hierarchies from available data. In all cases, discovering functional dependencies (FDs) is the cornerstone to automatically build hierarchies. Typically, most approaches look for FDs at the schema level, since instance-based approaches have been shown to be computationally too expensive for real scenarios [84].

Besides traditional approaches assuming the existence of a conceptual representation of the domain (e.g., a E/R or UML class diagram) or a well-formed (at least in Boyce-Codd normal form) logical relational database schema, some efforts have focused on less structured data models such as XML or logics-based formalisms. As mentioned, all of them focus on FD discovery at the schema level. The most relevant works related to ours are [171, 150]. [171] identifies FDs from XML schemata represented as a graph. The graphical representation of the XML schema facilitates finding the FDs, which is examined in the direction expressed by the arcs and according to cardinalities included in the dependency graph. Cardinalities are either provided or inferred from key attributes. Where no cardinality information can be inferred they shift to an instance-based approach by querying schema-compliant XML documents. [150] redefines the concept of FD for logics-based formalisms under open-world assumption. The paper presents new inference algorithms to identify FDs and, based on them, aggregation hierarchies. However, such approach works at the schema level (i.e., no instances are queried).

Now that a huge amount of data is available in the linked data cloud, providing techniques for its effective exploration is becoming more and more important [69, 74].

In this area researches have focused on providing intuitive and effective techniques for visualizing [73] and navigating [117] linked data, provide a high level and conceptual view of large linked data clouds [20] and on integrating and learning information from them [165]. In this line, several efforts focused on deploying multidimensional schemata on linked data to facilitate its exploration and visualization, according to the cube metaphor, have recently emerged (e.g., [124, 93, 90]). However, only [123] presents an approach to automate the discovery of dimension hierarchies on linked data. Similarly to our approach, the authors aim at automatically discovering multidimensional conceptual patterns (i.e., resembling a multidimensional star schema) that summarize linked data based on probabilistic graphical models. They propose the use of the statistics about the instance data to generate the multidimensional schemata and therefore to identify hierarchies.

Following the visionary ideas behind concepts such as *small analytics for big data* [163], *fusion cubes* [1], *drill-beyond* [37], or the *global cube* [91], OLAP and multidimensional analyses are highlighted as a perfect match for assisting and supporting the user when exploring the Web of Data. However, there is still a lack of research to automate the discovery of multidimensional schemata and enable automatic data exploration/crossing in the linked data setting. As a first step in this direction, a conceptual framework to perform exploratory OLAP over linked data has been proposed in [81]; the idea is to derive the multidimensional schemata from different data sources in order to run OLAP queries on the respective SPARQL endpoints. A key point recognized by the authors is that, due to the large volume and complexity of public knowledge bases, a user-guided process for multidimensional schema detection must be implemented. We believe iMOLD to be the first significative advance towards this goal. As already mentioned, iMOLD follows an instance-based approach but it neglects the problems identified in [84] by avoiding a pure supply-driven approach and incorporating the user to lead the process (as typically done in demand-driven approaches). As a consequence, iMOLD can be regarded as a *mixed* approach based on aggregation patterns. Unlike [123], we do not follow a probabilistic approach but one based on data modeling patterns (as typical of software engineering approaches) extracted from instances. Furthermore, by involving the user to guide the search we reduce the computational complexity of sampling instances.

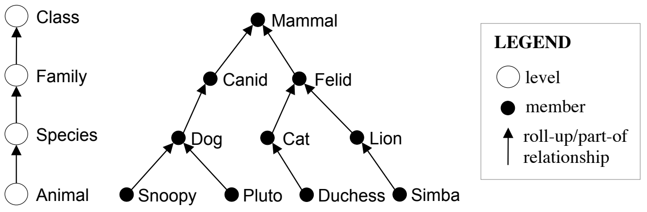


Fig. 6.1 An example of multidimensional modeling: levels and roll-up relationships (left), members and part-of relationships (right)

6.3 Background

6.3.1 Multidimensional modeling

In terms of multidimensional modeling, we build on Definitions 1 and 2 provided in Chapter 2 by providing also the definition of a hierarchy.

Definition 11 (Hierarchy) *A* hierarchy *h that conforms to a hierarchy schema $\mathcal{H}$ is a triple $h = \langle L, \succ, U\rangle$ where:*

- *L is the set of levels, where each level has a domain made by a set of members;*
- *$\succ$ is the partial order of L;*
- *U is the set of* roll-up relationships *in h; a roll-up relationship $u \in U$ is a triple (l_k, ρ, l_j) where $l_k, l_j \in L$ s.t. $l_k \dot{\succ} l_j$ and ρ is the semantics of the relationships* [1].

For simplicity, in this chapter we will consider roll-up relationships that abstract a many-to-one *part-of relationship* on the members of l_k and l_j, such that each member of l_k is part of exactly one member of l_j.

Example 17 *Consider the sample hierarchy in Fig. 6.1. Levels are shown as white circles; for instance,* Species *rolls-up to (i.e., is a child of)* Family*. Members are shown as black circles; for instance,* Canid *and* Felid *(member instances of* Family*) are part of member* Mammal *(member instance of* Class*).*

[1]Considering also the semantics ρ in the definition of roll-up relationship is necessary to cope with the case in which the same two levels are involved in two different relationships (e.g., persons can roll-up to cities according to two different semantics, namely *lives in* and *was born in*).

6.3.2 Linked data modeling

The Linked Data initiative was envisioned by Tim Berners-Lee as "published data that can be machine-readable, its meaning is explicitly defined, it is linked to other external data sets and can be linked to from other external data sets" [13]. In linked data, *Universal Resource Identifiers* (URIs) are used as names for available resources (note the *universal* scope of URIs as identifiers) and the HTTP protocol should be used to dereference URIs so that people can locate and look up those names.

The formalism used to describe and link resources is the *Resource Description Framework* language (RDF), a W3C recommendation. The basic RDF block is the triple, a binary relationship between a subject and an object; i.e., <*subject predicate object*>. The subject and the predicate must be resources (i.e., identified by a URI), whereas the object can be either a resource or a literal (i.e., a constant value such a string or an integer).

A set of RDF triples form an RDF graph. Indeed, it is rather usual to refer to RDF graphs as ontologies since RDF is considered to be the most basic ontology language. RDF Schema (RDFS), a W3C recommendation, was introduced to express basic constraints on RDF triples. By means of the RDFS core classes (namely rdfs:Resource, rdfs:Class, rdfs:Literal, rdf:Property, and rdf:Statement) and of some predefined properties one can distinguish between instances and classes (by using the rdf:type property), express property and class taxonomies by means of inclusion statements (by means of rdfs:subClassOf and rdfs:subPropertyOf) and type properties by specifying the allowed classes at its domain and range (by means of rdfs:domain and rdfs:range). RDF[2] ontologies can infer implicit knowledge from asserted triples (e.g., the subClassOf and subPropertyOf properties are transitive). The W3C recommends using the *Protocol and RDF Query Language* (SPARQL) to query RDF ontologies. Interestingly, SPARQL enables reasoning on RDF graphs by means of the SPARQL RDFS entailment regime[3]. Typically, RDF published graphs are exposed by means of (publicly available) SPARQL endpoints.

According to [13], reference RDF ontologies (typically referred to as *RDF vocabularies*; e.g., SKOS, FOAF, Schema, etc.[4]) should be created for each domain and reused whenever possible. Specific ontologies must then reuse the resources introduced in RDF vocabularies (i.e., refer to them in the newly created triples). Reusing instances or classes promotes linking data by pointing to external resources, whereas reusing

[2]Unless explicitly said, from here on we will use term RDF to refer to both RDF and RDFS.

[3]http://www.w3.org/TR/sparql11-entailment

[4]A detailed list of available vocabularies is at: http://lov.okfn.org/dataset/lov

properties facilitates the graph interpretation by using referent semantics defined in the vocabulary. QB4OLAP [39] is a reference conceptualization that extends the RDF Cube Vocabulary (QB[5]) with additional resources to allow structuring the cube dimensions in hierarchies and levels, relate measures with aggregation functions and represent observations at different aggregation levels, enabling roll-up and drill-down operations over RDF-based data [39]. As result, QB4OLAP captures all the constructs of the MultiDim model [168] and therefore all the multidimensional modeling features described in Section 6.3.1 —except for cardinalities, since RDF does not provide constructs to assert the cardinality of a property nor functional properties.

Following the reusability principle, we use QB4OLAP to annotate the aggregation hierarchies identified on RDF data. Similarly, we use SM4AM [170] to annotate the metadata generated during the discovery of aggregation patterns. SM4AM is able to represent user-related metadata such as queries, preferences and statistics, as well as system-related metadata such as data profiling and traceability metadata. In this chapter, we reuse the SM4AM *Data Quality*, *Navigational*, and *Ratings* artifacts to respectively store metadata regarding the source of the analyzed data, the users' interactions, and the users' preferences when using our tool. Section 6.4 elaborates on how these QB4OLAP and SM4AM are used in our approach.

6.4 Approach Overview

The work of this chapter is inspired by the *fusion cubes* vision [1], where a corporate, stationary data warehouse can be dynamically extended by the users on a self-service basis, by including some external situational data. In particular, this work supports the *data acquisition* step of [1], which requires a system capable of fetching relevant situational data from selected sources. Among the potential sources for situational data (e.g., sensor data, logs, social networks, etc.), linked data are a good candidate for building aggregation hierarchies, as they typically express a structured knowledge (either focused on a specific domain or cross-domain).

The basic idea of iMOLD is to extend the hierarchies in the corporate cubes through a user-guided process that explores selected linked data, derives hierarchies from them, and populates these hierarchies with data. This is done by identifying in the linked data the recurring modeling patterns that express roll-up relationships between RDF concepts and translating them into multidimensional knowledge, to be stored locally and shared with every user for reuse purposes. The knowledge base built

[5]http://www.w3.org/TR/vocab-data-cube

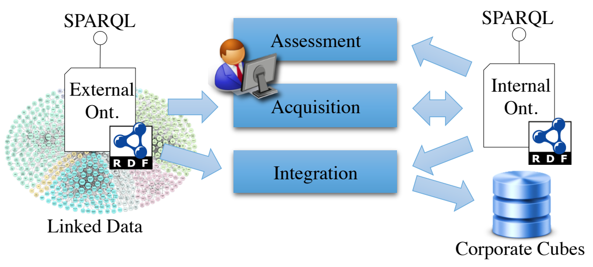


Fig. 6.2 Functional view of the iMOLD approach

and maintained by the system to store such information is called *Internal Ontology* (IO), whereas any source of linked data will be referred to as an *External Ontology* (EO).

From a functional point of view, the user locates a concept of interest in a selected EO (e.g., the concept of city on DBpedia), then she uses it as a starting point to build her hierarchies. The typical scenario that we envision (shown in Fig. 6.2) can be subdivided into three iterative phases.

1. **Assessment:** the user accesses the IO to check whether the concept of interest is already present and which conceptual schemata have already been built around it. The user is able to reuse the information previously acquired, either by herself or by other users in the company. Therefore, it is expected that this phase will increasingly acquire importance as the IO is filled with new information over time.

2. **Acquisition:** if the concept of interest is not present in the IO or it is not satisfactorily modeled (either because it is outdated, not relevant, or misaligned with the user's requirements), the user can search for aggregation patterns in the EOs, build her own hierarchy by selecting the concepts of interest, provide appropriate names to levels, and integrate the results into the IO.

3. **Population:** the user launches a set of system-generated queries that create and populate new dimension tables with the data selected. These dimension tables are then integrated with those in the corporate cubes to enable richer analyses.

The first phase is mainly a matter of delivering a smart user interface for effectively browsing the IO. The third phase requires to automatically create and execute some SPARQL queries to extract data, to transform and load them into ad hoc dimension tables, and to establish a correlation between rows in these dimension tables and rows in the corporate dimension tables. The only demanding step in this phase is the last

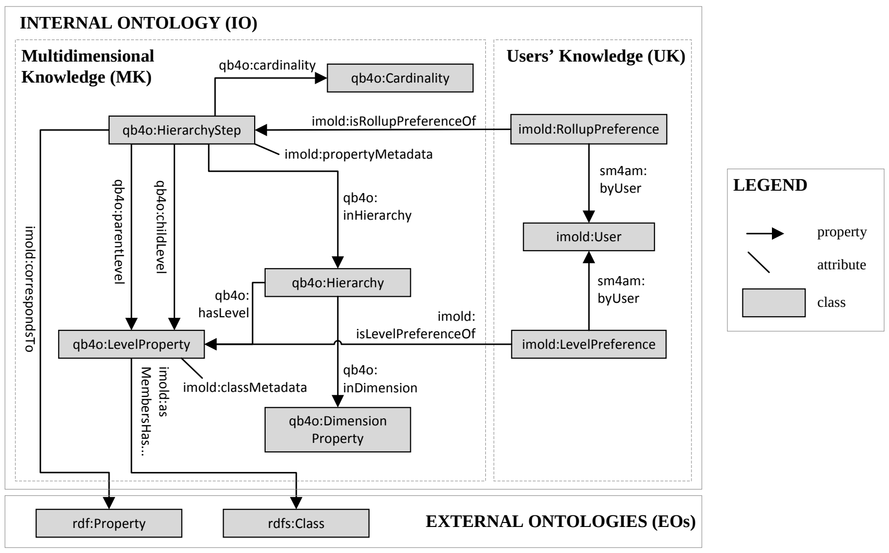


Fig. 6.3 The Internal Ontology (for simplicity, a property at the class level is represented as an arc linking the domain class to the range class, which implies the correct definition of the property by means of the rdfs:domain and rdfs:range properties)

one, which requires inter-member mappings to be found; though this problem has not been deeply investigated in the literature, some existing approaches could be adapted to implement it (e.g., [22, 87, 62]). The second phase, on which this chapter is focused, is the one that raises the most challenging and novel research issues.

In the remainder of this section we explain how the IO is structured. The IO is the container of all the information discovered and acquired by the users through the exploration of the EOs, and its role is similar to the one of catalogs in relational DBMSs. As already mentioned, the IO is collaboratively built and maintained by the users for reuse purpose. This means that the content of the IO is not limited to the information acquired from the EOs, but it also contains the metadata that enable its future reuse. In particular, the IO can be subdivided into two areas, as depicted in Fig. 6.3:

- **Multidimensional Knowledge** (MK), the core area that stores the multidimensional interpretation of the data explored in the EOs, i.e., the aggregation hierarchies detected.
- **Users' Knowledge** (UK) that marks the portions of the MK selected by the different users to build their hierarchies, thus serving as a sort of log.

The structure of each area mainly relies on the existing vocabularies that already propose a solution in these contexts. In particular, we adopt QB4OLAP for the MK, and SM4AM for the UK. However, the mere reuse of the two vocabularies is not sufficient: not only we need to instantiate the metamodel provided by SM4AM, but we also need to extend the original vocabularies with custom classes and properties. To this end, we define a new namespace (http://big.csr.unibo.it/imold#, prefix: imold) to create the instance of the metamodel and the additional classes and properties. In the following paragraphs we briefly discuss the content of the two areas of the IO; a complete glossary for the classes and properties used in the IO is included in Table A.2 in the Appendix.

As anticipated, the concepts found in the EOs are annotated in the MK according to the QB4OLAP vocabulary. In particular, we instantiate the classes qb4o:LevelProperty and qb4o:HierarchyStep to define, respectively, hierarchy levels and roll-up relationships between them, while the multiplicities of the latter are specified through the qb4o:Cardinality class. The mapping of levels and roll-up relationships on the EO (i.e., the identification of the original classes and properties from which these concepts are extracted) is made through three custom properties, imold:asMembersHasInstancesOf, imold:asMembersHasSubClassesOf, and imold:correspondsTo, which materialize the link between the IO and the EO. Noticeably, imold:asMembersHasInstancesOf and imold:asMembersHasSubClassesOf are used to retrieve the members of each level when hierarchies are populated with data. Finally, we extend QB4OLAP vocabulary with custom annotation properties in order to store statistical information regarding a level property or a hierarchy step (e.g., the number of instances of a class).

In the UK, we use the SM4AM vocabulary to define a model for the metadata that need to be saved. Specifically, we define the class imold:User (instance of sm4am:User) to model the users and we define the concepts of imold:LevelPreference and imold:RollupPreference (both instances of sm4am:UserAction) to represent the preference towards a specific level of the hierarchy or a specific roll-up relationship expressed by a user. Each preference is linked to its corresponding instance in the MK through the imold:isLevelPreferenceOf or imold:isRollupPreferenceOf property (both instances of sm4am:usesSchemaComponent), but it can be linked to multiple users through the imold:byUser property (instance of sm4am:byUser), meaning that different users may share one or more preferences on the MK.

Table 6.1 Hierarchy-related concepts and their representations within the different aggregation patterns

Concept	Patt. (1a)	Patt. (1b)	Patt. (1c)	Patt. (2a)	Patt. (2b)
Parent Level	Class	Class	—/Class	Powertype	Powertype
Roll-up rel.	Assoc.	—	—	—	—
Child Level	Class	Class	Class/—	Class	Powertype
Parent member	Instance	Instance	Instance	Class	Class
Part-of rel.	Assoc.	Assoc.	Assoc.	Instantiation	General.
Child member	Instance	Instance	Instance	Instance	Class

6.5 Aggregation Patterns in Ontologies

To semantically enrich the ontological knowledge about the data at hand, we propose to identify aggregation patterns in RDF data that would most likely correspond to potential hierarchies. To this end, we make two assumptions:

1. Though the EOs that we query may be incomplete (since they follow the open world assumption) and not fully correct, their data are statistically representative.
2. OLAP aggregation hierarchies only come from many-to-one relationships.

In the following, to explain the aggregation patterns, we will use the UML terminology as a reference for better clarity. Table 6.1 summarizes the five RDF patterns that can give rise to roll-up relationships; each cell shows how each hierarchy-related concept (as defined in Section 6.3) is represented according to each pattern. Out of all the concepts in the UML metamodel, we consider only those that have a correspondence in RDFS. Thus, as to classifiers, we consider classes (i.e., rdfs:Class) and datatypes (i.e., rdfs:Datatype); as to relationships, we consider associations (i.e., generic rdf:Property), generalizations (i.e., rdfs:SubclassOf), and instantiations (i.e., rdf:Type). Therefore, hierarchy levels and members can be mapped into either classes or data types, while roll-up and part-of relationships can be mapped into associations, generalizations, or instantiations. Out of all possible combinations, only the five in Table 6.1 make sense in this context, and they can be classified depending on whether they are association- or generalization-based. In the following they are described in more detail, using Fig. 6.4 as a general reference and the sample hierarchy depicted in Fig. 6.1 as an example.

6.5.1 Association-Based Patterns

OLAP aggregation hierarchies express part-whole relationships [3], which in UML are typically represented as either associations (e.g., a product *has* a brand) or aggregations (e.g., a city *is part of* a region). The part-of semantics of UML aggregation is not

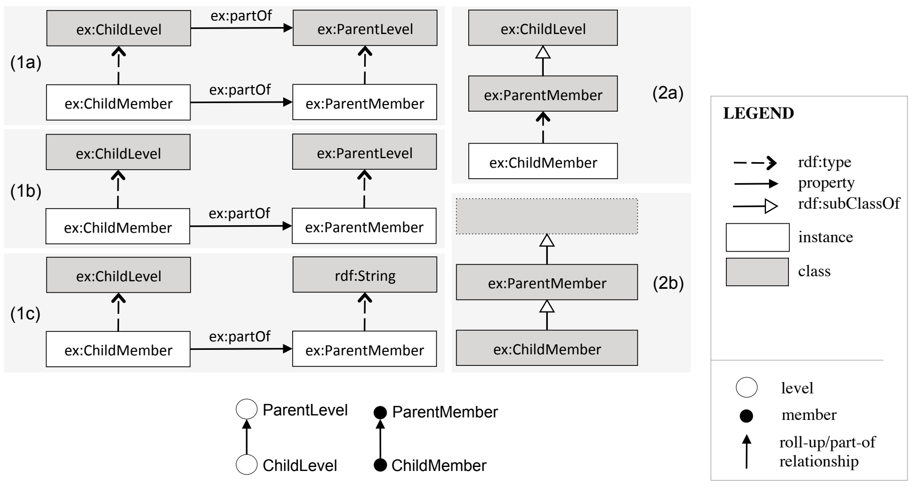


Fig. 6.4 RDF aggregation patterns represented in UML (top) and their multidimensional counterpart (bottom)

explicitly modeled in RDF, so we will just consider associations, which are the immediate superclass of aggregations in the UML metamodel and are represented in RDF using properties. So, as exemplified in Fig. 6.4, pattern (1a) corresponds to two classes (e.g., ex:Family and ex:Class in Fig. 6.5.a) related by an RDF property.

In cross-domain ontologies, associations are rarely defined at the model level, i.e., through rdfs:domain and rdfs:range properties; even in those cases, they are typically defined for very abstract classes only (e.g., property yago:isLocatedIn has yago:yago-PermanentlyLocatedEntity as domain and yago:yagoGeoEntity as range). So, because of the incompleteness of linked data (assumption ♯1 in Section 6.5), we need to relax pattern (1a) by allowing the property not to exist at the level of classes, giving raise to pattern (1b).

The third association-based pattern is (1c), which applies when the instances are associated to a data type instead of a class (i.e., we have literals instead of objects). Everything works as for the other two variants, but in this case there is no class modeling the parent (child) level. Consequently, the name of the parent (child) level must be either provided by the user or derived from the name of the property.

Either if the property exists at the class level or only at the instance level, we cannot rely on the existence of multiplicities. Thus, in all three patterns we have to sample the associated instances to check the proportion of existing many-to-one relationships (e.g., we retrieve from the SPARQL end-point how many instances of Class are related to each instance of Family). Then, also considering assumption ♯1 in Section 6.5 about potential incorrectness of linked data, these patterns are identified only if the average

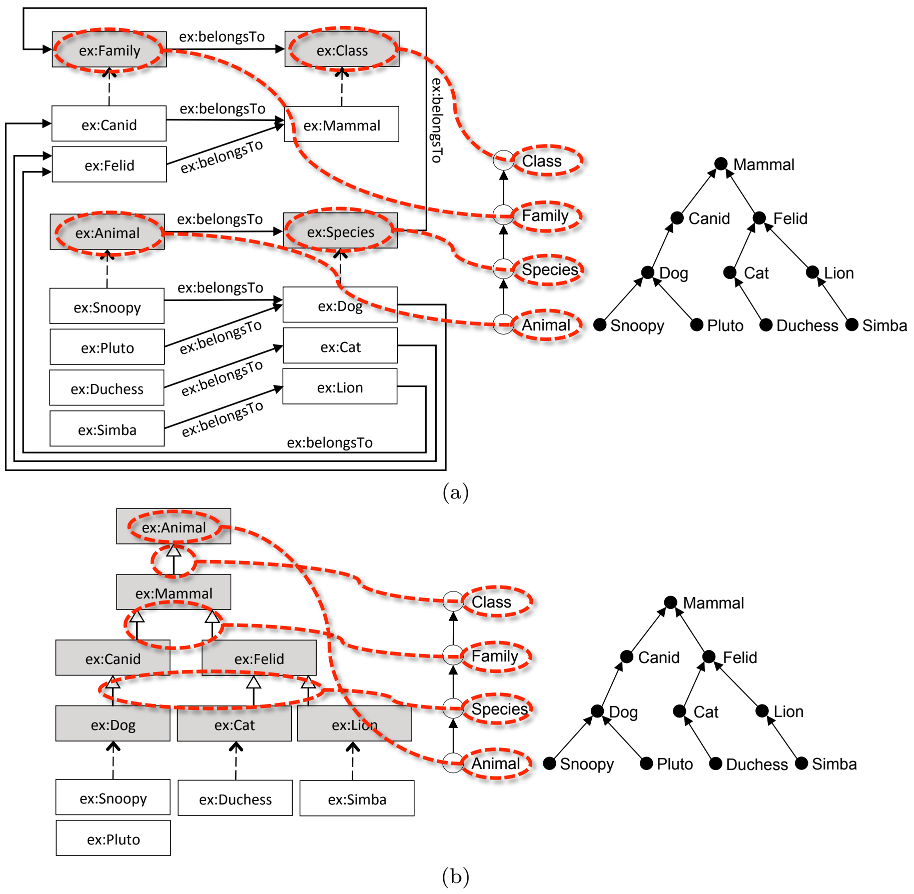


Fig. 6.5 The hierarchy in Fig. 6.1 modeled in RDF using associations (a) and generalizations (b); in thick dashed lines, the correspondence between ontology concepts and hierarchy levels

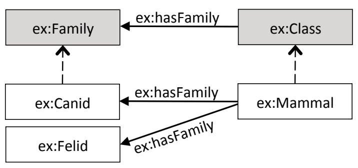


Fig. 6.6 Alternative RDF representation for the aggregation between ex:Family and ex:Class

cardinality of the association is close to one on the side of the parent, i.e., only if an *approximate functional dependency* holds [96, 78].

We close this section by observing that the ontology designer may have modeled the conceptual aggregation in one direction or the other. For instance, the aggregation between ex:Family and ex:Class in Fig. 6.5.a could also have been modeled with the RDF property ex:hasFamily, where ex:Class is the domain and ex:Family is the range, as depicted in Fig. 6.6. For this reason, in patterns (1a) and (1b) the child and parent roles can be inverted with regard to those shown in Fig. 6.4. Note that this cannot happen with pattern (1c), because the subject of an RDF triple cannot be a literal.

6.5.2 Generalization-Based Patterns

The second group of patterns is based on generalization. Generalizations express an *is a* semantics that induces a subsumption between sets of instances of related classes. In this sense we can say that, if one set of instances is superset of another, the corresponding classes are generalization of one another —for instance, Felid generalizes Cat and Lion since the set of animals instances of Felid is superset of the set of animals instances of Cat and Lion, and the same holds for Canid and Dog. But then, animals can be grouped into felids and canids, or into cats, lions, and dogs, and the former grouping is coarser than the latter, which in OLAP terms translates to a part-of relationship between members Cat + Lion and Felid on the one hand, between Dog and Canid on the other. This suggests that there is roll-up relationship between two different levels, which we will call Species and Family.

From the example above we can conclude that, differently from what seen in the previous subsection, here the classes in the EO (Felid, Cat, etc.) correspond to members instead of levels. So, to find an expression of levels in this case, we must look further. Indeed, though generalizations are binary relationships between pairs of classes, from a conceptual point of view they can be grouped depending on the criteria used (a.k.a. *powertype* or *generalization set* in UML terminology). Thus, in pattern (2a), the class corresponding to the child level is specialized into subclasses (i.e., subsets), each corresponding to a parent member, using the parent level as powertype. For instance, as depicted in Fig. 6.5.b, class Animal is transitively specialized into Dog, Cat, and Lion based on powertype Species. Therefore, these three subclasses (i.e, subsets) give rise to three parent members of level Species, and their instances (Snoopy, Pluto, etc.) to child members of level Animal.

Powertypes are not made explicit in RDF. In principle, they could be explicitly represented at the metalevel, i.e., using metaclasses (e.g., class Dog could be an instance of metaclass Species). However, this type of metamodeling is extremely rare in linked data (for instance, in DBpedia and YAGO there are just a very few metaclasses, and all of them are so abstract that they cannot be useful for building domain-specific hierarchies), so the user will have to provide names for the levels corresponding to powertypes upon detecting the pattern.

While in pattern (2a) child members correspond to instances (notice that the rectangle is white) in the linked data, in (2b) they correspond to classes (grey rectangle). This may happen because of incompleteness or because of a different level of abstraction chosen by the ontology designer. In this case, the different classes corresponding to child members (e.g., Canid and Felid in Fig. 6.5.b) would be generalized into a superclass (e.g., Mammal) that would be the corresponding parent member. Usually, the parent and child levels (Class and Family, respectively) will not be represented in the ontology, because they correspond to powertype, so they will be provided by the user.

As we will show in Section 6.6.2, the combination of (2a) and (2b) allows to iteratively build OLAP hierarchies based on conceptual generalization relationships. Remarkably, though generalization-based patterns are less intuitive than association-based ones, they do not require data to be queried, nor they rely on probabilistic assumptions like those made for association multiplicities.

We finally note that, in case of an overlapping powertype (i.e., individuals are instances of more than one subclass and subsets are not disjoint), a many-to-many relationship arises; since in multidimensional modeling only many-to-one relationships are normally considered (assumption ♯2 in Section 6.5), we will not consider this case. Besides, there is also a possibility of having a multiple specialization (i.e., the same class can be a specialization of several superclasses), which can be interpreted as multiple many-to-one relationships with different semantics thus generating a branch in the hierarchy. For instance, Fig. 6.7 shows how the generalization of ex:Dog, ex:Cat, and ex:Lion into Domestic and Wild leads to creating a branch towards level Dom/Wild.

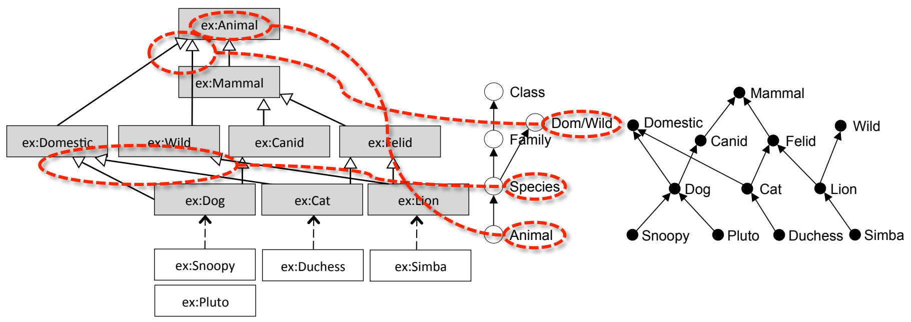


Fig. 6.7 Double specialization for classes ex:Dog, ex:Cat, and ex:Lion

6.6 Acquisition

In this section, the core phase of iMOLD is described in detail; its goal is to build aggregation hierarchies out of ontological knowledge by letting users extract the specific roll-up relationships of interest.

As previously stated, the acquisition of multidimensional knowledge is done by detecting aggregation patterns on a selected EO. Exploring an EO in its entirety to find every potential roll-up relationship is clearly unfeasible. Noticeably, some approaches have been devised in the literature for effectively exploring the linked data cloud using advanced visualization techniques or providing high-level conceptualizations (e.g., [73, 20]). However, since in this work we focus on how to give a multidimensional shape to linked data, we intentionally adopt a very basic approach for supporting the user interaction with the EO. Consequently, we simply require the user to first choose a class of interest c in an EO, to be used as an entry point for pattern detection. This class corresponds to a hierarchy level l, so it is mapped into the MK by creating an instance imold:c of qb4o:LevelProperty; since the members of l correspond to the instances of c, a connection between imold:c and c is established via property imold:asMembersHasInstancesOf (e.g., see Species in Fig. 6.8).

Every detection is focused, besides on c, on a direction dir, which can be either *outbound* or *inbound*; this means that, given c and dir, we detect the patterns by exploring the triples where c (or its instances) is either the subject (dir = 'outbound') or the object (dir = 'inbound'). Indeed, given a relationship a (either part-of or subclass-of) between subject s and object o, a is equally detected and modeled in the MK either by starting from s and moving to o in the outbound direction, or by starting from o and moving to s in the inbound direction.

Each pattern is mapped into a roll-up relationship, which is systematically included in the MK for possible reuse, while it may be included or not in the UK (hence, in

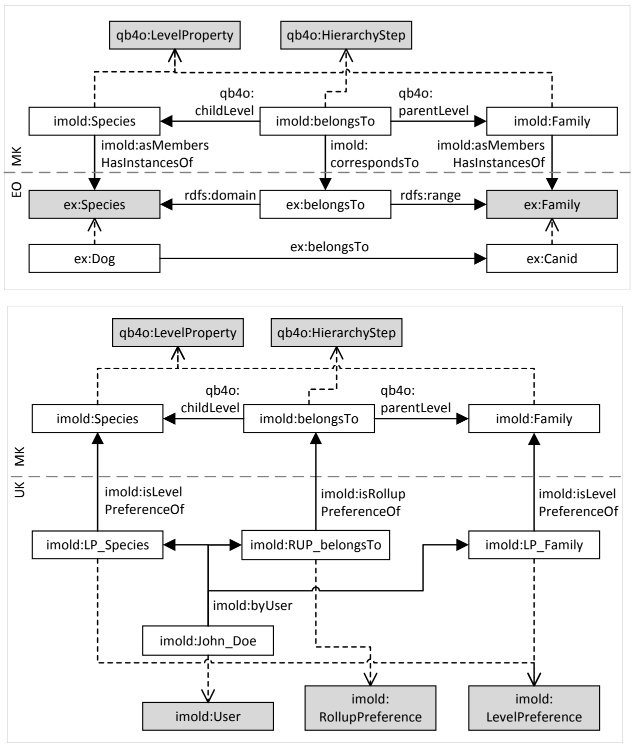


Fig. 6.8 Mapping the MK into the EO (top) and the UK (bottom)

the user hierarchy being built) upon the user's judgement. The detection process is done in a breadth-first fashion: this means that c is completely analyzed in its relationships with other classes or datatypes but no recursive detection is done to avoid an exponential explosion. The roll-up relationships selected by the user lead to new classes, from which the user can iteratively perform new searches.

In Section 6.5, we have defined two groups of aggregation patterns: one based on associations and one based on generalizations. Following this distinction, two different approaches are adopted to detect such patterns; Sections 6.6.1 and 6.6.2 will explain them in detail. For simplicity, we will start by assuming that the two groups of patterns are always applied separately; in Section 6.6.3 we will briefly discuss under which conditions they can be mixed. Then, in Section 6.6.4 we illustrate the user experience we have conceived and implemented to support the acquisition phase. Finally, in Section 6.6.5 we discuss some issues related to collaboration and knowledge reuse in iMOLD.

The detection of all patterns relies on a SPARQL query, to be directly submitted to the SPARQL endpoint of a selected EO. As the goal of patterns is to reach new concepts of interest from a starting one, the possible presence of materialized entailments in the

EO may lead to overwhelming complexity, both in the executed queries and in the retrieved results. For this reason, the queries use specific filters aimed at avoiding using triples derived from inferred classifications (e.g., <ex:a rdf:type ex:c> if <?a rdf:type ?b> and <?b rdfs:subClassOf ?c>); these filters are omitted in the query descriptions given below for the sake of simplicity.

Finally, we recall that a key feature of linked data is that of creating connections between different ontologies, so as to enable the reuse of knowledge. In an ontology, this connection is provided by pointing to objects of a different ontology with their original URIs. An example is the triple <dbpedia:Barcelona rdf:type yago:City108524735>, specified in DBPedia, which reuses a class defined in YAGO, therefore providing a link between the two ontologies. In iMOLD, the connectivity of linked data is exploited to enable users to jump from the currently explored ontology to a different one whenever the application of a pattern detects an external concept (i.e., one whose namespace is different from the one of the starting concept). This transition can be seamlessly made by launching the next searches for patterns on the SPARQL endpoints of both ontologies and then merging the results; however, for the sake of simplicity, in the following subsections the description of the acquisition algorithms will consider a single endpoint.

6.6.1 Acquisition of Association-Based Patterns

In the case of associations, the goal of the patterns is to determine whether a property p involving c can be mapped to a roll-up relationship, where the domain and range of p are mapped to a child and a parent level (or vice versa) in the hierarchy.

Definition 12 (Association) *An* association *is a triple $a = (d, p, r)$ where p is a property, d is a class that represents the domain of p, and r is either a class or a datatype that represents the range of p. Association a is characterized by its* right cardinality *$rightCard(a)$, i.e., the average number of distinct instances of r linked to each instance of d through p, and by its* left cardinality *$leftCard(a)$, i.e., the average number of distinct instances of d linked to each instance of r through p. Given $a = (d, p, r)$, we denote with a^{-1} its inverse, $a^{-1} = (r, p, d)$.*

An association a is a roll-up relationship if its multiplicity is either many-to-one or one-to-many; in particular, a corresponds to a roll-up relationship $u = a$ if its multiplicity is many-to-one, to a roll-up relationship $u = a^{-1}$ if its multiplicity is one-to-many. Since the RDFS vocabulary does not provide means to describe the multiplicity of a property, the only way to determine the multiplicity of a is through a

Table 6.2 Parameters used for detecting association patterns

Parameter	Constraint	Default	Description
maxCard	> 0	1000	Upper bound to the number of instances of c to be sampled
multTol	≥ 1	1.1	Tolerance for giving -to-one multiplicity to an association
maxRels	> 0	20	Upper bound to the number of roll-up relationships to be returned

Algorithm 1 Detect Association-Based Patterns

```
Input pt: a pattern (either 1a, 1b, or 1c), EO: an external ontology, MK: the multidimensional knowledge, c: a
    starting class, dir: a direction (either 'outbound' or 'inbound'); maxCard, multTol, and maxRels: the search
    parameters
Output U: a set of roll-up relationships
 1: if $card(c) \leq maxCard$ then                          ▷ Random offset for query $q$
 2:     $offset \leftarrow 0$
 3: else
 4:     $offset \leftarrow Random(0, card(c) - maxCard)$
 5: $U \leftarrow \emptyset$                                          ▷ Initialize $U$
 6: $q \leftarrow Query(c, dir, pt, maxCard, offset)$                ▷ Create $q$...
 7: $A \leftarrow Execute(EO, q)$                       ▷ ...and execute it against $EO$
 8: for each $a \in A$ do                     ▷ Find the roll-up relationships in $A$
 9:     if $rightCard(a) \leq multTol$ then                 ▷ If $a$ is many-to-one...
10:         $U \leftarrow U \cup \{a\}$                              ▷ ...add it to $U$
11:     else
12:         if $leftCard(a) \leq multTol$) then             ▷ If $a$ is one-to-many...
13:             $U \leftarrow U \cup \{a^{-1}\}$                 ▷ ...add its inverse to $U$
14: $MK \leftarrow MK \cup NewRURel(U)$                                ▷ Update $MK$
15: $U \leftarrow Top(U, maxRels)$             ▷ Keep the top-$maxRels$ relationships
16: return $U$
```

statistical analysis at the instance level, which means inspecting the relationships in which the instances of d and r are involved.

All association patterns share the need for inspecting instances. Therefore, the detection of these patterns can be implemented with a single generic method, discussed below. First of all, we emphasize that finding the roll-up relationships involving a given class d (which asks for determining the multiplicities of all associations involving d by querying their instances) may require a huge number of triples in the EO to be accessed. On the one hand, SPARQL endpoints may fail to provide the results of such a query within a reasonable time, as they may not have enough computational power available —or they may even put a restriction on the maximum number of triples processable by a query, possibly leading to an incomplete result. On the other hand, the list of resulting roll-up relationships may be too long and potentially chaotic for the user. For this reasons, we provide some parameters aimed at decreasing the complexity of both the search and the listing of results, at the price of introducing some uncertainty. These parameters are shown in Table 6.2 together with their default values and explanations.

The pseudocode for detecting association-based patterns is shown in Algorithm 1. We start by randomly generating an *offset*, aimed at inducing some randomness in the selection of the sample from the instances of c (lines 1—4). Then, a SPARQL query q is generated by function *Query* (line 6); given a starting class c and an offset, q returns a set A of associations involving c in direction dir, together with the left and right cardinality of each association. The specific form of q depends on the pattern pt and on the search parameters; for instance, this is the query generated for pattern (1a) in the outbound direction (see Table A.1 in the Appendix for an explanation of the variables):

```
SELECT ?p ?class (?nProp/?nO AS ?rightCard) (?nProp/?nS AS ?leftCard) ?nO ?nS
WHERE
{   SELECT ?p ?class (COUNT(*) AS ?nProp) (COUNT(DISTINCT(?o)) AS ?nO)
           (COUNT(DISTINCT(?s)) AS ?nS)
    WHERE
    {   ?p rdfs:domain ?c .                        ▷ Step 1: retrieve the properties of c
        ?p rdfs:range ?class .
        ?s a ?c .                                  ▷ Step 2: retrieve the property instances
        ?s ?p ?o .
        ?o a ?class
    }
    GROUP BY ?p ?class        ▷ Step 3: group the property instances to get the list of associations
}
```

The query for pattern (1b) does not work at the class level, and adds a check on the maximum number of instances of c to avoid overloading the endpoint:

```
SELECT ?p ?class (?nProp/?nO AS ?rightCard) (?nProp/?nS AS ?leftCard) ?nO ?nS
WHERE
{   SELECT ?p ?class (COUNT(*) AS ?nProp) (COUNT(DISTINCT(?o)) AS ?nO)
           (COUNT(DISTINCT(?s)) AS ?nS)
    WHERE
    {   {   SELECT ?s                                  ▷ Step 1: select instances of c
            WHERE
            {   ?s a ?c .
            }
            LIMIT ?maxCard
            OFFSET ?offset
        } .
        ?s ?p ?o .                                 ▷ Step 2: retrieve the properties of each s
        ?o a ?class .                              ▷ Step 3: retrieve the classification of each o
    }
    GROUP BY ?p ?class        ▷ Step 4: group the property instances to get the list of associations
}
```

When pattern (1c) is applied (i.e., when literals are inspected), at Step 3 the class of ?o, ?class, is replaced by its datatype, str(datatype(?o)). However, note that this pattern can only be applied in the outbound direction, because literals can only be objects in RDF triples.

Function *Execute* (line 7) submits q to the SPARQL endpoint of the EO. In the lines from 8 to 13, the associations in A are filtered according to their multiplicities and added to U. Threshold $multTol$ is applied to left and right cardinalities to determine if each association a can be considered as either many-to-one or one-to-many (lines 9 and 12). Then, all roll-up relationships in U are added to the MK using function $NewRURel$ (line 14). The mapping of each $u = (l, p, l')$ into the MK is carried out as follows:

- l and l' are mapped into two instances of qb4o:LevelProperty; the members of l and l' correspond to the instances of the domain and range of p, so a connection with the corresponding classes in the EO is established via property imold:asMembersHasInstancesOf;
- property p is mapped into an instance of qb4o:HierarchyStep, which is linked to the two corresponding qb4o:LevelPropertys by means of the qb4o:parentLevel and qb4o:childLevel properties, and also linked to its counterpart on the EO by means of the imold:correspondsTo property;
- the newly-created instances of qb4o:LevelProperty and qb4o:HierarchyStep are enriched with metadata about their labels, descriptions, and multiplicities.

Finally, U is sorted and its top-$maxRels$ elements are returned to the user, who will select those to be included in her UK. As to sorting, since computing graph-based ranking measures through a SPARQL endpoint would be hardly feasible, a local ranking strategy must be defined. To this end, let u be a roll-up relationship and c be the level of u corresponding to the starting class; we claim that a valuable indicator for the relevance of u is given by the support $supp(p)$ of p in c, i.e., the percentage of instances of c that are involved in p.

Example 18 *With reference to the example depicted in Fig. 6.5.a, Fig. 6.8 (top) shows the relationship between the MK and the EO. For instance, level* Species *in the MK has instances of class* ex:Species *in the EO as members. Fig. 6.8 (bottom) shows how the UK is related to the MK. Here, we represent the fact that user John Doe is interested in species, families, and in their roll-up relationship with semantics* belongsTo*.*

6.6.2 Acquisition of Generalization-Based Patterns

Generalization-based patterns are cheaper to detect than association-based ones because (i) no query at the instance level is required and (ii) the only inter-class link that must be considered is rdfs:subClassOf. On the other hand, their interpretation is less intuitive, because the kind of transformations applied to concepts move them along the instantiation-classification dimension. Whereas distinct classes always correspond to distinct levels in association-based patterns, in generalization-based ones distinct subclasses that belong to the same superclass can be grouped together to become members of a single level, which corresponds to the powertype of the subclasses. Furthermore, differently from association-based patterns, generalization-based ones always require additional information from the user to give names to powertypes.

Definition 13 (Generalization) *A* generalization *is an association $g = (d, p, r)$ where d and r are the subclass and the superclass, respectively, and $p =$* rdfs:subClassOf*. We denote with $PT(g)$ the (user-provided) powertype to which g belongs.*

Consistently with our acquisition approach, the generalizations g involving a given class c are detected by navigating the rdfs:subClassOf properties according to direction dir: the superclasses of c are found by bounding d to c and taking $dir =$ 'outbound', while the subclasses of c are found by bounding r to c and taking $dir =$ 'inbound'. In both cases, an interaction with the user is necessary to filter out non-relevant generalizations; more specifically:

- A class c may be specialized according to different powertypes. Since powertypes are normally not modeled in RDF, when operating in the inbound direction the user must manually select the subclasses of c that belong to the powertype of interest and provide its name.
- Multiple specialization is allowed in RDF. Thus, when operating in the outbound direction, the user must manually select one or more superclasses of interest when searching for generalizations of c.

To create multi-level aggregation hierarchies, generalizations must be iteratively navigated. This can be done adopting either (i) a top-down strategy, where a general concept is selected first and the inbound direction is followed to iteratively find its subclasses; or (ii) a bottom-up strategy, where a detailed concept is selected first and the outbound direction is followed to iteratively find its superclasses; or (iii) a mix of these two. In any case, the identification of a powertype leads to creating a new level in the MK.

The pseudocode for detecting generalization-based patterns —working for both the top-down and bottom-up strategies— is shown in Algorithm 2. The first operation (line 2) is the creation of a SPARQL query q that returns the set S of either the superclasses

Algorithm 2 Detect Generalization-Based Patterns

```
Input EO: an external ontology, MK: the multidimensional knowledge, c: a starting class, dir: a direction (either
    'outbound' or 'inbound'), H: a hierarchy including a level $l_c$ corresponding to c
Output U: a set of roll-up relationships
 1: $U \leftarrow \emptyset$                                          ▷ Initialize U
 2: $q \leftarrow Query(c, dir)$                                      ▷ Create q...
 3: $S \leftarrow Execute(EO, q)$                                     ▷ ...and execute it against EO
 4: for each $s \in S$ do                                             ▷ Find the roll-up relationships corresponding to S
 5:     if $dir =$'inbound' then                                      ▷ Moving top-down: s is a subclass of c
 6:         $g = (s, \text{rdfs:subClassOf}, c)$
 7:         if H only includes level $l_c$ then                       ▷ First iteration
 8:             $U \leftarrow \{(l_c, \text{subClassOf}, PT(g))\}$
 9:         else                                                      ▷ Other iterations
10:             $U \leftarrow \{(l_c, \text{subClassOf}, PT(g)), (PT(g), \text{subClassOf}, Par(l_c))\}$
11:     else                                                          ▷ Moving bottom-up: s is a superclass of c
12:         $g = (c, \text{rdfs:subClassOf}, s)$
13:         $U \leftarrow \{(l_c, \text{subClassOf}, PT(s))\}$
14: $MK \leftarrow MK \cup NewRURel(U, H)$                             ▷ Update MK
15: return U
```

or the subclasses of c, according to direction dir. For instance, the query generated for dir = 'inbound' is as follows:

```
SELECT DISTINCT ?type
WHERE
{   ▷ Retrieve the superclasses of c
    ?type rdfs:subClassOf ?c .
}
```

Then, function *Execute* (line 3) submits q to the SPARQL endpoint of the EO.While mapping each class of S into a generalization g is trivial because it simply depends on the direction dir (lines 6 and 12), mapping each g into a roll-up relationship in U and then into the MK is more complex, as the way the mapping is done also depends on the current state of the aggregation hierarchy, H. To explain how this is done, for simplicity we restrict to the case in which pattern (2a) is applied, i.e., the members of the bottom level of the hierarchy to be built correspond to instances (the mapping process for pattern (2b) is similar). We start by describing the process for the top-down strategy (dir = 'inbound', lines 5—10), using Fig. 6.9 as a reference example.

0. Before the first iteration, the user has selected the starting class c (Animal in Fig. 6.9.a) which has been mapped into a level of the MK (i.e., an instance of qb4o:LevelProperty) as already described.

1. At the first iteration (line 7), pattern (2a) is applied and the subclasses of c are found (e.g., Mammal in Fig. 6.9.b). For each powertype gs declared by the user, a new level l_{gs} (Class in the example) is created in the MK as an instance of qb4o:LevelProperty and the corresponding subclasses are mapped in the MK as

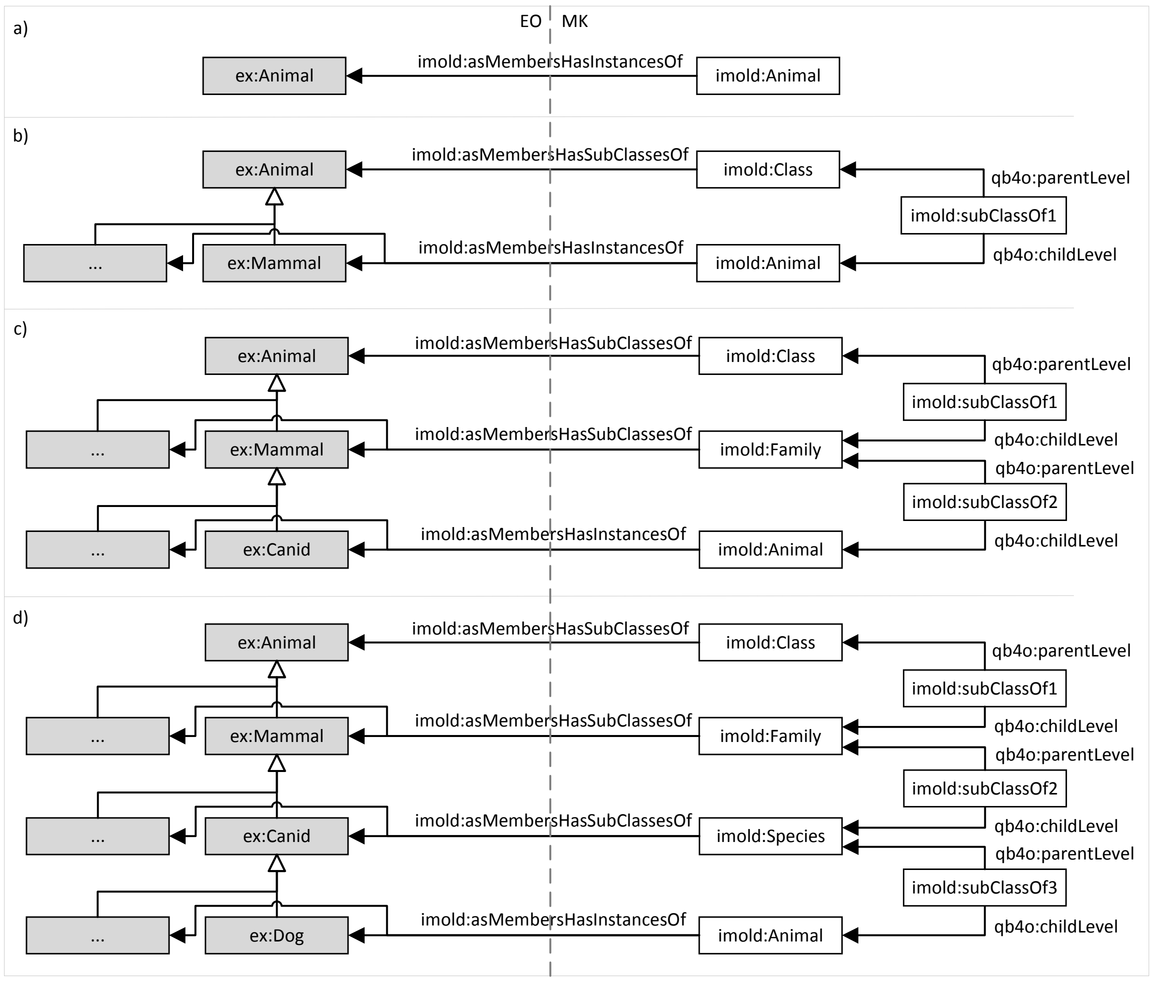


Fig. 6.9 Top-down strategy for detecting generalization-based patterns; the link between the instances in the MK and their classes (i.e., qb4o:LevelProperty and qb4o:HierarchyStep) is omitted for simplicity

members of l_{gs} via property imold:asMembersHasSubClassesOf. Moreover, the imold:asMembersHasInstancesOf property of the level l_c corresponding to c is pushed down to the subclasses of c (in the example, from Animal to Mammal and its siblings), to denote that the members of l_c are currently represented by all the instances of these subclasses. Finally, a new roll-up relationship from l_c to l_{gs} is created as an instance of qb4o:HierarchyStep.

2. At each following iteration (line 9), let $Par(l_c)$ be the parent of l_c in H. For instance, taking Fig. 6.9.c, let now c correspond to Canid, which implies that l_c and $Par(l_c)$ correspond to Animal and Family, respectively. Even in this case, after the subclasses of c have been found (e.g., Dog), a new level l is created in the MK for each powertype declared by the user (e.g., Species). However, since the new level must be inserted in the hierarchy between l_c and $Par(l_c)$, the existing roll-up relationship from l_c to $Par(l_c)$ is replaced by two roll-up relationships, one from l_c to l_{gs} and one from l_{gs} to $Par(l_c)$.

Note that, when a top-down strategy is followed to build a hierarchy H, the domain of H (meant as the domain of the finest level in H) progressively gets more selective as new levels are explored. For instance, in the example of Fig. 6.9, from all animals it shrinks to all canids. So, if *all* animals are actually relevant to the user, she has to explore all the other subclasses (by explicitly navigating through reptiles, birds, etc.). This is because Alg. 2 adopts a “lazy”, class-wise approach, i.e., it maps into the MK (as members) only the classes that are explicitly selected by the user. Remarkably, the algorithm can be easily modified to implement a more “eager”, level-wise approach where exploration is implicitly carried out by levels. For instance, at the iteration depicted in Fig. 6.9.c, this means enabling the user to select the subclasses of interest not only from the subclasses of Canid, but even from those of Felid and all the other families.

To conclude this section, we briefly exemplify the process for the bottom-up strategy (dir = ’outbound’, lines 11—13). Let c be the starting class, s one of its superclasses, and gs its powertype. Differently from the top-down case, here the hierarchy domain progressively enlarges as new levels are discovered, so when the roll-up relationship from l_c to l_{gs} is added, l_c must be renamed to s. Let for instance Dog be the starting class c, initially represented in the MK with a level l_c named Dog as depicted in Fig. 6.10. If Canid is selected in the EO as a relevant superclass of Dog, with powertype Species, a new level named Species is created, l_c is renamed to Canid, and a new roll-up relationship from Canid to Species is added.

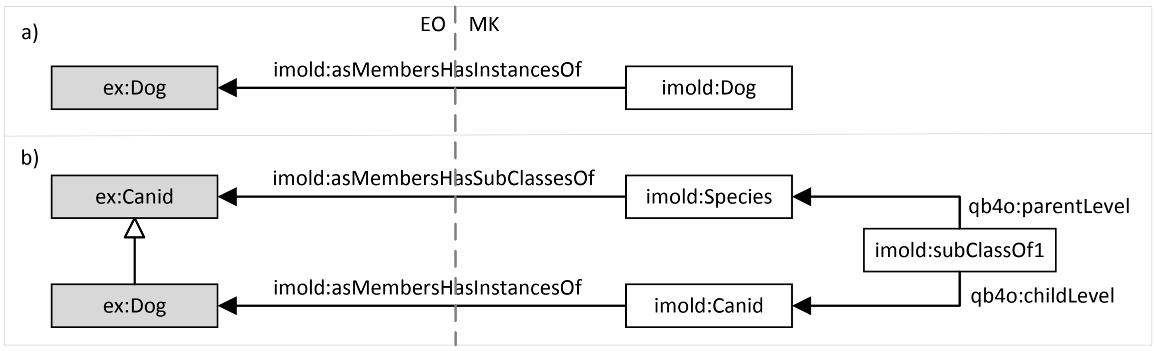


Fig. 6.10 Bottom-up strategy for detecting generalization-based patterns

6.6.3 Mixed patterns

Association- and generalization-based patterns explore different but coexisting aspects of an EO. This means that, starting from any class in the EO, both kinds of patterns can succeed in detecting (and mapping to the MK) roll-up relationships. However, because of their different mapping of classes into levels, a mix of association- and generalization-based patterns is feasible only under specific conditions. Firstly we will examine how associations can be inspected after applying generalization-based patterns, then how generalizations can be inspected after applying association-based patterns.

Consider a linear hierarchy H of n levels built by generalization-based patterns, where l_1 is the finest level and l_n is the coarsest one. Now let $a = (c, p, r)$ be a one-to-many association, where c is a class linked to a level of H; two cases arise:

- c is linked to l_1, so the members of l_1 correspond to the instances of all the classes in the EO (including c) that are linked to l_1 by means of the imold:asMembersHasInstancesOf property. Then the roll-up relationship $u = (l_c, p, l_r)$ can be added to H (clearly, since l_1 also has l_2 as a parent, u is added as a new branch of the hierarchy). However, since the members of l_1 also include the instances of other classes (i.e., the selected siblings of c), there is no certainty that a holds for the other classes as well, possibly resulting in a low support for u. An example of this situation is presented in Section 6.7 with reference to extending the domain of level Museum.

- c is linked to l_i, $i = 2, \ldots, n$, so the members of l_i correspond to the selected subclasses of all the classes in the EO that are linked to l_i by means of the imold:asMembersHasSubClassesOf property. Since in this case the instances of c have no correspondence in H, a cannot be mapped into a roll-up relationship.

Let us now examine the opposite situation. Let H be built using association-based patterns; then the members of l_i ($i = 1, \ldots, n$) correspond to the instances of the class in the EO that is linked to l_i by means of the imold:asMembersHasInstancesOf property. Now let c be a class linked to a level l_c of H, and $g = (d, p, c)$ be a generalization detected via a top-down strategy. Recalling how the first iteration of the top-down strategy is done (Section 6.6.2), selecting g causes a transformation of l_c: indeed, its members will no more correspond to the instances of c, but to the instances of the selected subclasses of c (i.e., d and its sibling possibly selected). As a result, the associations previously detected from (and to) c may not hold anymore and the current roll-up relationships involving l_c may be invalidated. Therefore, g can be safely selected only if every association that previously involved c is also valid for at least one of its selected subclasses. Finally, let $g' = (c, p, r)$ be a generalization detected via a bottom-up strategy. The transformation that l_c undergoes if g' is selected is just an extension of its domain, in which its previous members (i.e., the instances of c) are preserved and possibly joined by the instances of the selected siblings of c. Therefore, the requirement for the associations involving c to be still valid is always satisfied in this case.

Example 19 *Consider the situation depicted in Fig. 6.11, in which two different EOs are used:* ex1*, which models single animals, and* ex2*, which models the species taxonomy. The user first selects the* Animal *concept in* ex1*, which leads to creating level* Animal *in the MK. Then, by detecting pattern (1a) she finds roll-up relationship* belongsTo*, which leads her to discover the* Species *level and add it to the hierarchy. Within* ex2*, the user now detects pattern (2a) using the top-down strategy and selects concept* Mammal *with powertype* Class*, so level* Class *with member* Mammal *can be added to the hierarchy. Finally, the user detects pattern (2b) and selects concepts* Canid *and* Felid *with powertype* Family*, so the hierarchy is completed with level* Family*.*

6.6.4 User Experience

Implementing the acquisition phase of iMOLD poses one more important challenge, that is, how to provide a clean and simple user interface that hides the complexity of the detection process to let the user focus on the creation of hierarchies. We preliminarily note that the variety and veracity aspects of public ontologies make a complete automation of the acquisition process unfeasible, due to the inherent difficulties in reliably inferring the quality and relevance of the data explored. Additionally, there is a large degree of subjectivity in judging both the quality and the relevance of ontological

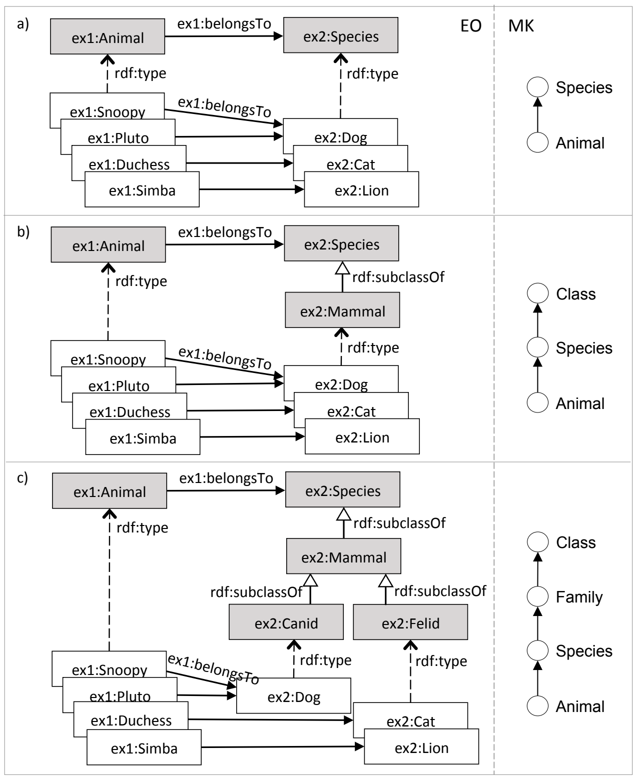


Fig. 6.11 Detecting mixed patterns in two linked EOs

data, so these evaluations may change from user to user. Finally, some concepts (e.g., powertypes) must be explicitly provided by the user. As a result, we claim that some degree of user supervision is mandatory to ensure the effectiveness of acquisition.

Acquisition builds on two basic operations:

- **Select a concept of interest**: with this operation, the user locates a class on a selected EO to identify a starting point for acquisition, i.e., to enable the subsequent detection of patterns. In practice, the user simply has to provide a search string and the ontology to query; then the system queries the SPARQL endpoint and retrieves a list of matching classes (together with their metadata). The matching classes are shown to the user, ranked by decreasing cardinalities (i.e., number of instances). As the user selects one class, her selection is stored in the MK and the UK.

- **Detect a pattern**: with this (iterative) operation, the user aims at creating a hierarchy involving the previously-selected class. Once the user has selected the specific pattern to be detected, the SPARQL endpoint of the EO is queried to find new roll-up relationship and update the MK accordingly. The system shows the list of candidate relationships, and as the user selects one (or more) of the results, her selections are saved in the UK as shown in Fig. 6.8.

To increase the effectiveness and the inclusiveness of the user experience, iMOLD proposes two alternative (and freely interchangeable) interaction approaches, both supporting the basic operations described above but tailored on users with different background and expertise. Both approaches are supported by the combined view shown in Fig. 6.12, which includes two panels: an *ontological panel* where the user has a picture of the classes and properties of the EO (represented with black dots and labeled arcs, respectively), and a *multidimensional panel* where the user sees how classes and properties translate to an aggregation hierarchy using a graphical notation inspired by the Dimensional Fact Model [63] (levels and roll-up relationships are represented with white dots and directed arcs, respectively).

- The *ontology-driven experience* is oriented to users who have good familiarity with ontologies and semantic web. Here the focus is set on the EO, a low-level view of the IO is provided, and a more intense user interaction is required. The user has a strict control over the acquisition phase; she can precisely specify the aggregation patterns to detect and finely tune the search parameters shown in Table 6.2 to refine the search based on her preferences. By clicking on a class in the ontological panel, a wizard for detecting a pattern is launched.

- The *OLAP-driven experience* is targeted to users who have good familiarity with the field of data warehousing and multidimensional modeling. Here the focus is set on the hierarchy being built, a high-level view of the IO is provided, and a lower degree of interaction is required. In this case pattern detection and search parameters are transparent to the user, who simply interacts by iteratively selecting, given a level l of interest on the multidimensional panel, one or more multidimensionally-inspired operations:

  – the search for a parent level of l, which triggers the detection of association-based patterns and that of generalization-based ones with top-down strategy;

  – the search for a child level of l, which triggers the detection of association-based patterns;

  – the extension of the domain of l, which triggers the detection of generalization-based patterns: with bottom-up strategy first, to find the parent levels of l (i.e., the superclasses of the class corresponding to l), and then with

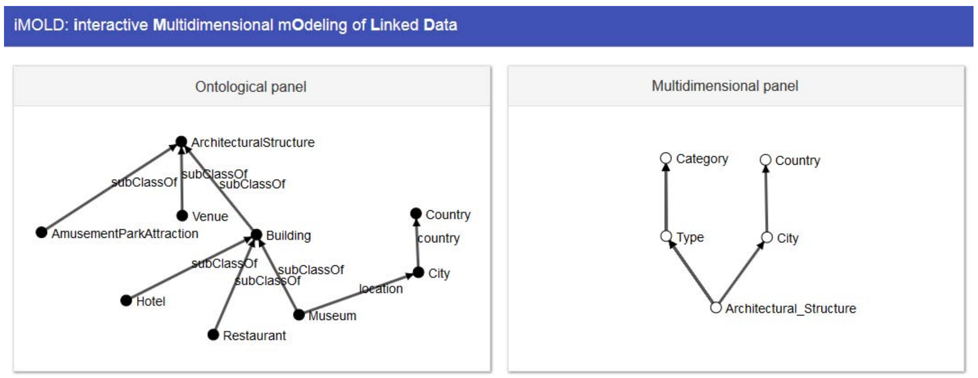


Fig. 6.12 The user interface of iMOLD, featuring the ontological (left) and the multidimensional (rigth) panel

top-down strategy to find new members for *l* and its parent levels (i.e., the subclasses of the superclasses previously found).

For every operation, the system uses for all search parameters the default values shown in Table 6.2.

6.6.5 Collaboration and Reuse

One of the features of iMOLD is that of enabling users to collaborate by sharing and reusing their findings. This can be achieved thanks to the MK and the UK. The MK stores all the levels and roll-up relationships ever selected by all users, thus avoiding further accesses to the SPARQL endpoint for a user interested in a concept previously explored by others. The UK shows which levels and roll-up relationships have been selected by users, thus enabling each user to share her point of view on analyses with the others. Both activities take place during the assessment phase (see Section 6.4), and could employ advanced techniques for ontology exploration such as [73].

During the acquisition phase, collaboration is achieved in two distinct ways. On the one hand, the ranking of the roll-up relationships discovered by Algorithm 1 is completed by showing, for each relationship, with which frequency it has been selected by other users, i.e., its "popularity". On the other hand, the user experience described in Section 6.6.4 is further enhanced by including a specific *suggest* operator to be used, while exploring a given concept, to discover what previous users have done while exploring that same concept. By choosing this operator, the user can see the list of the roll-up relationships ever selected by other users.

Computing the popularity and implementing the *suggest* operator both require to measure the frequency of a roll-up relationship a, which entails counting the instances of property imold:RollupPreference related to the qb4o:HierarchyStep that corresponds to the property and the two classes involved in a (see Figure 6.3). For instance, the SPARQL query that ranks the roll-up relationships with child ?childLevel stored in the MK with according to their frequency can be written as follows:

```
SELECT ?parentLevel ?property (COUNT(?pref) as ?frequency)
WHERE
{    ?pref a imold:RollupPreference ;
          imold:isRollupPreferenceOf ?rollupRel .
     ?rollupRel a qb4o:HierarchyStep ;
          qb4o:childLevel ?childLevel ;
          qb4o:parentLevel ?parentLevel ;
          imold:correspondsTo ?property .
     ?childLevel a qb4o:LevelProperty .
     ?parentLevel a qb4o:LevelProperty
}
GROUP BY ?parentLevel ?property
ORDER BY ?frequency DESC
```

6.7 Case Study and User Evaluation

To demonstrate the potential of our approach, in the first part of this section we present a case study in which a fictional user builds a hierarchy, one step at a time, by exploring a small portion of DBpedia. In the second part, we propose a more quantitative evaluation by discussing the results of some tests made with real users.

To evaluate the iMOLD approach we chose to focus on the so-called *cross-domain* ontologies, as their chaotic status makes it harder for both humans and machines to efficiently extract structured and coherent knowledge. While the generic theme of this kind of ontologies opens to a broad set of examples in different domains, some degree of uncertainty and incorrectness appears in the data. Indeed, differently from small and confined ontologies that are typically developed in-house and provide a reliable representation of the area of interest, big and cross-domain ontologies are more inclined to errors and imprecisions, especially if they are built through a collaborative effort across the web. In particular, we focus the case study on DBpedia, one of the best-known public ontologies available on the web, which is also compliant with the linked data principles and covers a wide range of domains.

The prototype we built is implemented as a web application and fully supports the user experience described in Section 6.6.4. The back-end functionalities are implemented in Java; in particular, we extensively rely on the Jena Library, as it provides solid APIs for the communication with remote SPARQL endpoints and for in-memory manipulation of a local ontology. As to the IO, we currently store it within a simple RDF file; however, the migration to a triplestore is planned in future evolutions of the prototype. Finally, we used Javascript to implement the user interface and we adopted the D3 library for the graphical visualization of the IO.

6.7.1 Case Study

As a case study we consider a social BI scenario [43] in which a fictional user user is collecting and analyzing reviews about museums and is interested in discovering a hierarchy rooted in museums. The most obvious candidate as a starting concept is class dbo:Museum, which currently hosts 1884 instances. A few simple queries on dbo:Museum show that the distinct properties that make use of its instances as either domain or range in the triple are 387 and 156, respectively, for a total of 125086 instances. Although not impressive, this numbers prove that a manual approach to identify the properties that map to roll-up relationships would be at least wearying, if not unfeasible.

Table 6.3 Detecting the parents of Museum

p (property/semantics)	r (range/parent)	$supp(p)$	$supp(r)$
dbp:imagesize	xsd:integer	64.5%	—
dbp:established	xsd:integer	50.7%	—
dbo:location	dbo:City	40.9%	1.4%
dbo:location	dbo:Country	40.6%	4.8%
dbo:location	yago:MemberStatesOfTheUnitedNations	32.6%	36.6%
dbo:location	dbo:Settlement	32.5%	36.6%
dbo:location	yago:MemberStatesOfNATO	18.6%	85.2%
dbo:location	yago:CountriesBorderingTheAtlanticOcean	17.3%	38.6%
dbp:location	yago:English-speakingCountriesAndTerritories	17.0%	21.8%
dbp:visitors	xsd:integer	14.9%	—

To begin the exploration of the EO following the ontology-driven experience, we assume that the user clicks on the dbo:Museum class on the ontological panel and activates the detection of association-based patterns in the outbound direction to find the parents of level Museum. The results of the application of association-based patterns are shown in Table 6.3, where each row corresponds to a many-to-one association $a = (d, p, r)$ with domain $d =$ dbo:Museum and range r, i.e., to a roll-up relationship with semantics p from Museum to r; $supp(p)$ is the percentage of instances of dbo:Museum for which p is present, while $supp(r)$ is the percentage of instances of r for which p is present.

Among the associations where r is a class (i.e., those detected with patterns (1a) and (1b)), the only relevant property is dbo:location, which provides the geographical position of each museum. Interestingly, this property links to instances that belong to different classes (see rows 3–9 of Table 6.3). This is due to multiple classification, which is widely used in DBpedia. For example, objects like dbr:Italy and dbr:Spain are not just instances of dbo:Country, but also of yago:MemberStatesOfTheUnitedNations and yago:MemberStatesOfNATO. The iMOLD interface helps the user in this case by orienting her towards the classes with most references (i.e., higher support). As to the associations where r is a literal (i.e., those detected with pattern (1c)), property dbp:established (which indicates when each museum was opened) could be interesting for analysis. Conversely, properties dbp:imagesize and dbp:visitors are not likely to be selected by the user, as the first one is clearly not relevant while the second one has a very low support.

For the sake of the example, we assume that the user selects association (dbo:Museum, dbo:location, dbo:City). The following detection of association-based patterns from dbo:City leads to finding new associations, among which the most relevant is (dbo:City, dbo:location, dbo:Country). Assuming that this one is selected, a 3-level linear hierarchy is eventually built: Museum, City and Country.

Table 6.4 Extending the domain of level Museum

r (superclass/new name of level)	d' (sibling class/member for new level)	$card(d')$
dbo:Building	dbo:Castle	457
	dbo:HistoricBuilding	7024
	dbo:Hospital	1855
	dbo:Hotel	1200
	dbo:Library	889
	dbo:Prison	620
	dbo:ReligiousBuilding	3731
	dbo:Restaurant	725
	dbo:ShoppingMall	2499
	dbo:Skyscraper	3

Back to our social BI scenario, we now assume that the user wants to broaden its scope from museums only to other places of interests that are subject to reviews, such as hotels and restaurants. In the context of the OLAP-driven experience, she can click on the Museum level of the multidimensional panel and activate the operation to extend the domain of that level. As a consequence, the generalizations $g = (d, p, r)$ with domain $d =$ dbo:Museum and property $p =$ rdfs:subClassOf are explored to find the superclasses r of dbo:Museum, then for each r its other subclasses d' are explored to find candidate new members for level Museum. The results are shown in Table 6.4.

Class dbo:Museum has only one superclass, dbo:Building. The user can now select the siblings that relate the most to her area of interest (e.g., dbo:Hotel and dbo:Restaurant) and provide the name of the powertype (in this case, Type could be an option). As a result, the hierarchy is changed by (i) renaming level Museum into Building, whose members are now the union of the instances of dbo:Museum, dbo:Hotel, and dbo:Restaurant; and (ii) adding level Type as a parent of Building, with members Museum, Hotel, and Restaurant. As explained in Section 6.6.3, the roll-up relationship from Museum to City is not affected by the transformation of Museum into Building, but its support may be changed; hence, the existence of associations (dbo:Hotel,dbo:location,dbo:City) and (dbo:Restaurant,dbo:location,dbo:City) should be checked.

Table 6.5 Extending the domain of level Type

r (superclass/new name of level)	d' (sibling class/member for new level)	$card(d')$
dbo:ArchitecturalStructure	dbo:AmusementParkAttraction	499
	dbo:MilitaryStructure	372
	dbo:Tunnel	70
	dbo:Venue	657

The domain extension operation can be iteratively activated to explore a larger part of the EO. For instance, extending level Type leads to searching for the siblings of dbo:Building; the results are shown in Table 6.5.

If dbo:AmusementParkAttraction and dbo:Venue are selected as relevant concepts and the new powertype is named Category, the hierarchy is further extended by (i) renaming Building into ArchitecturalStructure and adding new members (i.e., the instances of dbo:AmusementParkAttraction and dbo:Venue); and (ii) adding level Category as a parent of Type, with members Building, AmusementParkAttraction, and Venue. Level Type does not incur in any change; however, further user explorations will be required to add the specializations of dbo:AmusementParkAttraction and dbo:Venue as new members of this level.

The final shape of the hierarchy is the one shown in Fig. 6.12.

6.7.2 User Evaluation

To give a quantitative assessment of the benefits of our approach and compare the effectiveness of the ontology-driven and OLAP-driven scenarios, we conducted a set of tests with a group of 21 real users, mainly PhD and master students with basic or advanced knowledge of ontologies and multidimensional modeling. The goal of the tests was to evaluate the execution of the same task (i.e., build a hierarchy starting

Table 6.6 Average results of user test for the three tasks (times are expressed in minutes)

Figure	SPARQL-based task	Ontology-driven task	OLAP-driven task
Total time	71.6	26.7	21.8
- Exploration time	-	19.6	14.7
- System time	-	3.0	3.3
- Browsing time	-	4.1	3.8
Num. interactions	11	27	25
Score (1..5)	2.3	3.4	3.1

from a concept selected from the DBpedia ontology) according to the two interaction scenarios of iMOLD.

To prepare the tests, three concepts in different application domains were chosen (namely banks, museums, and screenwriters), and for each concept a task was stated using either the multidimensional or the ontological terminology (e.g., *extend the domain of level* Museum *by building a multi-level hierarchy to include also hotels and other types of architectural constructions, then build a hierarchy to aggregate these constructions according to the currencies of the countries they are located in*). Each user was asked to read an instruction sheet explaining the iMOLD goals, how to operate the interface in both scenarios, and the five aggregation patterns; then, (s)he was asked to execute two tasks, one for each interaction scenario, on two randomly-chosen domains. To determine a testing baseline, we also asked to execute an additional task using plain SPARQL to the users who had some knowledge of this language; in this case, the users were relieved from the construction of the hierarchies in the IO and only had to manually draw the inferred hierarchy.

Table 6.6 shows the average results obtained for each task. By monitoring the user activities on the prototype, we split the time taken to complete each task (*total time*) into three blocks: *exploration time* (i.e., the time taken by the user to check the status of the hierarchy and choose the next operator to be applied), *system time* (i.e., the time taken by the system to detect one or more patterns and provide the list of candidate relationships), and *browsing time* (i.e., the time taken by the user to browse these relationships and select one or more of them). The table also shows the average *number of interactions* made by each user, referring to the number of SPARQL queries, of pattern searches, and of multidimensional operations, respectively for SPARQL-based, ontology-driven, and OLAP-driven tasks. Finally, the hierarchies built by each user have been manually evaluated for correctness and completeness and scored on a scale from 1 (worst result) to 5 (best result); row *score* shows the average results.

In SPARQL-based tasks, users have mostly failed in achieving an acceptable result despite spending a considerable amount of time. The main difficulties encountered are the design of the queries to apply association-based patterns (an activity that took away almost half of the total time) and the interpretation of generalization-based patterns: although most users built a correct hierarchy with two levels, no one was able to build a correct hierarchy with three levels. This is due to the fact that mere query results obviously do not support users in finding the correct interpretation key.

Conversely, the iMOLD prototype enabled users to adequately complete the exercise in a short time. Indeed, manually writing the queries requires a lot more time than simply launching the queries automatically written by iMOLD; so, in the same time iMOLD users employed to acquire a comprehensive picture of the ontology, SPARQL users could only catch a very partial glimpse of it. Users spent less time on OLAP-driven tasks than on ontology-driven ones (22 minutes against 27), showing that the OLAP-driven experience succeeds in reducing the amount of work for the user. Interestingly, however, most users declared to feel more comfortable with the ontology-driven experience, especially when mixing patterns was necessary: in fact, the low-level view adopted for the former experience reduces the potential misinterpretations of the multidimensional mapping of the patterns, thus enabling users to better identify the correct relationships. This is reflected in the results, where the average score for ontology-driven tasks is slightly better than that of the OLAP-driven ones (3.4 against 3.1); one more clue in this direction comes from the fact that the best results in the OLAP-driven task came from the users who had already carried out the ontology-driven one (3.3 against 2.8).

To measure the efficiency of iMOLD, we also measured the average system and decision times for the single patterns. The results show that the slower pattern to be executed is 1b in the inbound direction (15 seconds on average). This is expected, because many-to-one relationships in EOs tend to be expressed from subjects to objects rather than vice versa (i.e., one-to-many relationships as the one in Fig. 6.6 are less used); as a result, the number of triples to be considered by the SPARQL query in the inbound direction is sensibly higher, causing higher system times. On the other hand, pattern 1b in the outbound direction turns out to be the one with the highest decision time (38 seconds on average, vs. a few seconds for generalization-based patterns). This is also expected, as this pattern is the one that statistically returns the highest number of candidate relationships.

6.8 Conclusions

In this chapter we have presented iMOLD, an approach to build aggregation hierarchies at exploration time to integrate the corporate cubes with situational data. We assume these situational data come in the form of linked data in RDF format, whose lack of concrete schema presents important challenges. Specifically, we have identified five different patterns that go beyond the classical functional dependencies approach for multidimensional design, and also consider the possibility of defining aggregations hierarchies based on taxonomies and combinations of both.

Though iMOLD is a significant step towards integrating linked data into corporate cubes, some relevant aspects still need to be considered. First of all, there is a possibility that some hidden FDs are present in linked data; for instance, with reference to our working example, each animal may be associated to its species and family, but an explicit connection between species and families may be missing in the source ontology. In this case, using algorithms for instance-based discovery of approximate FDs (e.g., TANE [78]) would enable hierarchies to be more accurately reconstructed. One more issue is related to automatic recognition and management of cycles in source data, which can give rise either to shared hierarchies (e.g., a building is located in a country and was designed by an architect who was born in a country, but the two countries may be different) or to convergences (e.g., a museum is located in a country and is managed by an institution of the same country). Finally, the approach could be extended to detect not only hierarchies to enrich corporate cubes, but also new cubes from linked data.

Chapter 7

Profiling hidden schemata on schemaless data

In this chapter we propose a technique, called schema profiling, to capture the hidden rules explaining the use of different schemata within a collection in document-oriented databases; we express these rules in the form of a decision tree (schema profile). Consistently with the requirements we elicited from real users, we aim at creating explicative, precise, and concise schema profiles. The algorithm we adopt to this end is inspired by the well-known C4.5 classification algorithm and builds on two original features: the coupling of value-based and schema-based conditions within schema profiles, and the introduction of a novel measure of entropy to assess the quality of a schema profile. A set of experimental tests made on both synthetic and real datasets demonstrates the effectiveness and efficiency of our approach.

7.1 Introduction

One of the main advantages of schemaless DBMSs is the flexibility the designers (and the implementers) have in changing data structures. Such flexibility reduces the time needed to release new versions of the applications and makes it easier to add new information, to handle emerging requirements, and to recover from errors occurred during schema design. These features are particularly relevant for development teams adopting agile methodologies, which are inherently iterative and evolutive. Most frequently, schema changes are due to either (i) evolutions in time of the application logic that manages data, or (ii) new user requirements, or (iii) the need for uniformly handling instances with specific features. Typical changes consist in adding or dropping some attributes, changing their names or types, and "restyling" the structure of the instances for example by moving a set of attributes into a sub-structure.

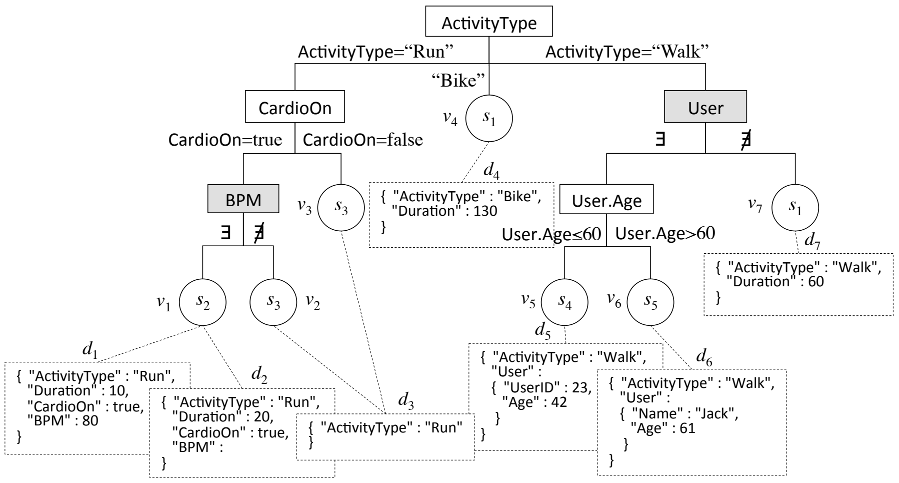
Unfortunately, this increased flexibility and the absence of a unique, well-defined schema turn to disadvantages when moving from operational applications to analytical applications and business intelligence. Business intelligence analyses typically involve large sets of data, so a single analysis often involves instances with different —and possibly conflicting— schemata; in turn, this requires some extra effort to understand the rules that drove the use of alternative schemata, and an integration activity to identify a common schema to be adopted for analysis. These task are even harder when no detailed documentation is available, because the analyst has to search for the necessary knowledge either in the code that manages data or in the data themselves.

In this chapter we propose a technique to capture the hidden rules explaining the use of different schemata within a collection in document-oriented databases. We call this activity *schema profiling.* Schema profiling technique is beneficial in different contexts:

- when trying to decode the behavior of an undocumented application that manages a document-base;
- when carrying out a data quality project on schemaless data;
- when adopting a schema-on-read approach to query a document-oriented database [109, 34];
- when designing a data warehouse on top of a schemaless data source, for instance a corporate data lake.

Identifying the rules of schema usage is much like building a descriptive model in a classification problem. A *classifier* is a model that predicts to which of a set of classes a new observation belongs, based on a training dataset containing observations whose class membership is known. Besides for predicting, classifiers are also used to *describe* the rules for assigning a class to an observation based on the other observation features —which corresponds to our goal if we consider the schema of a document as its class. So we can rely on the existing literature on classification to build schema profiles; in particular, based on the requirements we collected from potential users of our approach, among the different types of classifiers we consider decision trees.

Straightly reusing traditional decision trees for schema profiling would mean classifying documents based on the *values* of their attributes only. However, this would often lead to trees where a single rule serves different classes (i.e., different schemata are



```
{ "ActivityType" : "Bike",
  "Duration" : 130
}
```

```
{ "ActivityType" : "Walk",
  "Duration" : 60
}
```

```
{ "ActivityType" : "Run",
  "Duration" : 10,
  "CardioOn" : true,
  "BPM" : 80
}
```

```
{ "ActivityType" : "Run",
  "Duration" : 20,
  "CardioOn" : true,
  "BPM" :
}
```

```
{ "ActivityType" : "Run"
}
```

```
{ "ActivityType" : "Walk",
  "User" :
  { "UserID" : 23,
    "Age" : 42
  }
}
```

```
{ "ActivityType" : "Walk",
  "User" :
  { "Name" : "Jack",
    "Age" : 61
  }
}
```

Fig. 7.1 A schema profile in the physical fitness domain

explained by the same rule), which would be give an imprecise information. To address this issues, in our technique documents are also classified using *schema-based conditions* related to the presence or absence of attributes. To better understand this point, consider for example Figure 7.1, showing a portion of a decision tree (which we call *schema profile*) built in the domain of physical fitness to profile a collection of documents generated by training machines or by their users through mobile applications. Each internal node in the tree is associated to a document attribute a and can express either a value-based condition (white box; each outgoing arc is related to one or more values of a, e.g., User.Age < 60) or a schema-based condition (grey box; the two outgoing arcs represent the presence or absence of a in the document, e.g., $\exists$BPM). Each path in the tree models a *rule*; it leads to a leaf (represented as a circle) that corresponds to a schema found for the documents that meet all the conditions expressed along that path (document examples are shown in dashed boxes). So, for instance, schema s_2 is used in all the documents for which ActivityType = “Run”, CardioOn = true, and field BPM is present (independently of its value, or even if a value is missing).

Another drawback of traditional decision trees is that they often give several rules for the same class. While this may be correct for some specific collections (e.g., schema s_1 in Figure 7.1 appears in two leaves, i.e., it is explained by two different rules), in general we wish to keep the number of rules to a minimum aimed at given users a more concise picture of schema usage. This is achieved in our approach by adopting a novel measure of entropy to be coupled with the one typically used to characterize and build decision trees.

The outline of this chapter is as follows.

- In Section 7.2 we discuss the related literature.
- In Section 7.3 we analyze the desiderata of real users that inspired our approach.
- In Section 7.4 we provide the necessary formal background.
- In Section 7.5 we introduce a novel measure of entropy, called *schema entropy*, which contributes to assess the quality of a schema profile.
- In Section 7.6 we describe an algorithm that implements a divisive approach to deliver precise, concise, and explicative schema profiles by mixing value-based and schema-based conditions to better capture the rules explaining the use of different schemata within a collection.
- In Section 7.7 we present a set of experimental results on both synthetic and real datasets.
- In Section 7.8 we draw the conclusions.

7.2 Related Literature

The task of schema discovery is not new to the academic world. Early works focused on finding ways to describe semi-structured data retrieved from web pages and were mainly based on the Object Exchange Model (OEM). The general idea was to automatically derive a concise [128, 172, 118] or approximate [127, 174] labeled graph model to allow efficient querying of data. In the latter group, approximation was achieved by applying clustering algorithms on the data, so that a different model could be defined for each syntactically-similar group. As XML became a standard for data exchange on the web, researchers had to deal with the widespread lack of DTDs and XSDs for XML documents. Works such as [54, 70, 12] focused on extracting the regular expressions that described the contents of the elements in a set of XML documents. Most commonly, the derived schema was necessary to help software tools in processing XML documents for various tasks (e.g., search, translation, validation) or to integrate data through schema matching.

With the replacement of XML with JSON, similar issues are now being addressed on this new standard, with the goal to capture and represent the intrinsic variety of schemata within a collection of JSON objects. [83] proposes to build a unique schema for a set of JSON objects returned by an API service; its goal is to eventually create a unified model of the schemata derived from a set of API services. The schema representation is based on a class model, where each nested object is represented by a distinct class and whose properties are the union of the simple attributes within the object. [97] proposes and evaluates two different graph structures that model the union of all the attributes in a collection of JSON objects, aimed at measuring the heterogeneity of a collection and at detecting attributes with low support for data quality investigations. [153] proposes a reverse-engineering process to derive a versioned schema model for a collection of JSON objects. Here, the authors introduce the concept of *versions*: instead of considering the mere union of the attributes within a nested object (here called *entity*), a new version of the same entity is created for every intensional variation detected in the JSON objects. This kind of schema modeling allows both to trace back the original schemata of the single JSON objects (so as to perform validity checks on new objects) and to compute the union of the attributes within the single entities. Oddly, the frequency of each version is not considered in the approach. In [173] a clustering technique is first applied to the JSON objects of a collection to identify the groups of similar schemata corresponding to specific business objects. Afterwards, the schemata of each group are summarized into a *skeleton*, i.e., a rooted tree containing the smallest set of core attributes according to a frequency-based formula. The proposed skeleton is a valid instrument to facilitate the recognition of the underlying schemata by the user and to significantly reduce the cost of computations on the summarized objects without compromising the accuracy of the results. However, the assumption that schema-similarity is the metric that characterizes the different object types is potentially misleading; indeed, this is not always true in our experience.

Overall, the ultimate goal of the works mentioned so far is to provide either a synthetic [173, 97] or a comprehensive [83, 153] way to describe the intensional aspects of the documents, not only for an easier recognition of the schemata, but also to enable automated activities such as querying, integration, and validation. However, to the best of our knowledge, no approach has ever considered the extensional point of view to explain the different usages of schemata.

With the increasing diffusion of document-based stores, several tools are currently known to perform schema detection on the available DBMSs. Starting from MongoDB[1], a handful of free tools to help users in the analysis of the hidden schema have been designed by third-party developers, such as variety.js[2], schema.js[3] (both MongoDB

[1]https://www.mongodb.org/
[2]https://github.com/variety/variety
[3]http://skratchdot.com/projects/mongodb-schema/

tools) and mongodb-schema[4] (a module for NodeJS). In ElasticSearch[5], the embedded functionality of *dynamic mapping* automatically infers the schema to enable search capabilities on the documents. In Couchbase[6], the Spark connector can perform automatic schema inference that will enable Spark SQL operations. Also Apache Drill[7], which supports a variety of NoSQL databases, dynamically infers the schema at query time —although it is used only internally and cannot be exported by the user. In the end, by applying different strategies for conflicting attributes and datatypes, each of these tools derives a unique schema which collects the union of the attributes found in the documents; at best, some statistics are collected for each attribute, such as the datatypes used and its support. Finally, it is worth mentioning that the JSON-schema initiative[8] has risen with the idea to provide standard specifications to describe JSON schemata; however, its adoption is still quite limited and restricted to validation software.

In parallel to schema discovery, plenty of research focused on the issues related to schema matching. [11] provides a comprehensive summary of the different techniques envisioned for generic schema matching, which ranges from the relational world to ontologies and XML documents. Other papers have also investigated the applications of mining techniques specifically to XML documents: for instance, [122] and [104] provide clustering algorithms to be applied to XML schemata, while [66] provides an overview of the different similarity measures that can be used to cluster XML documents.

An interesting branch of schema matching is the one that exploits contextual information in the data. For example, [16] proposes an enhancement of schema matching algorithms by extending matches (which refer to table attributes) with selection conditions, which enables to identify the cases in which the match is actually valid. The principle of analyzing instance values is used also by [130] and [32], which apply machine learning techniques to schema matching. Interestingly, while these works use contextual information to identify schema matching conditions, we use contextual information for the opposite reason, i.e., to justify schema heterogeneity.

Our approach relies on decision trees to describe the features that rule the schema usage. The huge amount of data stored by NoSQL databases could possibly undermine the efficiency of the algorithms for decision tree learning. Early works in this direction

[4]https://www.npmjs.com/package/mongodb-schema
[5]https://www.elastic.co/guide/en/elasticsearch/reference/2.3/mapping.html
[6]http://developer.couchbase.com/documentation/server/4.0/connectors/spark-1.0/spark-sql.html
[7]https://drill.apache.org/
[8]http://json-schema.org/

propose techniques to incrementally build a decision tree: the goal is to eventually provide the same tree that a static algorithm would have provided, but in an incremental manner, which allows for early results to be displayed and for a more careful selection of the training instances [167], [26], [89]. Approximation of the final result has been introduced first by scalable approaches, which rely on efficient bootstrapping [56] or subsampling [33] of the training data. A whole new set of challenges was then brought by big data and its 4Vs (variety, volume, velocity and veracity); [177, 98] are recent surveys that discuss the latest problems and advancements in this area. Finally, [27] is an interesting work that proposes an implementation of the original C4.5 algorithm [142] on the Hadoop platform. Noticeably, the purity measures and stop criterion we propose in this paper are general enough to be applied to all decision tree algorithms independently by their specific implementations.

7.3 Requirements for Schema Profiling

The first stage of our work has been devoted to capture user requirements for schema profiling. We selected a group of users (typically data designers or programmers of the data management application) and interviewed them to understand their goals, the type of information they require, and the visualization format they prefer for the results. We also asked them to "manually" describe their own documents and explain the reasons that determine the use of each specific schema so as to better understand the relevant features of a description, its imprecisions and limits, and the main difficulties users may encounter in its construction.

The main hints we obtained are summarized below:

- Users need an easy-to-read, graphical, and compact schematization of schema usage. When asked to give their own description of documents, they delivered a tree-based representation, each node representing a condition on either instances (e.g., Date $\leq$ 2016) or schemata (e.g., Date is present).

- Value-based conditions are more relevant than schema-based ones. This is because schema-based conditions (i.e., presence/absence of some given attributes in a document) obviously describe the difference between schemata but do not provide any clues about the the reason of this difference.

- Users tend to describe the conditions that, according to their domain knowledge, are most significant first. The importance of a condition is subjective to some extent, but it is typically related to the usage of different applications/functionalities rather than to the actual inter-schema similarity. Indeed, we found documents with very similar schemata that are used in quite different contexts; this suggested to discard solutions, like [173], that exploit clustering techniques or skeleton concepts to reduce the total number of schemata and provide an initial summary of schemata.

- Even for very experienced users, it is quite difficult to provide a precise and complete description of the conditions ruling the schema usage. In most cases either the given description was incomplete (i.e., users were not able to provide all the rules for obtaining a precise characterization of each single schema) or even wrong (i.e., the rules provided did not correctly split the documents according to their schemata).

Based on these considerations, we claim that an automated approach to schema profiling is potentially beneficial and should have the following features:

- the result must take the form of a decision tree (called *schema profile* from now on) to provide a comprehensible description;
- the schema profile should be *explicative*, i.e., it should give priority to value-based conditions;
- the schema profile should be *precise*, i.e., it must accurately characterize each single schema;
- the schema profile should be *concise*, i.e., it must provide a small set of rules.

7.4 Formal Background

The central concept of a document-oriented database is the notion of a *document*, which encapsulates and encodes data in some standard format. Formats in use include XML, YAML, JSON, and BSON; the most widely adopted format is currently JSON, which we will use as a reference in this work.

The JSON grammar specifies that an *object* is formed by a set of *name/value* pairs, commonly referred to as *elements*. A *value* can be either a primitive value (i.e, a number, a string, or a Boolean), an array of values, an object, or even `null`. Names cannot be repeated within the same object, but they can be repeated at different nesting levels. An important feature of JSON is that arrays have no constraint on the type of their values, i.e., an array can simultaneously contain numbers, strings, other arrays, as well as objects with different internal structures.

```
{ "ActivityType" : "Run" ,
  "Duration" : 108 ,
  "CardioOn" : true,
  "Notes" : null ,
  "Details" :
  { "MusicTracks" : [ 4 , 8 ] ,
    "Comments":
    [ { "UserID" : 15 ,
        "Comment": "Well done!"
      } ,
      { "UserID" : 16,
        "Vote" : "6/10"
      } ] ,
  } ,
  "User" :
  { "UserID" : 23 ,
    "Name" : "Jack" ,
    "Age" : 42 ,
    "FacebookID" : "jack42"
  }
}
```

(a)

```
{ ...
  "Comments" :
  { "type" : "array" ,
    "items" :
    [ { "type" : "object" ,
        "properties" :
        { "UserID" : { "type" : "number" } ,
          "Comment" : { "type" : "string" }
        }
      } ,
      { "type" : "object" ,
        "properties" :
        { "UserID" : { "type" : "number" } ,
          "Vote" : { "type" : "string" }
        } ]
  } ,
  ...
}
```

(b)

Path	*Type*
ActivityType	primitive
Duration	primitive
CardioOn	primitive
Notes	primitive
Details	object
Details.MusicTracks	array
Details.Comments	array
User	object
User.UserID	primitive
User.Name	primitive
User.Sex	primitive
User.FacebookID	primitive

(c)

Fig. 7.2 A JSON document representing a training session (a), a portion of its JSON schema (b), and its r-schema (c)

Definition 14 (Document and Collection) *A* document *d is a JSON object. A* collection *D is a set of documents.*

Figure 7.2.a shows a document representing a training session containing an array of objects with different schemata (element Comments).

The JSON schema initiative provides the specifications to define the schema of a document; however, as highlighted in the example in Figure 7.2, the resulting schemata are quite verbose and they provide a complex representation of arrays. Indeed, the schema of an array is defined as the ordered list of the schemata of its values; in other words, two arrays share the same schema only if they contain the same number of values and these values share the same schemata in the same order. For the purpose of this study we adopt a more concise representation for the schema of a document, called *reduced schema*, which does not enter into the content of arrays, but simply denotes the presence of an array structure.

Definition 15 (R-Schema of a Document) *Given document d, the reduced schema (briefly,* r-schema*) of d, denoted $rs(d)$, is a set of* attributes*, each corresponding to one element in d. Attribute $a \in rs(d)$ is identified by a* pathname*, $path(a)$, and by a type, $type(a) \in \{\text{primitive}, \text{object}, \text{array}\}$. While $path(a)$ is a string in dot notation reproducing the path of the element corresponding to a in d, $type(a)$ is the type of that element (type* primitive *generalizes numbers, strings, Booleans, and nulls).*

Note that, although r-schemata are defined as sets of attributes, their pathnames code the document structure (short of the internal structure of arrays).

Example 20 *Figure 7.2 shows a sample document, its JSON schema (as per the specifications of the JSON schema initiative), and its r-schema. Note how, in the r-schema, the complexity and heterogeneity of array* Comments *is hidden in attribute* Details.Comments *with generic type* array.

To put together the relevant information coded by different r-schemata, we need to define a sort of global schema for the documents within a collection. This is done as follows.

Definition 16 (R-Schema of a Collection) *Given collection D, we denote with $S(D)$ the set of distinct r-schemata of the documents in D (where two attributes in the r-schemata of two documents are considered equal if they have the same pathname and the same type). The* r-schema *of D is defined as*

$$rs(D) = \bigcup_{s \in S(D)} s$$

Given $s \in S(D)$, with denote with $|D|_s$ the number of documents in D with r-schema s.

Intuitively, $rs(D)$ includes all the distinct attributes that appear in the r-schemata of the documents in D. Since our goal is to explain the usage of syntactically different documents, no approach to determine attribute equivalence (e.g., [173]) is adopted.

The last concept we need to introduce are *schema profiles.*

Definition 17 (Schema Profile) *Let D be a collection. A* schema profile *for D is a directed tree T where each internal node (including the root) corresponds to some attribute $a \in rs(D)$ and can be either* value-based *or* schema-based*:*

- *a schema-based node has exactly two outgoing arcs, labelled as $\exists$ and $\nexists$ respectively;*
- *a value-based node has two or more outgoing arcs, each labelled with a condition expressed over the domain of a.*

Value-based nodes can only correspond to attributes of type primitive. *Given node v, we denote with $D_v \subseteq D$ the set of documents that meet all the conditions expressed by the nodes in the path from the root to v, with $S(D_v) \subseteq S(D)$ the set of distinct r-schemata of the documents in D_v, and with $|D_v|_s$ the number of documents with r-schema $s \in S(D_v)$ belonging to D_v.*

Intuitively, each internal node in a schema profile models a condition (on a value-based column of the dataset if the node is value-based, on a schema-based column if it is schema-based), and each path models a *rule* that includes a set of conditions for selecting one or more r-schemata. We also remark that Definition 17 can accommodate both binary and n-ary trees; in Section 7.5 we will show that binary trees are preferable to n-ary trees in our context.

Definition 17 does not explain how to treat missing attributes and missing values. The approach we adopt to deal with this issue is consistent with the one used by the decision tree algorithm we chose, i.e., C4.5 [142] in its Weka implementation. In particular, we assume that a document d such that $a \notin rs(d)$ (missing attribute) or $d.a = \mathsf{null}$ (missing value) always meets all the conditions expressed by a value-based node corresponding to a. As a consequence, in presence of a value-based node corresponding to a, such document d is considered to belong to two or more leaves; in other words, the sets D_{v_j} for $j = 1, \ldots, m$ are not necessarily disjoint. Algorithm C4.5 also adopts a weighting mechanism in these cases; specifically, when computing leaf cardinalities (namely, $|D_{v_j}|$ and $|D_{v_j}|_s$ in Section 7.5), a document d such that $d \in D_{v'} \cap D_{v''}$ is weighted depending on how the other documents are distributed in v' and v'' [142].

Example 21 *Figure 7.1 shows an n-ary schema profile with two schema-based nodes (*User *and* BPM*) and three value-based nodes (*ActivityType*,* CardioOn*, and* User.Age*). In this case, it is $D = \{d_1, \ldots, d_7\}$ and $S(D) = \{s_1, \ldots, s_5\}$. The schema profile has leaves $v_1, \ldots, v_7$, with $D_{v_1} = \{d_1, d_2\}$. Note that document d_3 belongs to both v_2 and v_3, since attribute* CardioOn *is missing.*

7.5 Evaluating Schema Profiles

In Section 7.3 we claimed that, according to the users' feedback, a good-quality schema profile should be explicative, precise, and concise. In the following subsections we discuss how these three features can be quantitatively evaluated.

7.5.1 Explicativeness

Divisive approaches build decision trees by first creating a single node that includes all the observations, then by iteratively splitting each node into two or more nodes that include subsets of observations. The split point is chosen in a greedy fashion by trying to maximize the quality of the classification. In our context, splits come in different flavors.

Definition 18 (Split) *Let D be a collection and $a \in rs(D)$ be an attribute in the r-schema of D. Let T be a schema profile for D and v be one of its leaves. A* split *of v* on *a, denoted $\mu_a(v)$, transforms T into a schema profile T' where two or more new leaves are added to T as children of v and v is an internal node corresponding to a. In particular, $\mu_a(v)$ is a* schema-based split *if v is a schema-based node in T' (i.e., v has two children and the corresponding arcs are labelled as $\exists$ and $\nexists$); $\mu_a(v)$ is a* value-based split *if v is a value-based node in T' (i.e., v has two or more children and the corresponding arcs are labelled with a condition expressed over the domain of a). Value-based splits are made as follows:*

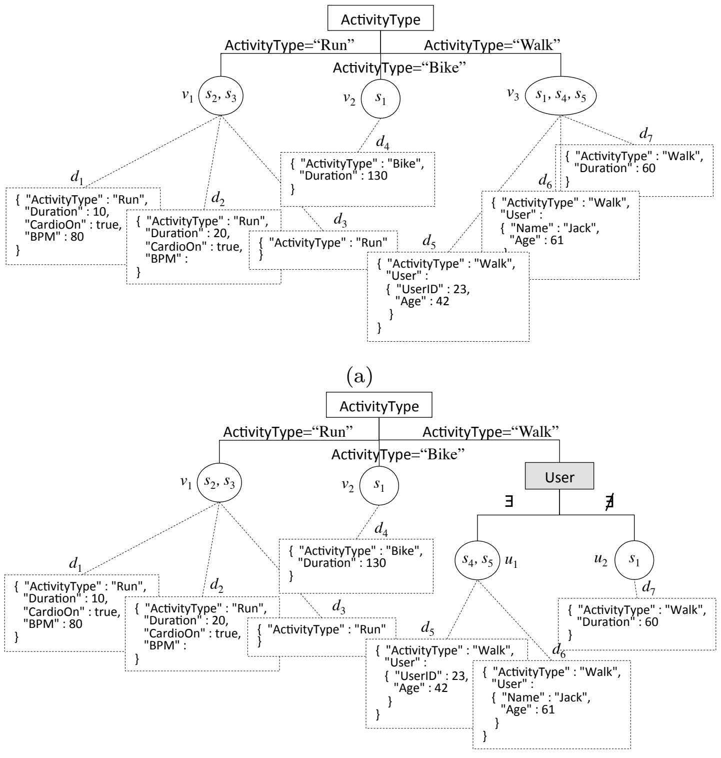
- *If a is numeric, $\mu_a(v)$ has two children and the corresponding arcs are labelled with conditions $a \leq$ "val" and $a >$ "val", respectively.*
- *If a is categorical, $\mu_a(v)$ can be either* binary *or* multi-way*. In the first case, it has two children and the corresponding arcs are labelled with conditions $a =$ "val" and $a \neq$ "val", where "val" is a value of the domain of a. In the second case, it has one child for each value "val" in the domain of a and each corresponding arc is labelled with condition $a =$ "val".*

Example 22 *Figure 7.3 shows an example of schema-based split at an early stage of the building of the schema profile of Figure 7.1. Initially, T has three leaves v_1, v_2, v_3 (resulting from a multi-way, value-based split on the categorical attribute* ActivityType*). The schema-based split on attribute* User *creates a new schema profile where v_3 becomes a schema-based node with two children, u_1 and u_2. A further value-based split on the numeric attribute* User.Age *creates two additional leaves as depicted in Figure 7.1.*

As mentioned in Section 7.3, we consider a schema profile to be explicative if it prefers value-based nodes over schema-based nodes, because the latter acknowledge that there is a difference between two r-schemata but do not really explain its reason. Thus, we will evaluate explicativeness using the number of schema-based nodes: the lower this number, the more explicative the schema profile.

7.5.2 Precision

A distinguishing feature of divisive approaches is the function they adopt to quantify the "purity" of the leaves where observations are classified, where a leaf is said to be



```
d1: { "ActivityType" : "Run",
      "Duration" : 10,
      "CardioOn" : true,
      "BPM" : 80
    }
d2: { "ActivityType" : "Run",
      "Duration" : 20,
      "CardioOn" : true,
      "BPM" :
    }
d3: { "ActivityType" : "Run"
    }
d4: { "ActivityType" : "Bike",
      "Duration" : 130
    }
d5: { "ActivityType" : "Walk",
      "User" :
      { "UserID" : 23,
        "Age" : 42
      }
    }
d6: { "ActivityType" : "Walk",
      "User" :
      { "Name" : "Jack",
        "Age" : 61
      }
    }
d7: { "ActivityType" : "Walk",
      "Duration" : 60
    }
```

Fig. 7.3 Schema profile before (a) and after (b) a schema-based split

pure if all its observations share the same class. The most common function used to this end is *entropy* [159], whose definition —properly contextualized to our application domain— we include below.

Definition 19 (Entropy and Gain) *Let D be a collection, $S(D)$ be the set of distinct r-schemata of the documents in D, and T be a schema profile for D with leaves $v_1, \ldots, v_m$. The entropy of leaf v_j is*

$$entropy(v_j) = -\sum_{s \in S(D_v)} \frac{|D_{v_j}|_s}{|D_{v_j}|} \log \frac{|D_{v_j}|_s}{|D_{v_j}|} \tag{7.1}$$

where $\frac{|D_{v_j}|_s}{|D_{v_j}|}$ is the probability of r-schema s within leaf v_j. Leaf v_j is said to be pure *if $entropy(v_j) = 0$. The* entropy *of T is then defined as the weighted sum of the entropies of the leaves of T:*

$$entropy(T) = \sum_{j=1}^{m} \frac{|D_{v_j}|}{|D|} \cdot entropy(v_j) \tag{7.2}$$

where $\frac{|D_{v_j}|}{|D|}$ *is the probability of leaf* v_j*. Let* $\mu_a(v_j)$ *be a split that creates two or more leaves,* $u_1, \ldots, u_r$*, resulting into a schema profile* T'*; the* gain *related to* $\mu_a(v_j)$ *is defined as the difference between the entropy of* T *and the one of* T'*:*

$$gain(\mu_a(v_j)) = entropy(T) - entropy(T') = \\ = \frac{|D_{v_j}|}{|D|} \cdot entropy(v_j) - \sum_{k=1}^{r} \frac{|D_{u_k}|}{|D|} \cdot entropy(u_k) \quad (7.3)$$

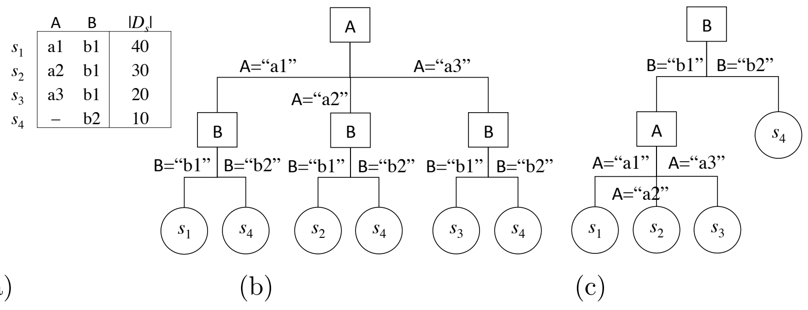
Entropy is strictly related to precision as informally defined in Section 7.3: within a schema profile with null entropy, all the documents included in each leaf share the same r-schema, thus the schema profile has maximum precision.

Entropy tends to decrease with each split; the higher the gain (i.e., the decrease in entropy), the more convenient the split. It is well-known [30] that gain-based criteria are biased and tend to privilege multi-way splits on attributes with several values. A classical solution to this problem is to consider binary splits only, or to add a weighting factor to normalize the gain for different attributes (so-called *gain ratio* criterion) [141].

7.5.3 Conciseness

Entropy is focused on node purity, hence its minimization often leads to split observations of the same class among several leaves; this is more frequent when the number of classes is high [154], as normally happens in our context. While this is a secondary problem in generic classification problems, where the precision of the resulting model is more important than its readability, it becomes critical in schema profiling since it conflicts with the conciseness requirement. Indeed, in our context, each r-schema might end up for being explained by a wide set of rules, thus precluding users from getting a concise picture of schema usage.

Example 23 *Let* D *be a collection with 100 documents, 4 r-schemata* $s_1, \ldots, s_4$*, and two attributes,* A *and* B*. The values of the attributes for the different r-schemata are listed in Figure 23.a (symbol "–" means "any value"), while two possible schema profiles are shown in Figures 23.b,c. From the point of view of schema usage, the tree in (c) is clearly the one that best represents the collection, with each r-schema being reached by a single path in the tree. Nevertheless, the tree chosen by an entropy-based algorithm would be that in (b), where* s_4 *is doubled. Indeed, called* v *the root, the split of* v *on attribute* A *has gain 1.39, while the split on* B *has gain 0.47.*



	A	B	$\|D_s\|$
s_1	a1	b1	40
s_2	a2	b1	30
s_3	a3	b1	20
s_4	–	b2	10

Fig. 7.4 A collection (a) and two possible schema profiles (b,c) (see Example 23)

To evaluate conciseness of schema profiles, in this section we propose a measure called *schema entropy*. As stated in Section 7.3, our requirement is reduce the number of rules provided by maximizing the cohesion of the documents that share the same r-schema —a maximally concise schema profile is one where there is a single rule for each r-schema. So we invert the original definition of entropy to relate it to the purity of the r-schemata instead of the purity of the leaves: in terms of entropy, a leaf is pure if it contains only documents with the same r-schema; in terms of schema entropy, an r-schema is pure if all its documents are in the same leaf. The schema entropy of a degenerate schema profile where all the documents are included into a single node (the root) is 0; clearly, when a split is made, the schema entropy can never decrease, so the concept of gain is replaced by that of *loss* (the lower the loss, the more convenient the split).

Definition 20 (Schema Entropy and Loss) *Let D be a collection, $S(D)$ be the set of distinct r-schemata of the documents in D, and T be a schema profile for D with leaves $v_1, \ldots, v_m$. The schema entropy of r-schema $s \in S(D)$ is*

$$sEntropy(s) = \sum_{j=1}^{m} \frac{|D_{v_j}|_s}{|D|_s} \log \frac{|D_{v_j}|_s}{|D|_s} \tag{7.4}$$

The schema entropy *of T is then defined as*

$$sEntropy(T) = -\sum_{s \in S(D)} \frac{|D|_s}{|D|} \cdot sEntropy(s) \tag{7.5}$$

Let $\mu_a(v_j)$ be a split resulting into schema profile T'; the loss *related to $\mu_a(v_j)$ is defined as:*

$$loss(\mu_a(v_j)) = sEntropy(T') - sEntropy(T) \tag{7.6}$$

Definition 20 implies that the loss of split $\mu_a(v_j)$ is 0 iff, for each r-schema $s \in S(D_{v_j})$, all the documents with r-schema s belonging to D_{v_j} are put into a single leaf of T'. As a consequence, all schema-based splits have null loss.

Example 24 *With reference to Example 7.4, the split on attribute* A *has loss 0.16, while the split on attribute* B *has loss 0; therefore, the schema profile in Figure 7.4.c is the most convenient from the point of view of schema entropy.*

Let m be the number of leaves and n the number of r-schemata; we note that:

- The schema entropy of a schema profile T can be 0 only if $m \leq n$ (because clearly, if $m > n$, at least one r-schema appears in more than one leaf).
- The entropy of a schema profile T can be 0 only if $m \geq n$.

Although these observations apparently suggest that entropy and entropy schema are conflicting, actually their goals are not mutually exclusive. Indeed, splitting a node so that documents with the same r-schema are kept together, means putting documents with different r-schemata in separate children; so, the splits determining low or null loss tend to yield a high gain as well. In Section 7.6 we will show how our approach builds on both entropy and entropy schema to achieve a trade-off between conciseness and precision of schema profiles.

We close this section with an important remark. As already stated, our definition of schema profile (Definition 17) accommodates both binary and n-ary trees —depending on whether binary or multi-way splits are allowed. However, we observe that an n-ary schema profile can always be translated into binary form without changing its entropy and schema entropy (because they only depend on how documents are partitioned among the leaves). Besides, multi-way splits on high-cardinality categorical attributes produce a strong fragmentation of r-schemata into leaves and an undesired increase in schema entropy. For these reasons, in the following we will focus on binary schema profiles.

7.6 Building Schema Profiles

As already mentioned, our algorithm is inspired by C4.5 [142] in its Weka implementation (named J48). C4.5 implements a divisive approach that starts by creating a single node that includes all the observations in the dataset; as such, according to Definition 19, this root node has maximum entropy. Then the algorithm iteratively splits each leaf into two new leaves that include subsets of observations; since the final goal is to minimize the entropy, at each iteration the split that maximizes the gain is greedily chosen. Eventually, some post-processing can be applied to reduce the size of the tree. The algorithm stops when either all the leaves are pure or all the splits are statistically irrelevant.

In our domain, recalling the user requirements illustrated in Section 7.3, we can informally state our final goal as follows:

Given collection D find, among the schema profiles for D with null entropy, one that (i) has minimum schema entropy on the one hand, and (ii) has the minimum number of schema-based nodes on the other.

Null entropy ensures that each leaf includes documents belonging to one r-schema only (i.e., the schema profile is *precise*), schema-entropy minimization calls for having a single rule to explain each r-schema (the schema profile is *concise*), while minimization of schema-based nodes ensures that most conditions are meaningful (the schema profile is *explicative*). Obviously, to fit this goal, the splitting and stop strategies of the C4.5 algorithm have to be modified by considering schema entropy on the one hand, and by taking into account the distinction between schema-based and value-based splits on the other.

A solution for our problem can be always found since, as no two r-schemata with exactly the same attributes exist, any schema profile can be extended using schema-based nodes until each leaf includes documents belonging to one r-schema only (i.e., the schema profile has null entropy). On the other hand, the (trivial) schema profiles including only schema-based nodes have null schema entropy but they do not meet subgoal (ii). Indeed, subgoals (i) and (ii) may be conflicting; to rule the trade-off, in our approach we allow schema-based splits only if no value-based split satisfies the following quality criterion:

Definition 21 (Valid Split) *A value-based (binary) split $\mu_a(v)$ is* valid *if $loss(\mu_a(v)) \leq \epsilon$ and $gain(\mu_a(v)) \geq \omega$, where $\epsilon, \omega \geq 0$ are thresholds.*

As we will show in Section 7.7, through ϵ and ω users can fine-tune the trade-off between subgoals (i) and (ii) so as to obtain satisfactory schema profiles depending on the specific features of the dataset. More specifically: the higher ϵ, the more the user is privileging schema profiles with a few schema-based nodes —i.e., explicative ones; the higher ω, the more the user is favoring schema profiles whose value-based nodes considerably decrease entropy —i.e., concise ones.

Algorithm 3 BuildSchemaProfile (BSP)

```
Input $T$, a schema profile; $v$, the leaf of $T$ to be split; $\epsilon$ and $\omega$, two thresholds
Output $T$, a new schema profile where $v$ is split
 1: $bestSplit \leftarrow \varnothing$
 2: $rs_v \leftarrow \cup_{s \in S(D_v)} s$                    ▷ Set of attributes in the r-schemata of the documents in $v$
 3: for $a \in rs_v$ s.t. $type(a) =$ primitive do                              ▷ Evaluate value-based splits
 4:     $\mu_a(v) \leftarrow FindBestSplit(a, \epsilon, \omega)$
 5:     if $\mu_a(v)$ is valid $\wedge \mu_a(v) \prec bestSplit$ then
 6:         $bestSplit \leftarrow \mu_a(v)$
 7: if $bestSplit = \varnothing$ then                                ▷ If no valid value-based split is found...
 8:     for $a \in rs_v$ do                                              ▷ ...evaluate schema-based splits
 9:         $\mu_a(v) \leftarrow SchemaBasedSplit(a)$
10:         if $\mu_a(v) \prec bestSplit$ then
11:             $bestSplit \leftarrow \mu_a(v)$
12: for $u \in Children(\mu_a(v))$ do                          ▷ For each leaf $u$ generated by the split...
13:     $AddChild(T, v, u)$                                         ▷ ...add $u$ to $T$ as a child of $v$...
14:     if $|S(D_u)| > 1$ then                                          ▷ ...and, if $u$ is not pure, ...
15:         $T \leftarrow BuildSchemaProfile(T, u)$                                          ▷ ...split it
16: return $T$
```

Consistently with the final goal stated above, we define a criterion for comparing two splits as follows:

Definition 22 *Given two splits $\mu(v)$ and $\mu'(v)$ (possibly on different attributes), we say that $\mu(v)$ is* better *than $\mu'(v)$ (denoted $\mu(v) \prec \mu'(v)$) if either (i) $loss(\mu(v)) < loss(\mu'(v))$, or (ii) $loss(\mu(v)) = loss(\mu'(v))$ and $gain(\mu(v)) > gain(\mu'(v))$.*

The pseudocode of Algorithm 3 (called BSP from now on) implements our approach. It is a recursive procedure that splits a generic leaf v of the tree T representing the schema profile; it is initially called with a degenerate tree consisting of one single node, and stops when all leaves are pure. Value-based splits are tried first (lines 3 to 6); candidate attributes for splitting are those that are present in at least one of the documents in D_v. For each candidate attribute a, function *FindBestSplit* finds the best binary split by computing gain and loss for each condition on a (as per Definition 18) and using Definition 22 for split comparison. If no valid value-based split is found for the given threshold ϵ, schema-based splits are tried (lines 8 to 11). Function *SchemaBasedSplit*(a) returns the schema-based split for attribute a; since all schema-based splits are considered to be valid, one schema-based split is always found at this stage. Finally, T is extended by adding the new leaves generated by the best split found as children of v (lines 12 to 15). If a new leaf u is pure, recursion stops; otherwise, u is recursively split.

In practice, the Weka J48 algorithm we use for experimental comparisons requires a dataset structured as a table where each row represents an *observation* (in our case, a document) and each column represents a feature to be used for classification; one

Table 7.1 Portion of the dataset for our running example

rsId	ActivityType_primitive	CardioOn_primitive	BPM_primitive	User.Age_primitive	...	exists_ActivityType_primitive	exists_CardioOn_primitive	exists_BPM_primitive	exists_User_object	exists_User.Age_primitive	...
s0	Bike	null	null	null	...	1	0	0	0	0	...
s1	Run	false	null	null	...	1	1	0	0	0	...
s1	Walk	false	null	null	...	1	1	0	0	0	...
s2	Run	true	120	null	...	1	1	1	0	0	...
s3	Run	true	null	null	...	1	1	0	0	0	...
s4	Walk	true	null	42	...	1	1	0	1	1	...
s5	Walk	true	null	65	...	1	1	0	1	1	...

additional column is used to store the class each observation belongs to. So we define the *dataset* associated to a collection as follows.

Definition 23 (Dataset) *Let D be a collection. The* dataset *associated to D is a table with the following structure:*

1. *a column named* rsId*;*
2. *one* schema-based column *named* exists_*path*(a)_*type*(a) *for each attribute* $a \in rs(D)$*; and*
3. *one* value-based column *named path*(a)_*type*(a) *for each attribute* $a \in rs(D)$ *such that type*$(a) =$ primitive.

The dataset includes a row for every $d \in D$*, such that*

1. rsId *stores a unique identifier given to* $rs(d)$ *within* $S(D)$*;*
2. exists_*path*(a)_*type*$(a) = 1$ *if* $a \in rs(d)$*, 0 otherwise;*
3. *path*(a)_*type*(a) *stores the value taken by a in d if* $a \in rs(d)$*,* null *otherwise.*

Example 25 *Table 7.1 shows a portion of the dataset for our running example; for space reasons, some columns are omitted.*

Table 7.2 Dataset features

Name	*Origin*	$\lvert D\rvert$	$\lvert S(D)\rvert$	*#columns*	*Optimal*	*Balanced*
SD1					yes	yes
SD2	synthetic	10 K	8	$4+7$	no	yes
SD3					yes	no
RD1	fitness	5 M	6	$6+10$	yes	no
RD2	fitness	767 K	139	$35+38$	no	no
RD3	sw develop.	473 K	122	$90+149$	no	no

7.7 Experimental Results

In this section we evaluate the efficiency and effectiveness of our approach; we recall from Section 7.5 that our tests are focused on binary trees. For testing we use a set of both synthetic and real-world datasets, whose characteristics are summarized in Table 7.2. *Optimal* indicates whether there exists a schema profile that is both concise (i.e., with null schema entropy) and explicative (i.e., with no schema-based nodes); *Balanced* indicates whether all the r-schemata have roughly the same number of documents; *#columns* is the number of value-based columns plus the number of schema-based columns in the dataset.

Synthetic datasets (i.e., SD1, SD2 and SD3) are relatively small and have been created using a simple rule-based generator: we devised two schema profiles (shown in Figure 7.5) and then used them as a model to generate the data. The four value-based columns correspond to primitive attributes, shared by all documents (i.e., no value-based column has missing values); the presence/absence of three more object attributes allows eight r-schemata to be distinguished. SD1 is the simplest dataset, where each r-schema has about 1/8 of the 10 000 documents. SD2 presents a balanced but non-optimal situation obtained by assigning, for each r-schema, multiple values to attribute subtype —thus forcing a loss in schema entropy when splitting on that attribute. Lastly, SD3 is generated with the same rules as SD1 but presents an unbalanced situation, where four r-schemata have 9/10 of the documents.

As to real-world datasets, RD1 and RD2 have been acquired from a company that sells fitness equipment, and they contain the registration of new workout sessions and the log of the activities carried out during a workout session, respectively. RD3 has been acquired from a software development company and contains the log of the errors generated by the deployed applications. These three datasets have a key role in evaluating our approach, because they include a large number of documents and present a higher number of attributes and r-schemata.

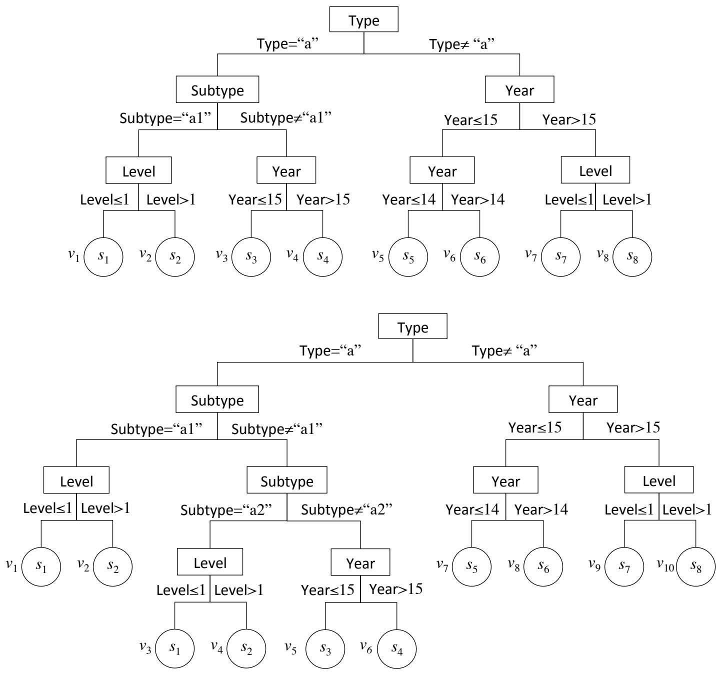


Fig. 7.5 The schema profiles used to generate the synthetic datasets SD1 and SD2 (top), and SD3 (bottom)

While synthetic datasets have been directly created by a rule-based generator, real-world datasets have been derived from MongoDB collections as follows: we first used a Javascript script for MongoDB to parse the documents, extract the r-schemata, and export the data in tabular form; then, an R script has been used to pre-process data (e.g., to discretize numeric columns and reduce the number of distinct values in categorical columns) before submitting them to BSP.

7.7.1 Effectiveness

To evaluate the effectiveness of our approach we must verify whether BSP is really capable of identifying the rules that drive the use of different schemata in documents, which can be done by comparing the schema profiles obtained by BSP with a baseline. The schema profiles are also compared with those obtained from the original version of the Weka J48 algorithm, which only uses entropy. Since schema-based splits typically yield higher gain than value-based ones (and, therefore, are favored by J48), we also run J48 on reduced versions of the datasets that only include value-based columns; we will label with J48-V and J48-VS the schema profiles obtained by ignoring or considering schema-based attributes, respectively.

To compare two schema profiles we consider the document partition induced by the tree leaves but also the tree structure (as two trees differing in their split attributes and/or structure may induce similar partitions of the documents); more specifically:

- To measure the difference in document partition we rely on the *HR-index* [79], which is a fuzzy variation of the well-known *Rand index* [143]. The basic idea behind the Rand index is that two partitions p_1 and p_2 are similar if the couples of documents that are together (separate) in p_1 are also together (separate) in p_2. The HR-index simply extends this idea to fuzzy partitions. The value of the index ranges from 0 (completely different partitions) to 1 (identical partitions).
- To measure the difference in tree structure we adopt a variation of the classic *tree edit-distance* (TED), which originally defines the distance between two trees as the length of the minimal cost-sequence of node-edit operations (i.e., insert, delete, or rename) that transforms one tree into the other [158]. TED cannot be directly calculated between two schema profiles, as it works only on labeled trees with no semantics on the arcs; therefore, given schema profile T, for the sake of TED calculation we define a simplified tree where
  - each internal node is labeled with the name of its split attribute;
  - each leaf is labeled with keyword “leaf”; and
  - the conditions on the arcs are removed.

  Importantly, TED does not take into account the depth of a node in the tree (i.e., an edit operation on the root costs the same as an edit operation on a leaf). Since this would have a strong impact when comparing decision trees (where the order of decisions is a crucial aspect), we adopt a modified version of TED which weighs the cost of an edit operation made on a node with the depth of that node; in particular, given a tree of height p and a node v at depth α ($1 \leq \alpha \leq p$), the cost of an edit operation on v is $1/\alpha$. The minimum value for TED is 0 (identical trees), while the maximum value clearly depends on the size of the compared trees.

For synthetic datasets, our testing baseline are the schema profiles used to create the datasets (Figure 7.5). Table 7.3 shows the results of the tests made on these datasets to measure the quality of schema profiles with reference to the following three critical situations (the value of ω is set to zero in these tests, as its manipulation has no remarkable effect on the synthetic datasets due to their small size):

Table 7.3 Effectiveness tests on synthetic datasets

Test	*Dataset*	*Algorithm*	ϵ	*Schema entropy*	*#leaves*	*% schema-based nodes*	*TED*	*HR-index*
T1	SD1	J48-V	—	0	8	0%	0	0
		J48-VS	—	0	8	14%	0	0
		BSP	0.1	0	8	0%	0	0
T2	SD1	J48-V	—	1.92	38	0%	9.06	0.061
		J48-VS	—	0	8	29%	1.33	0
		BSP	0.1	0	8	14%	0.5	0
T3	SD1	J48-V	—	2.65	33	0%	9.03	0.097
		J48-VS	—	0	8	42%	1.67	0
		BSP	0.1	0	8	42%	1.17	0
T4	SD2	J48-V	—	0.59	15	0%	3.45	0
		J48-VS	—	0	8	28%	2.32	0.013
		BSP	0.1	0	8	14%	1.48	0.013
			0.3	0.21	10	0%	0	0
T5	SD3	J48-V	—	0.10	12	0%	4	0.001
		J48-VS	—	0	8	14%	2.33	0.125
		BSP	0.1	0	8	0%	0	0

1. *Incomplete information*: in this case the use of different schemata is ruled by either the values of attributes that are not present in the documents (i.e., in the application logic there is a conditional instruction that creates documents with different schemata depending on the value of some variable v, but the value of v is not stored in the documents) or the values of non-primitive attributes for which there is no value-based column in the dataset (e.g., arrays). We start from the simplest dataset, SD1 (test T1), and progressively remove value-based columns Subtype (test T2) and Level (test T3); threshold ϵ is set at 0.1 for BSP. Table 7.3 shows that, in T1, both BSP and J48-V produce a schema profile identical to the baseline (both TED and HR-index are zero). Conversely, the structure of the tree obtained by J48-VE is different: since the algorithm does not take into account the difference between value- and schema-based columns, it chooses to create schema-based splits —because they yield a higher gain— though this is in contrast with our goal of explicativeness. Obviously, as value-based columns are removed, the algorithms are no more able to reproduce the baseline (as reflected by the values of TED); however, they behave quite differently. The split strategy of BSP is specifically designed to keep documents with the same r-schema together; as a result, BSP resorts to schema-based conditions to recover the lost information

and eventually provides the same document partitioning as the baseline (e.g., by replacing the split on Subtype with the split on Exists_b in T2). The entropy measure guiding J48 does not equally encourage conciseness, thus J48-V finds alternative splits among those that only minimize entropy. This leads J48-V to build a more complicated schema profile (as shown by the higher number of leaves) with a document partitioning that differs from the baseline. The baseline partitioning is reproduced by J48-VS instead, although schema-based conditions are overused. Overall, TED shows that the schema profile provided by BSP is always the closest to the baseline.

2. *Non-optimal dataset*: in dataset SD2, the use of different schemata is ruled by complex conditions that are not considered by our search strategy; hence, no concise and explicative schema profile can be found and one of these two features must be sacrificed. Test T4 shows how this trade-off can be achieved by adjusting the value of ϵ. With the lowest value of ϵ (i.e., 0.1), BSP is less tolerant to the separation of documents of the same r-schemata into different leaves (i.e., to the definition of multiple rules for the same r-schema); this results in BSP preferring schema-based conditions and generating a concise (8 leaves and zero schema entropy) but inaccurate (with high TED and non-zero HR-index) schema profile. The baseline actually splits the documents of r-schemata s_1 and s_2 into 4 leaves (as shown in Figure 7.5), and this result can be achieved by increasing ϵ to 0.3 —at the expense of a substantial loss in terms of schema entropy. As to J48-V and J48-VS, the results show that they perform very poorly: the former is able to reproduce the baseline partitioning, but at the expense of a complex tree; the latter provides a more concise tree, but schema-based conditions are used quite differently, eventually leading to a schema profile that is quite far from the baseline.

3. *Unbalanced dataset*: in this case the documents are unevenly distributed among the r-schemata. In this case we run a single test (T5) on the unbalanced dataset SD3, with threshold ϵ still set at 0.1. The results show that only BSP is capable of perfectly reproducing the baseline, as the schema entropy measure favors a pure separation of the r-schemata even if the entropy gain is limited (due to the unbalanced distribution). Conversely, the entropy measure tends to maximize the gain by separating the documents belonging to the most frequent r-schemata, even if the documents of the less frequent r-schemata end up to be split in multiple leaves. Both J48-V and J48-VS provide a schema profile quite different

from the baseline, as the first inverts the order of the value-based conditions, while the second still tends to rely on schema-based conditions.

For our real-world datasets a complete baseline is not available. As evidenced in the introduction, this is quite common when the applications undergo several evolutions and retrieving all the rules driving schema usage requires a huge effort from users and direct access to the application logic. In particular, for all real dataset except RD1 (which is quite simple), our users, after a four-hours meeting, have only been able to provide a very small schema profile consisting of the two or three (value-based) conditions that —to the best of their knowledge— are representative of the schema usage.

Given such premise, we start by studying the behavior of BSP with different values of ϵ and ω. The three diagrams in Figure 7.6 show the effects of the two thresholds on the schema profile for RD2 in terms of conciseness (given by the number of leaves and by the schema entropy) and its explicativeness (given by the percentage of value-based nodes). The first observation is that, when a very concise schema profile is requested by the user (i.e., $\epsilon = 0$ and the schema entropy is therefore forced to be null), the major risk is to renounce to explicativeness (as shown by the low percentage of value-based nodes). As ϵ is increased, the general tendency is to create more explicative schema profiles. However, by allowing splits that increase the schema entropy, BSP may be inclined to choose those splits that single out only a small fraction of the documents (i.e., splits yielding a very low loss, but also a very low gain); in this case, the risk is to obtain a very complex schema profile. Conversely, the increase of ω favors conciseness by filtering out the aforementioned splits. The obvious drawback is the decrease of value-based conditions: if the only available splits have a gain lower than ω, BSP has no choice but to rely on schema-based splits.

As the order of the conditions is very relevant, users may be interested more in the upper parts of the schema profile than in the lower ones. So, with the next test we analyze the structure of the schema profile by segmenting the tree at different levels of depth. Figure 7.7 shows the level-by-level decomposition of the schema profile generated on RD2. Based on the results of the previous test, we set $\epsilon = 0.01$ and $\omega = 0.1$. The figure shows that the entropy rapidly decreases, while the low schema entropy ensures the conciseness of the schema profiles across every stage. Most importantly, the percentage of schema-based nodes is kept quite low within the first levels, denoting a good degree of explicativeness of the upper part of the schema profile. Schema-based conditions tend to increase in the lower parts, where a good degree of separation of the r-schemata has already been achieved (as proven by the low entropy).

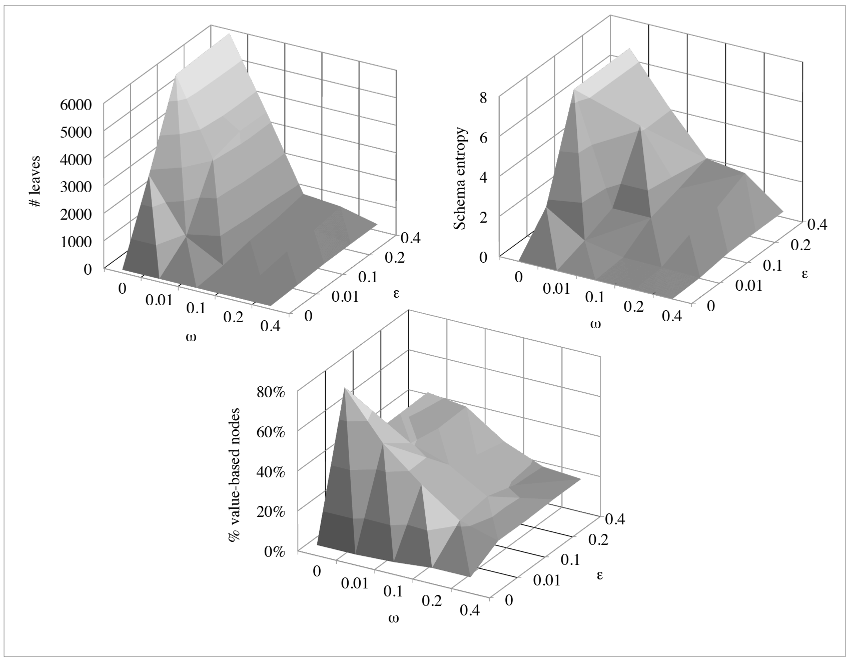


Fig. 7.6 The effects of different values of ϵ and ω on RD2

As a final test, we evaluate the quality of the schema profiles generated for real-world datasets with respect to the baseline provided by the user. Since the baseline is limited to two or three value-based conditions, a comparison against complete schema profiles would inevitably be unfair and reveal great dissimilarities. Therefore, the values of TED have been determined by limiting each schema profile to a number of levels that equals the one in the baseline; the calculus of the HR-index is omitted for the same reason. As with synthetic datasets, the comparison is also made with J48-V and J48-VS. The values of ϵ and ω for RD1 and RD3 have been determined by a statistical evaluation, as previously shown for RD2. Table 7.4 shows the results of the test. The most obvious observation is the higher complexity of the schema profiles generated by J48-V, together with its lack of conciseness —which is expected. In RD1, both J48-VS and BSP provide a very concise schema profile; however, only the one by BSP matches with the baseline, as the first split chosen by J48-VS is a schema-based one. The overuse of schema-based columns by J48-VS is evident in RD2 and RD3, where a very concise but not explicative schema profile is found. The quality of the schema profile by BSP is validated by TED, which shows that BSP always gets the closest to the baseline.

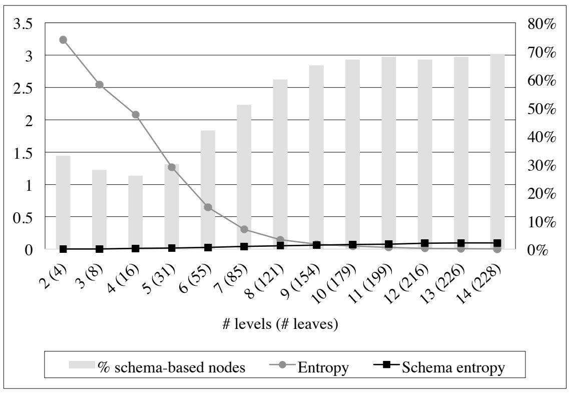


Fig. 7.7 Level-by-level decomposition of the schema profile generated on RD2 with $\epsilon = 0.01$ and $\omega = 0.1$

Table 7.4 Effectiveness tests on real-world datasets

Dataset	*Algorithm*	ϵ	ω	*Schema entropy*	*#leaves*	*% schema-based nodes*	*TED*
	J48-V	-	-	2.16	48	0%	0
RD1	J48-VS	-	-	0	6	60%	1
	BSP	0.01	0.1	0	6	80%	0
	J48-V	-	-	3.69	521	0%	2.33
RD2	J48-VS	-	-	0	115	93%	1.33
	BSP	0.01	0.1	0.10	228	69%	1.33
	J48-V	-	-	2.07	243	0%	1
RD3	J48-VS	-	-	0	85	73%	1
	BSP	0.01	0.01	0.10	231	50%	0.5

7.7.2 Efficiency

The driving factor in estimating the complexity for computing the gain and loss of a split is the number of documents, $|D|$ (since the number of r-schemata, $|S(D)|$, and the number of leaves, m, are clearly quite lower than $|D|$). Under this assumption, we know from the literature that the complexity for evaluating all the possible splits of leaf v on attribute a by computing their gain and loss is $O(|D_v|)$ if a is categorical (including the case in which a is a schema-based columns), $O(|D_v| \cdot (log|D_v| + 1))$ if a is numeric [152]. Within an iteration of BSP, this cost is paid whenever functions *FindBestSplit* and *SchemaBasedSplit* are called. In turn, the number of calls depends on the number of attributes belonging to rs_v.

Table 7.5 Efficiency tests on real-world datasets

Dataset	*Algorithm*	ϵ	ω	*Avg time*	*#FBS*	*#SBS*	*Avg #docs per FBS*	*Avg #docs per SBS*
	J48-V	-	-	172	1270	-	225 K	-
RD1	J48-VS	-	-	43	27	12	2839 K	3290 K
	BSP	0.01	0.1	42	26	11	3320 K	3162 K
	J48-V	-	-	125	7515	-	27 K	-
RD2	J48-VS	-	-	139	2784	550	59 K	184 K
	BSP	0.01	0.1	116	2851	602	57 K	91 K
	J48-V	-	-	156	4282	-	36 K	-
RD3	J48-VS	-	-	229	991	1095	124 K	138 K
	BSP	0.01	0.01	191	1984	558	64 K	39 K

Formally modeling the complexity of the basic operations is not sufficient to evaluate the overall BSP execution time. The number of recursive calls is related to the effectiveness of the purity measures, to the availability of useful value-based attributes, and to the inherent complexity of the dataset. To properly analyze this holistic bundle of factors we compare the BSP execution time with the ones of J48-V and J48-VS on real-world datasets[9]. For the same tests of Table 7.4, Table 7.5 shows:

- the average execution times (in seconds) to build the schema profiles;
- the number of times that functions *FindBestSplit* and *SchemaBasedSplit* are called (shown as *#FBS* and *#SBS*, respectively);
- the average number of documents involved when functions *FindBestSplit* and *SchemaBasedSplit* are called (shown as *Avg #docs per FBS* and *Avg #docs per SBS*, respectively);

Note that (i) the number of iterations does not necessarily reflect the size of the tree, because J48 adopts post-pruning techniques that eventually reduce the size of the schema profile; (ii) the time for post-pruning is not considered because of its irrelevance (not greater than a second); (iii) BSP is not penalized by the added calculation of schema entropy, since this requires the same data used to calculate the entropy (as per Definition 20).

[9]Synthetic datasets are not considered for evaluating efficiency since their execution time is below one second.

By looking at the results, we initially observe that BSP is always faster than J48-VS; this is due to the fact that, differently from BSP, J48-VS always tries all schema-based splits. In RD1 and RD3, this is confirmed by the higher value of #SBS, while in RD2 it is explained by the average number of involved documents: whereas BSP and J48-VS make approximately the same number of calls to the functions, BSP resorts to schema-based splits only in the lower levels of the schema profile (in fact, the average number of documents processed per SBS is half the one of J48-VS), thus gaining a better performance on the upper levels. As to J48-V, the effect of the absence of schema-based columns is twofold: on the one hand, the performance of function FBS is obviously faster, since less attributes must be processed; on the other hand, the split strategy (which only relies on the entropy gain related to value-based splits) does not converge quickly to a schema profile with null entropy, thus requiring a much larger number of calls to the FBS function.

7.8 Conclusions

In this chapter we have presented an approach to schema profiling for document-oriented databases. To the best of our knowledge, ours is the first approach to schema profiling based on extensional information. The idea is to capture the rules that explain the use of different schemata within a collection through a decision tree whose nodes express either value-based or schema-based conditions. Guided by the requirements elicited from users, we have proposed an algorithm called BSP that builds precise, concise, and explicative schema profiles. The experimental tests have shown that BSP is capable of achieving a good trade-off among these features and of delivering accurate (as compared to a baseline) schema profiles.

Our future work in this field will develop along two different perspectives. On the implementation side, we will incorporate user interaction in the schema profile building algorithm to give users a closer control on the trade-off among the different features of schema profiles and to inject additional knowledge in it. From a more research-oriented point of view, we will investigate how to take advantage of schema profiles when querying document collections; for instance, we will use schema profiles to give more effectiveness and flexibility to schema-on-read approaches to analytical queries in business intelligence contexts [109, 34].

Chapter 8

Cubeload: a benchmark generator of OLAP sessions

In this chapter we present CubeLoad, a parametric generator of workloads in the form of OLAP sessions, based on a realistic profile-based model. Differently from OLTP workloads, OLAP workloads are hardly predictable due to their inherently extemporary nature. Besides, obtaining real OLAP workloads by monitoring the queries actually issued in companies and organizations is quite hard. On the other hand, hardware and software benchmarking in the industrial world, as well as comparative evaluation of novel approaches in the research community, both need reference databases and workloads. After describing the main features of CubeLoad, we discuss the results of some tests that show how workloads with very different features can be generated.

As we recall from Section 1.2.3, this work is preparatory to the work on OLAP recommendations presented in Chapter 9.

8.1 Introduction

Differently from OLTP workloads, that are 90% frozen within operational applications, OLAP workloads are hardly predictable due to their inherently extemporary nature. Besides, obtaining real OLAP workloads by monitoring the queries actually issued in companies and organizations is quite hard because (i) OLAP queries are at the core of the decision-making process, hence they are jealously guarded by managers and administrators, and (ii) reconstructing OLAP sessions by interpreting the query log of a multidimensional engine operating in a multi-user context is very complex.

On the other hand, hardware and software benchmarking in the industrial world, as well as comparative evaluation of novel approaches in the research community, both need reference databases and workloads. To this end, some efforts have been done over the years to provide standard benchmarks. Specifically, in the OLAP context, the TPC-DS benchmark [136] has been recently developed; it is based on a fixed set of star schemata including 7 fact tables and 17 dimension tables, and it provides a workload featuring queries that address complex business problems and use a variety of access patterns.

The TPC-DS benchmark is carefully designed and offers a solid reference. However, especially in research papers, there is often a need for using benchmarks based on schemata with varying characteristic and on multiple alternative workloads with different features. For instance, it could be interesting to understand how the performance of a proposed approach varies with the number of dimensions in a cube, with the average branching factor of hierarchies, with the maximum length of sessions, or with the average selectivity of queries. In particular, generating parametric OLAP workloads is crucial to the experiments made in the context of OLAP prediction and recommendation, where the features of sessions and queries may have a strong impact on the approach effectiveness and efficiency. So, the papers in this context often rely on synthetically generated OLAP workloads, where queries and session are built in a completely random way based on a set of structural and statistical parameters [6–9]. Unfortunately, while these synthetic workloads serve well for efficiency tests, they cannot provide significant results for effectiveness tests because they do not lean on a realistic user model.

To fill this gap, in this chapter we present *CubeLoad*, a parametric generator of OLAP workloads [148]. The main features of CubeLoad are:

- No predefined multidimensional schema is used. The benchmarker[1] can create a workload for any multidimensional schema provided it has been exported in XML compliant with the Mondrian format.
- The workload is generated in the form of sessions, each including a variable number of aggregate queries. The main parameters used are related to a realistic profile-based workload model.
- Sessions are generated according to a set of four templates, that model recurrent types of user analyses.

[1]To distinguish users of OLAP front-ends from the users of CubeLoad, we will call *benchmarkers* the latter.

- If an instance of the multidimensional schema is available (in particular, in the form of a set of dimension tables), its data are used for generating instance-dependent (hence, more realistic) workloads.
- The generated workload is exported in XML to ensure maximum usability.

CubeLoad is written in Java and can be downloaded at http://big.csr.unibo.it/ downloads/CubeLoad.zip. It can be freely used by researchers, practitioners, and vendors whenever they need to create parametric bulk OLAP workloads for benchmarking and testing.

The chapter outline is as follows.

- In Section 8.2 we discuss the related literature.
- In Section 8.3 we describe the overall functional architecture of CubeLoad.
- In Section 8.4 we present the workload model
- In Section 8.5 we present the session templates.
- In Section 8.6 we discuss the results of some tests we made to profile the generated workloads
- In Section 8.7 we draw the conclusions.

8.2 Related Literature

A milestone in OLAP benchmarking is the TPC-DS [136], that models the decision support functions of a retail product supplier relying on multiple snowflake schemata with shared dimensions. The TPC-DS provides four classes of queries; in particular, the class of *iterative OLAP queries* is distinguished by the tendency of one query to be related to the previous query so as to create sequence of queries —essentially, OLAP sessions. Queries are randomly generated starting from four templates; however, there is no way of parameterizing the generation of sessions.

In [65] the authors introduce the concept of *workload profile* as a way for summarizing the features of an OLAP workload to support designers during logical and physical design. However, the profile used there has a merely statistical nature, and has no relationship with classes of users. Besides, only stand-alone queries are generated.

A workload for evolutionary analytics is proposed in [105] together with several test metrics and with a methodology for running the workload. The emphasis there is not on standard OLAP sessions but rather on queries that evolve over time (which may imply much more drastic changes than those obtained through OLAP operations) and are formulated over changing data and schemata.

A *Data Warehouse Engineering Benchmark* (DWEB) that allows to generate various ad-hoc synthetic data warehouses and workloads is presented in [28]. Though the DWEB workload is parameterized to fulfill data warehouse design needs, it does not create queries in sessions and is ruled by statistical parameters rather than by realistic assumptions.

The author of [135] starts from the query generator of the TCP-DS to define a set of rules that transform a SQL query into another SQL query similar to the original. However, this transformation works at a merely syntactical level (e.g., a new query can be created by changing the comparison operator in the selection predicate) and does not consider OLAP operations such as slicing and drilling.

In [162] the authors introduce a query generator to evaluate the quality of a query optimizer. Similarly to ours, the generator presented is schema-independent and is able to produce valid queries on any database. However, only OLTP queries are generated and, therefore, there is no mention of query sessions.

Finally, a benchmark on star schemata that extends the TPC-H is presented in [131]; the emphasis here is more on data schemata than on queries, so only 4 non-parameterized OLAP sessions (called query flights here) are provided.

8.3 Overview

A functional overview of the CubeLoad architecture is sketched in Figure 8.1. The main input is the multidimensional schema on which the workload is to be generated. To provide this input we adopt the XML specification used by Mondrian for its metadata [80].

To generate realistic selection predicates and enable report sizes to be estimated, dimension data are needed. These data can be fed into CubeLoad using the CSV (comma-separated values) format, which can be easily obtained by benchmarkers by exporting dimension tables.

Internally, CubeLoad includes five components:

1. The **user interface**, that allows benchmarkers to select the XML multidimensional schema to be used and choose values for global and profile parameters.

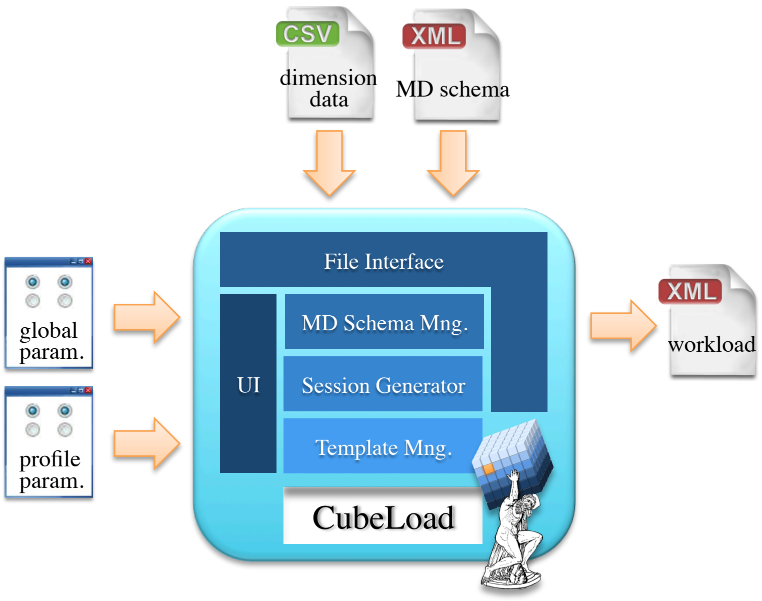


Fig. 8.1 Functional overview of CubeLoad

2. The **file interface**, in charge of reading and parsing XML and CSV input files and of writing XML output files.
3. The **multidimensional schema manager**, that builds an internal representation of cubes and dimension data.
4. The **session generator**, that runs the basic procedures for creating sessions respecting the constraints posed by global and profile parameters.
5. The **template manager**, that gives the session generator additional rules for creating sessions based on each template.

8.4 The Workload Model

The output of CubeLoad is an OLAP workload, defined as a set of *sessions*, i.e., a sequence of OLAP queries (see Chapter 2 for a formal definition of OLAP sessions). We call *report* the result of a single query; its size is the number of facts returned. Roughly, the size of a report can be estimated as the product of the domain cardinalities for all levels in the query group-by, reduced by considering the selectivity factors of the selection predicates; more accurate estimates can be computed if the sparsity of the cube is known [24].

In company settings, users of OLAP front-ends are normally grouped into profiles with different skills (e.g., CEO, marketing analyst, department manager) and involved in business analyses with different features (e.g., more or less repetitive, more or less complex). Importantly, different profiles generally have quite different permissions for accessing data; often, a profile has one or more *segregation predicates*, i.e., it can only view a specific slice of the cube data (e.g., a department manager can only access the sales for her department).

When a user logs to the OLAP front-end, she is typically shown a page where some predefined queries (which we call *seed queries*) are linked. Sometimes seed queries include a *prompt*, meaning that the front-end asks the user to select one value out of the domain of a level (often, the year). After choosing and executing one of these queries, the user starts applying a sequence of OLAP operations that progressively transform a query into another so as to build an analysis session. Features such as the number of seed queries available, the maximum size and complexity of reports returned by seed queries, and the average length of sessions may significantly depend on the typical ICT skills and business understanding for the users of each profile —besides on the quality of the OLAP fron-end.

To simulate the above setting, CubeLoad uses a set of parameters that rule workload generation and are distinguished into *global parameters* and *profile parameters.* The global parameters rule:

- the **number of distinct user profiles** to be simulated. Each profile simulates a specific class of OLAP users and is characterized by different values of the profile parameters. Each session is generated for one profile, so the sessions in the resulting workload can be naturally grouped into clusters; the more different the parameters for the profiles, the sharper the clusters.

- the **maximum number of measures** that can be returned by a single query. A report including several measures is hardly readable by anyone, so the value for this parameter mainly depends on how sophisticated the visualization modes supported by the OLAP front-end are.

- the **minimum and maximum size of seed query reports**. The size (i.e., number of cells) of a query result depends on the query group-by and on the presence of selection predicates. While during an unconstrained OLAP sessions users can (either consciously or unconsciously) formulate a query that returns a report with either negligible or huge size, seed queries are typically created by front-end programmers in such a way that their report size is reasonable. This is reason the reason why in our model the size of seed query reports ranges within a parametric interval.

- the **number of surprising queries**, whose meaning will be explained in Section 8.5 in relationship to the explorative template.

Each profile is then associated to a further set of parameters, that rule:

- the **number of seed queries**. Specialists' profiles have a large number of seed queries; managers' profiles may have a low number of seed queries.
- the **minimum and maximum length of sessions**. The values for these parameters depend on the ICT skills of the users of each profile and on the complexity of the analyses they usually carry out.
- the **number of sessions** to be created. The more intensive the use of the OLAP front-end for the users of a profile, the higher the value of this parameter.
- the **fraction of seed queries that include a year prompt**. This fraction depends on the time scope of decision-making tasks for each profile (operative profiles typically analyze daily to monthly trends, while managerial profiles are often interested in yearly trends).
- the **presence of a segregation predicate**. A segregation predicate is typically present in departmental or geographically-distributed profiles (e.g., production manager and sales manager for Italy).

The workload model is summarized in Figure 8.2 in the form of a UML class diagram.

8.5 Session Templates

Each session generated by CubeLoad for a given profile starts from one of the seed queries for that profile and evolves, consistently with global and profile parameters, according to a *template*. In its current implementation, CubeLoad uses four different templates for generating sessions:

1. **Slice-and-Drill**. In several OLAP front-ends, the default behavior when a user clicks on a row/column of a pivot table is to disaggregate the values for that row/column into its components, which in OLAP terms means slicing and drilling

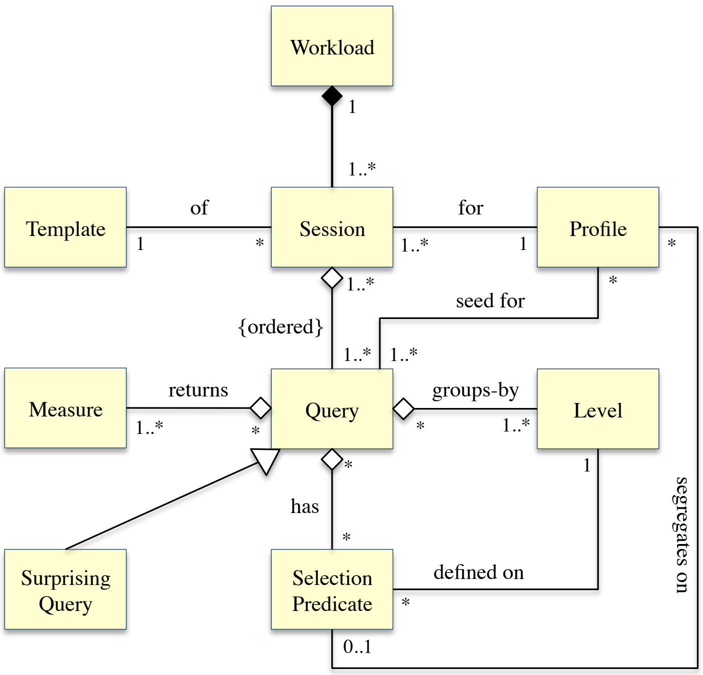


Fig. 8.2 UML workload model

down. For instance, starting from a report showing sales per state and year, clicking on 2013 would trigger a query showing sales per state and month of 2013, while clicking on Florida would trigger a query showing sales per Florida cities and year. In sessions based on this template, (non-segregated) hierarchies are progressively navigated by choosing a hierarchy h, a member v of the current group-by level l and creating a new query with selection predicate $l = v$ and group-by on the level l' that precedes l within h (i.e., $l' \dot{\succ} l$).

2. **Slice-All**. Users are sometimes interested in navigating a cube by slices, i.e., in repeatedly running the same query but with different selection predicates. In sessions based on this template, a level l of the group-by of the seed query is chosen, and new queries are generated by keeping the same group-by and adding selection predicates on the different members of l. For instance, starting from a query asking for the monthly sales by state for the video department, the subsequent queries could ask for the same report for the audio, the photo, and the PC departments.

3. **Explorative**. Some queries may return reports that are particularly interesting for most users, for instance because they show unexpected results (e.g., they show that the impact of a social policy is not the one that had been predicted)

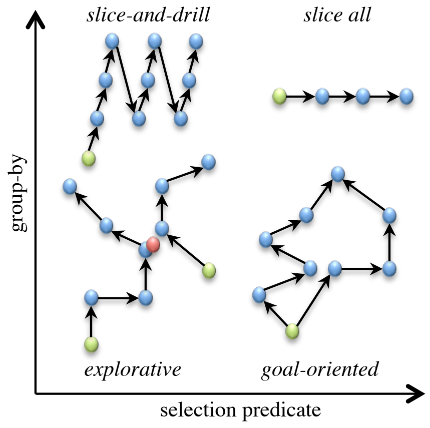


Fig. 8.3 Session templates (seed queries in green, surprising queries in red)

or have a strong impact on business (e.g., they show that the level of qualified employment in a given area is extremely low, which requires a corrective action to be taken). Following [156], we call them *surprising queries*. The motivation for this template is the assumption that several users, while exploring the cube in search of significant correlations, will be "attracted" by one surprising query. So, sessions based on this template tend to converge "near" to one of the surprising queries, then they evolve casually. Note that the overall number of surprising queries is fixed by a global parameter, while each surprising query is randomly generated.

4. **Goal-Oriented**. Sessions of this type are run by users who have a specific analysis goal, but whose OLAP skills are limited so they may follow a complex path to reach their destination. All the goal-oriented sessions starting from the same seed query q end in the same (randomly-generated) query p, but the sequence of OLAP operations to be applied to reach p from q is generated randomly.

Figure 8.3 shows an intuition of sessions based on the four templates in a qualitative group-by/selection predicate space.

8.6 Experiments

To verify that the CubeLoad parameters and templates actually allow a wide spectrum of workloads to be generated, and to help benchmarkers better understand the relationships between those parameters/templates and the workload features, we use a similarity function that was specifically proposed in [7] for comparing OLAP queries and sessions. The query similarity function, σ_{que}, is a combination of three components: one related to group-by's, one to selection predicates, and one to measure sets.

Definition 8.6.1 (Similarity of OLAP queries) *Let q and q' be two queries on the same n-dimensional schema. The* similarity *between q and q' is*

$$\sigma_{que}(q,q') = 0.35\sigma_{gbs}(q,q') + 0.50 \cdot \sigma_{sel}(q,q') + 0.15 \cdot \sigma_{meas}(q,q') \in [0..1]$$

where:

- *The similarity between the group-by's of q and q', $\{l_1,\ldots,l_n\}$ and $\{l'_1,\ldots,l'_n\}$ respectively, is*
$$\sigma_{gbs}(q,q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{lev}(l_i,l'_i)}{L_i-1}}{n}$$
where L_i is the total number of levels in the i-th hierarchy, h_i, and $Dist_{lev}(l_i,l'_i) \in [0..L_i-1]$ is the distance between its two levels l_i and l'_i.

- *The similarity between the selection predicates of q and q', $\{p_1,\ldots,p_n\}$ and $\{p'_1,\ldots,p'_n\}$ respectively, is*
$$\sigma_{sel}(q,q') = 1 - \frac{\sum_{i=1}^{n} \frac{Dist_{pred}(p_i,p'_i)}{L_i}}{n}$$
where the distance $Dist_{pred}(p_i,p'_i)$ between predicates p_i and p'_i, both formulated on levels of hierarchy h_i, is 0 if they are expressed on the same level and using the same constant, 1 if they are defined on the same level but not on the same constant, greater than 1 if they are defined on different levels.

- *The similarity between the measure sets returned by q and q', $Meas$ and $Meas'$ respectively, is*
$$\sigma_{meas}(q,q') = \frac{|Meas \cap Meas'|}{|Meas \cup Meas'|}$$

The session similarity function, $\sigma_{ali}(s,s') \in [0..1]$, is based on the best alignment between the queries belonging to sessions s and s'. The best alignment is computed by means of the Smith-Waterman algorithm, which efficiently matches subsequences of two given sequences by ignoring the non-matching parts [160]. It is a dynamic programming algorithm based on a matrix whose value in position (i,j) expresses the score for aligning subsequences of s and s' that end in queries s_i and s'_j, respectively. This score is computed starting from the similarity between the queries included in the aligned subsequences [7].

Table 8.1 CubeLoad parameters used for generating the three sample workloads

Sample workload	$W1$	$W2$	$W3$
Number of profiles	1	1	1
Max number of measures	2	2	2
Size of seed query reports	$10 \div 100$	$10 \div 100$	$10 \div 100$
Number of surprising queries	5	2	1
Number of seed queries	50	5	1
Length of sessions	$7 \div 12$	$7 \div 12$	$7 \div 12$
Number of sessions per seed query	4	40	200
Year prompt fraction	0.25	0.50	1.00
Segregation predicate	No	Yes	Yes

To explore the range of possibilities of CubeLoad we generated three sample workloads with the following "extreme" features:

1. Workload $W1$ is a sparse one, i.e., the sessions generated are quite different one from another. This result is mainly obtained by using a high number of seed queries and generating a few sessions per seed.

2. Workload $W2$ is a clustered one, i.e., the sessions generated are similar to each other in five groups. This is mainly obtained by defining five seed queries.

3. Workload $W3$ is a dense one, i.e., the sessions generated are all quite similar to each other. This is mainly obtained by defining a single surprising query and by generating all sessions starting from the same seed query.

For a fair comparison, all three workloads include the same numbers of sessions (200); the values for the other parameters are summarized in Table 8.1.

A qualitative analysis of these three workloads can be made by observing Figure 8.4, that shows for each of them the session-to-session similarity. Each row and column corresponds to one of the 200 sessions of the workload, so each cell shows the similarity between two different sessions of the same workload: white means $\sigma_{ali} = 0$, black $\sigma_{ali} = 1$, gray shades mean $0 < \sigma_{ali} < 1$. As expected, in Figure 8.4.a we find a very low average similarity between sessions, while in Figure 8.4.c the average similarity is much higher. In Figure 8.4.b we can easily find the five cluster as areas with higher-than-average similarity. A quantitative confirmation of this fact can be found in Figure 8.5, that shows for each workload the average session-to-session similarity and its standard deviation: they are both lower for the sparse workload $W1$ (where all sessions are different), while they increasingly grow higher for the clustered workload

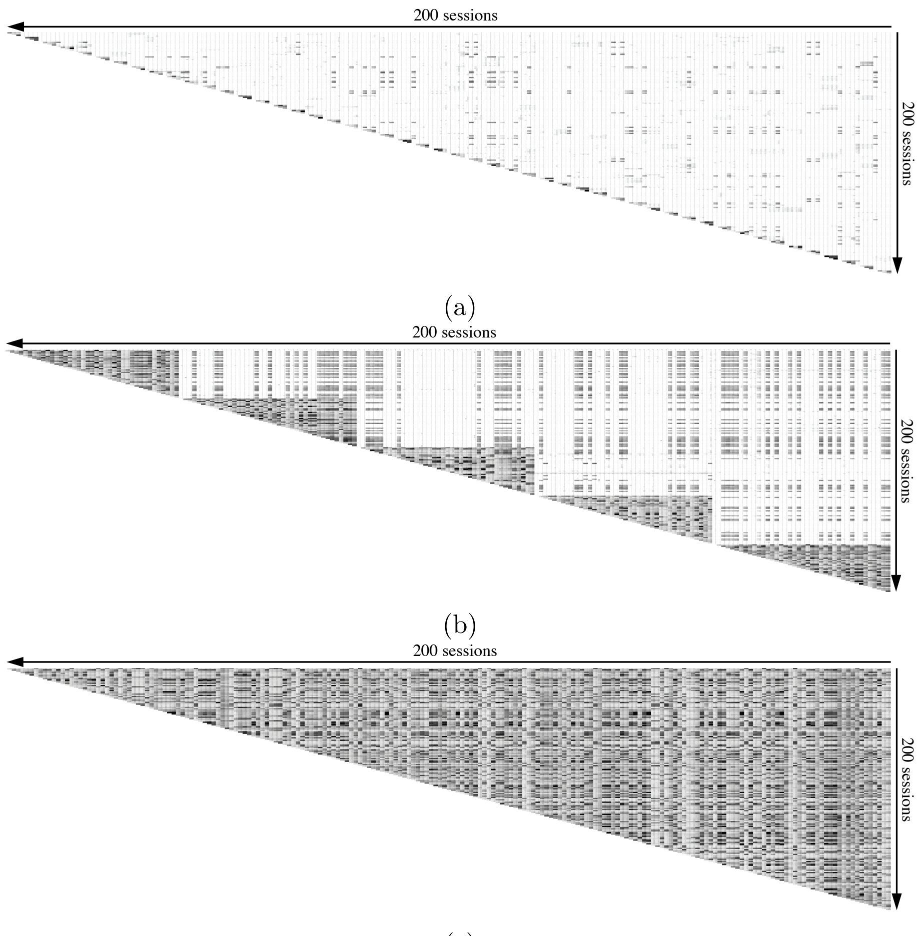


Fig. 8.4 Session-to-session similarities for the three sample workloads

$W2$ (where sessions in the same cluster are very similar to each other and very different from those in the other clusters) and the dense workload $W3$ (in the latter case, the standard deviation is high because the four templates adopted inevitably introduce a scattering in the sessions generated).

Figure 8.5 also shows the propensity of each workload to being clustered. The indicator we adopted to this end is the *Hopkins statistics* [75]. Given a workload W, i.e., a set of n sessions, we first generate a set S of m fake sessions ($m \ll n$) that are randomly and uniformly distributed in the space of possible sessions. For each fake session $s_i \in S$, let u_i be its distance from the nearest-neighbor session in W (where $Distance(s, s') = 1 - \sigma_{ali}(s, s')$). Then, m sessions are randomly chosen from W; let w_i be the distance of the i-th of these sessions from its nearest-neighbor in W. The

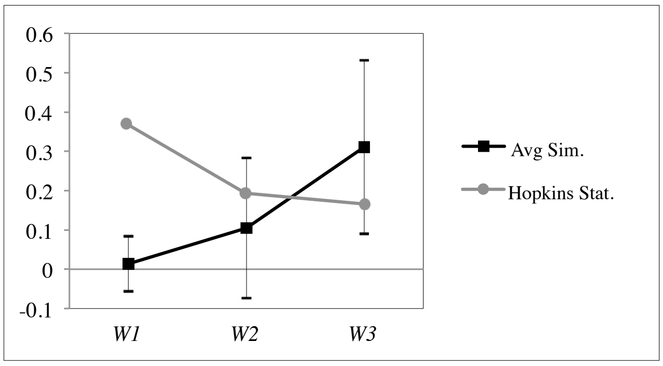


Fig. 8.5 Average session-to-session similarity and Hopkins statistics for the three sample workloads

Hopkins statistics is then defined as

$$\kappa = \frac{\sum_{i=1}^{m} w_i}{\sum_{i=1}^{m} u_i + \sum_{i=1}^{m} w_i}$$

For workload $W1$, κ is near to 0.5; this means that the distance of each session in $W1$ from its nearest-neighbor is very similar to the distance of each fake session, i.e., that $W1$ has a random distribution. For $W2$ is quite small; this is because the w_i's are small, which means that sessions are well clustered. For $W3$ κ is even smaller, because all sessions are part of a single, dense cluster.

Finally, Figure 8.6 gives a quantitative explanation of the differences between our four templates by showing the similarity σ_{que} between the first query and the subsequent queries for sessions based on each template. In the slice-and-drill template, the saw-tooth trend arises because when a sequence of slice-and-drill clicks along hierarchy h leads to a query grouped by the finest level of h, the simulated user behavior is to go back to the seed query and start a new slice-and-drill sequence along a different hierarchy (three such sequences are clearly visible in the figure). In the slice-all template, only the specific member appearing in the query selection predicate is changed during the session, so the query similarity is mostly constant and quite high. In the explorative template, the session rapidly converges towards the surprising query (the sixth query in the session in this case), then it moves randomly in the query space (in this case, it tends to reapproach the seed query). Finally, in the goal-oriented template the session randomly moves towards its goal query.

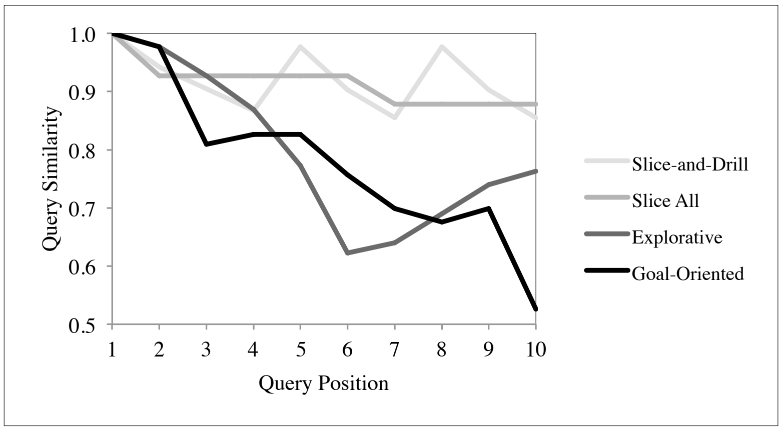


Fig. 8.6 Intra-session query similarity for the four templates

8.7 Conclusions

In this chapter we have described the features of CubeLoad, a generator of OLAP sessions aimed at simulating realistic workloads [148]. The sessions generated are currently based on four templates and ruled by a set of parameters. The template features and the impact of parameters on the resulting workload have been discussed with the support of some tests using a similarity function specifically devised for OLAP sessions.

Some comparison between CubeLoad and TPC-DS is useful at this point. Overall, the focus in the TPC-DS is more on the complexity of single queries rather than on query sessions. Indeed, while the query model is more expressive than in CubeLoad because nesting is supported, three of the four classes of queries provided in the TPC-DS (namely, *ad hoc queries*, *reporting queries*, and *data mining queries*) only include stand-alone queries; as such, they could be generated with CubeLoad by setting the maximum length of sessions to 1 and properly tuning the maximum size of seed query reports (differently from the first two classes, data mining queries are characterized by high cardinality of the results). Conversely, the class of *iterative OLAP queries* comprises four base sessions each including exactly 2 queries; more sessions can be generated from each base session by randomly changing a selection predicate. In two of the base sessions, the subsequent queries are not related by the application of a single OLAP operator like in CubeLoad, so they can be quite "distant" from each other, but still they are finalized to the same analysis goal. In the other two base sessions, the two subsequent queries differ from their selection predicate. Thus, an effective way to generate sessions like these ones with CubeLoad is to use the goal-oriented and the slice-all templates and fix the number of seed queries to 4, with a session length equal to 2.

Our future work on this topic will be mainly aimed at enhancing the capabilities of CubeLoad in three directions: (i) by allowing benchmarkers to distinguish *skilled* and *non-skilled* profiles, so as to enable a finer tuning of the workload features; (ii) by defining other templates, so as to make CubeLoad more flexible and usable for a wider array of benchmarks; and (iii) by adopting a more complex query model, so as to make the generated workloads more realistic still. From the engineering point of view, we plan to refactor the CubeLoad code according to an open architecture where each benchmarker can write her own templates in the form of a plugin.

Chapter 9

A recommendation approach for OLAP analyses based on OLAP sessions

In this chapter we propose a recommendation approach for OLAP exploration of multidimensional cubes stemming from collaborative filtering. The founding claim is that the whole sequence of queries belonging to an OLAP session is valuable because it gives the user a compound and synergic view of data; for this reason, our goal is not to recommend single OLAP queries but OLAP sessions. Like other collaborative approaches, ours features three phases: (i) search the log for sessions that bear some similarity with the one currently being issued by the user; (ii) extract the most relevant subsessions; and (iii) adapt the top-ranked subsession to the current user's session. However, it is the first that treats sessions as first-class citizens, using new techniques for comparing sessions, finding meaningful recommendation candidates, and adapting them to the current session. After describing our approach, we discuss the results of a large set of effectiveness and efficiency tests based on different measures of recommendation quality.

9.1 Introduction

OLAP is the main paradigm for accessing multidimensional data in data warehouses. As anticipated in Section 2.3, OLAP provides a set of operations (such as drill-down and slice-and-dice) that transform one multidimensional query into another, so that OLAP queries are normally formulated in the form of sequences called *OLAP sessions* [7, 155].

During an OLAP session the user analyzes the results of a query and, depending on the specific data she sees, applies one operation to determine a new query that will give her a better understanding of a business phenomenon. The resulting query sequences are strongly related to the user's skills, to the analyzed phenomenon, to the current data, and to the OLAP tool adopted. Since the huge number of possible aggregations and selections that can be operated on data may make the user experience disorientating, different approaches were taken in the literature to address this issue; in particular, in the area of personalization, both preference-based (e.g., [64, 86]) and recommendation techniques (e.g., [157, 57]) were specifically devised for OLAP systems.

In this chapter we are specifically interested in recommendations. The original claim underlying our approach is that an OLAP session issued by a user is not just a casual path aimed at leading her to a single, valuable query (the one at the end of the session). Indeed, as stated in [119] with reference to clickstream analysis, path data contain information about a user's goals, knowledge, and interests. Undoubtedly, in the case of OLAP interactions, sessions are first-class citizens: the whole sequence of queries belonging to a session is valuable in itself, because (i) it gives the user a compound and synergic view of a phenomenon; (ii) it carries more information than a single query or set of queries by modeling the user's behavior after seeing the result of the former query; and (iii) several sessions may share the same query but still have quite different analyses goals. For this reasons, like done in [68] for recommending sets of pages to users of a Web site, we propose an approach whose goal is not to recommend single OLAP queries, but rather OLAP sessions. Some existing approaches recognize that sessions carry much more information about users' behavior than queries and recommend OLAP queries based on an analysis of past sessions (e.g., [155]); still, like all the other previous work on OLAP recommendation, they are focused on suggesting only single queries to users. In this sense our approach is highly innovative in the OLAP field, and therefore it requires brand new techniques.

Consistently with collaborative filtering approaches, our goal in deciding which sessions to recommend is to reuse knowledge acquired by other users during previous sessions. So let s_{cur} be the current user session for which we have to give a recommendation, and D be the session log. The approach we propose features three phases: *alignment* (search D for sessions that are similar to s_{cur} and optimally align each of them with s_{cur}), *ranking* (extract from the selected sessions the common subsessions and rate them according not only to their degree of similarity with s_{cur}, but also to how frequent they are in D), and *fitting* (adapt the top-ranked subsession r to s_{cur} and recommend the resulting session r') [5]. To assess session similarity we readapt the alignment-based similarity function specifically devised for OLAP sessions in [7]. Note that OLAP users are normally grouped into *profiles* (e.g., CEO, marketing analyst, department manager); in this case, the session log can easily be partitioned by profiles so that each user can get recommendations based on sessions performed by users that share her background and expertise.

The qualifying properties of the recommendations we aim to give are *relevance*: this is obtained by ranking the log subsessions according to their frequencies so as to identify dense area of analysis that could be interesting for users; *foresight*: this is achieved by allowing even a subsession that is "far" from the alignment point with the current session (i.e., intuitively, far in the future) to be recommended; *novelty*: thanks to the fitting phase, the recommendation we give may not be part of the log; and *suitability*: during the fitting phase, the top-ranked subsession found in the log is adapted to the current session.

The outline of the chapter is as follows.

- In Section 9.2 we discuss the related literature.
- In Section 9.3 we provide the formal background and introduce the working example.
- In Section 9.4 we present our approach to recommendation and describe its phases.
- In Section 9.5 we discuss the results of experimental tests.
- In Section 9.6 we draw the conclusions.

9.2 Related Literature

Recommender systems [4] are now well established as an information filtering technology and used in a wide range of domains. They are traditionally categorized as either *content-based* (suggestions are based on the user's past actions only), *collaborative* (suggestions are based on similarities between users), or hybrid combinations.

A comprehensive survey of collaborative recommender systems evaluation is presented in [71]. Recommender systems are mostly evaluated with accuracy-based measures [67], typically predictive accuracy like MAE, or classification accuracy like precision and recall.

Accuracy alone fails to capture the usefulness of recommendations, so other objective metrics related to suitability are being developed.

For instance, coverage [71, 55] is the degree to which recommendations cover the set of available items and the degree to which recommendations can be generated to all users. Novelty directly measures non-obviousness, often by referring to a fixed list of obvious recommendations, while serendipity measures how far novel recommendations may positively surprise users, for instance by reporting the rate of useful novel recommendations.

Recently recommender systems started to gain interest in the database community, with approaches ranging from content-based [36] to collaborative [38] query recommendation, especially to cope with the problem of interactively analyzing database instances [94, 92]. This problem is particularly important in a data warehousing context, where one prominent use of such systems is to analyze the warehouse instance with OLAP queries as a basis for decision support.

In a data warehouse context, the peculiarities of the multidimensional schema and queries can be leveraged. In [157], operators are proposed to analyze a query answer by automatically navigating to more detailed or less detailed aggregation levels, in order to detect surprising values to recommend. In [155], the log is represented as two Markov models: one that describes the transitions between query patterns (where a pattern is a simplified query expression) and one that describes transitions between selection predicates. These two models are used to construct the most probable query that follows the current query.

More recently, [86] proposes a content-based recommendation approach that synthesizes a recommendation by enriching the current query answer with elements extracted from a user's profile. [57] introduces a generic framework for query recommendation, where a distance measure between sessions is used to compare log sessions with the current session, and the set of final queries belonging to the closest log sessions are recommended and ranked according to their distance with the final query of the current session. [58] proposes to recommend a graph of former queries, based on the application of the operators of [157] on the query log, to detect surprising values regarding the current query answer. In [9], queries are recommended using a probabilistic model of former sessions, inspired by [155], where a similarity tailored for OLAP queries is used to group queries.

The existing approaches for query recommendation in data warehouses are summarized in Table 9.1 where, for each approach, we indicate (1) the category of the recommender system: content-based, collaborative filtering, or hybrid; (2) the source of information, i.e., whether the approach is log-driven, answer-driven, or both; (3) whether the approach is session-based and, if so, whether sequential aspects are con-

Table 9.1 Query recommendation approaches in data warehouses

Ref.	Cat.	Input				Output			Quality
		Source	Session?	Model	Tech.	Form	Source	Tech.	metrics
[157]	CB	ans.	not seq.	ans.	stoch.	tuples	instance	sel.	acc.
[155]	CF	log	seq.	expr.	stoch.	query	curr.	synth.	-
[57]	H	log	seq.	ans.	sim.	query	log	sel.	acc.
[86]	CB	ans.	no	ans.	pref.	query	curr.	synth.	-
[58]	H	log/ans.	not seq.	ans.	stoch.	query	log	sel.	acc.
[9]	CF	log	seq.	expr.	stoch.	query	log	sel.	acc., cov.
Ours	CF	log	seq	expr.	sim.	session	log/curr.	synth.	acc., cov., nov., fore.

sidered or not; (4) the query model used, i.e., whether the approach leverages query expressions or query answers; (5) the technique used to process the input, i.e., whether the approach is similarity-based, preference-based, or stochastic; (6) the form (query or tuples) of the recommendation; (7) whether the recommendation is taken from a log, from the database instance, or from the current query; (8) the technique used to output the recommendation, i.e., if it is simply selected from the source or if it is synthesized from it; and finally (9) the metrics used for assessing the quality of recommendations: accuracy, coverage, novelty, foresight.

The first lesson learned from this literature review is that sessions are rarely treated as first-class citizens. Sequential aspects are seldom taken into account, and no approach ever considered recommending a sequence of queries. Approaches taking sessions into account only use them as input, to construct a model subsequently used for recommending. In addition, the stochastic approaches that consider sessions use a first order Markov Model, and therefore base their predictions only on the user's current query. Recommendation can be too much prescriptive (only one query) or too little prescriptive (a graph of queries). Besides, recommendations are rarely synthesized queries, but more often queries chosen among past queries stored in some query log or tuples retrieved from the database instance. Many of the techniques proposed rely on query answers, which may result in a poor scalability compared to techniques using only query expressions, as reported in [38] in the case of databases. Finally, none of the former approaches assesses the system quality beyond accuracy and coverage. The approach we propose in this article overcomes these limitations. Sessions are treated as first-class citizen all along the process, query expressions are leveraged with a similarity tailored for OLAP sessions, the recommendation is synthesized from the log and the current session, and the quality of the recommender system is assessed using suitability metrics.

9.3 Background

In this chapter we start from Definitions 1, 3 and 1 of multidimensional schema, group-by set and cube (provided in Section 2.4) to define OLAP queries and sessions. The basic form of an OLAP query, centered on a single cube, is characterized by a group-by set, a selection expressed by a conjunctive predicate, and a set of measures.

Definition 24 (OLAP Query) *An* OLAP query *on a cube C is a triple $q = \langle b, P, M \rangle$ where:*

1. *$b \in Dom(B)$ is a group-by set;*
2. *P is a set of* selection predicates*, i.e., boolean clauses of type $l_k = value$ where $l_k \in L$*
3. *M is a set measures such that $M \subseteq Meas$.*

An OLAP query q can be decomposed into fragments*, where each fragment is either (i) a level $l \in b$, or (ii) a predicate $p \in P$, or (iii) a measure $m \in M$.*

Example 26 *An example of query on the multidimensional schema in Figure 2.3 is $q = \langle$(Product, Month, Country), (Country=Italy), (Unit sales, Profit)$\rangle$.*

An OLAP session is an ordered sequence of correlated queries formulated by a user on a multidimensional schema; each query in a session is typically derived from the previous one by applying an OLAP operator (such as roll-up, drill-down, and slice-and-dice).

Definition 25 (OLAP Session) *An* OLAP session *is a sequence $s = \langle q_1, q_2, \ldots \rangle$ of queries on schema $\mathcal{M}$. A* log *D is a set of sessions.*

Moreover, given a session s, we will denote with $length(s)$ the number of queries in s, with $s[w]$ ($1 \leq w \leq length(s)$) the w-th query of s, and with $s[v, w]$ ($1 \leq v \leq w \leq length(s)$) the subsession of s spanning from its v-th query to the w-th one. The last query of s, $s[length(s)]$, is briefly denoted with $s[.]$, so $s[v, .]$ is the subsession of s spanning from its v-th query to the end.

The reference database used in this chapter is IPUMS, a public database storing census microdata for social and economic research [116]. Its CENSUS multidimensional schema has five hierarchies, namely RESIDENCE, TIME, SEX, RACE, and OCCUPATION, and several measures. The complete roll-up orders are shown in Figure 9.1.

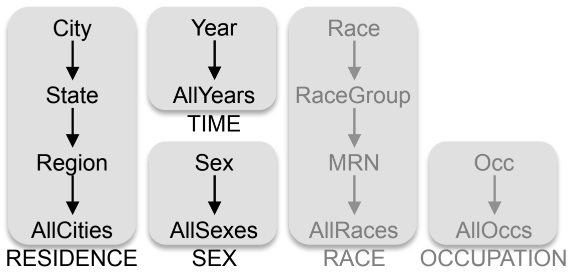
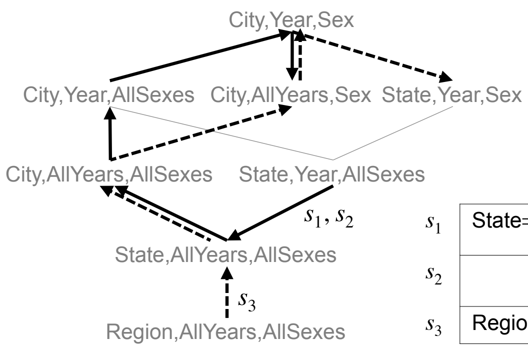


Fig. 9.1 Roll-up orders for the five hierarchies in the CENSUS schema



	P	*M*
s_1	State=TN	AvgCostGas, SumIncTot
s_2	–	AvgPerWt, AvgCostGas
s_3	Region=ENC	AvgPerWt

Fig. 9.2 A log for our working example

In the following sections, we will use a simplified version of the CENSUS schema, featuring only the RESIDENCE, TIME, and SEX, while the experimental tests in Section 9.5 will be carried out on the complete schema. All examples will be based on a log that consists of 3 sessions, s_1, s_2, and s_3, each including 6 queries. For simplicity, predicates and measures are not changed during each session, while group-by's are changed as shown in Figure 9.2 with reference to a portion of the multidimensional lattice of the CENSUS schema.

9.4 Our Approach

Let s_{cur} be the current OLAP session and D be a log of past sessions. As sketched in Figure 9.3, our approach to compute a recommendation for s_{cur} based on D includes three consecutive phases, described in detail in the following subsections:

1. *Alignment*, described in Section 9.4.1, that selects from D a set of sessions that are similar to s_{cur} by finding an optimal alignment between each of them and s_{cur}. For each session d in this set, its *future* is defined as the subsession of d following the query that has been aligned with the last query of s_{cur} (i.e., with

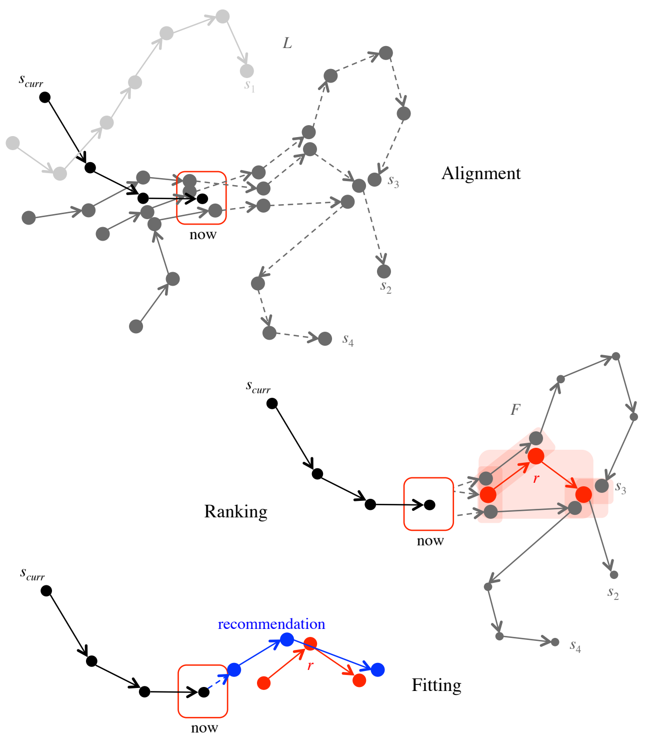


Fig. 9.3 The three recommendation phases

the last query currently issued by the user). The set of futures obtained in this way constitutes the set F of *candidate recommendations* for s_{cur}.

2. *Ranking*, detailed in Section 9.4.2, that chooses a *base recommendation* r as a subsession of a candidate recommendation in F by considering both its similarity with s_{cur} and its frequency in D.

3. *Fitting*, described in Section 9.4.3, that adapts r to s_{cur} by looking for relevant patterns in the query fragments of s_{cur} and r and using them to make changes to the queries in r, so as to deliver a recommendation r'.

Noticeably, the user has a possibility of influencing the recommendation by acting on a parameter called *minimum foresight*, denoted δ, used in the ranking phase to select the base recommendation. With this threshold, the user indicates how distant from the current query the recommended session should be: higher values result in recommendations being farer from, and less strictly related to, the current session. Automatically adjusting this value based on the history of the user interactions with the system is beyond the scope of this work and is left as future work.

9.4.1 Alignment

The goal of this phase is to identify in the log D a set F of candidate recommendations for the current session s_{cur}. To do so, first we try to find an alignment between s_{cur} and each session d in D. The alignment algorithm we adopt is an adaptation of the one that was proposed in [7] to effectively capture the similarities between OLAP sessions; in turn, that algorithm is based on the Smith-Waterman algorithm [160], whose goal is to efficiently find the best alignment between subsequences of two given sequences by ignoring their non-matching parts.

The Smith-Waterman algorithm is a dynamic programming algorithm that computes the optimal alignment between two sequences s and s' based on a score matrix. In position (v, w), this matrix reports a score that expresses how well s and s' match when they are aligned ending in elements $s[v]$ and $s'[w]$; each score is recursively calculated by progressively adding the similarities between all pairs of matching elements in the two sequences (the similarity between two elements can also be negative, to express that they do not match). Intuitively, the algorithm seeks an optimal trade-off between the cost for introducing a gap in the matching subsequences and the cost for including a poorly matching pair of elements. In the extension proposed in [7] for OLAP sessions, sequence elements correspond to queries. A query similarity function, $\sigma_{que} \in [0..1]$, is defined as a combination of three components: one related to group-by's (based on how distant the two group-by's are in the multidimensional lattice), one to selection predicates (based on the distance of the levels on which predicates are formulated), and one to measure sets (their Jaccard index). A similarity threshold is then used to distinguish matches from mismatches. Besides, a time-discounting function is introduced to promote alignments based on recent queries, and a variable gap penalty is used to discourage discontinuous alignments. The output of the alignment algorithm when applied to two sessions s and s' is their best *alignment* a, defined by two matching subsessions $s[v_{start}, v_{end}]$ (denoted $s^{(a)}$) and $s'[w_{start}, w_{end}]$ (denoted $s'^{(a)}$). If an alignment between s and s' is found we say that s and s' are *similar*, denoted $s \sim s'$, and their similarity (computed as explained in [7]), is denoted with $\sigma_{ses}(a) \in]0..1]$. Otherwise it is $s \not\sim s'$.

For our alignment phase we use a variant, denoted SW_{cand}, that promotes alignments between the end of s_{cur} and the beginning of each log session d, so as to favor log sessions capable of providing a "long" candidate recommendation. This is achieved by modifying the two-dimensional logistic sigmoid function originally used as a time-discounting function as defined below and shown in Figure 9.4:

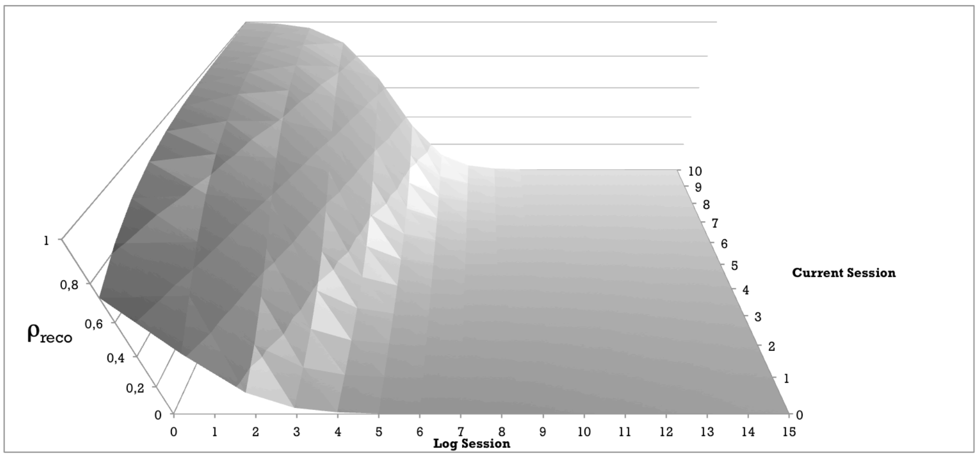


Fig. 9.4 The sigmoid function used for SW_{cand} ($|s_{cur}| = 10$, $|l| = 15$, $\gamma_{min} = 0$)

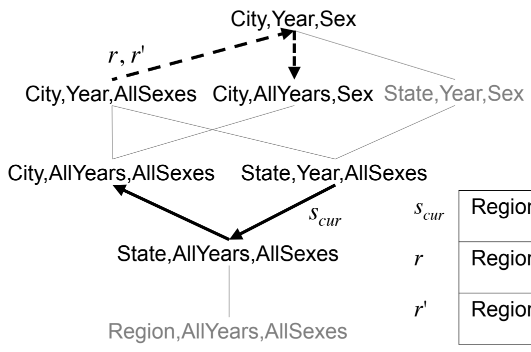
Algorithm 4 Alignment

```
Input $s_{cur}$: the current session, $D$: the log
Output $F$: set of candidate recommendations
1: $F \leftarrow \emptyset$                                              ▷ Initialize $F$
2: for each $l \in L$ do
3:     $a \leftarrow SW_{cand}(s_{cur}, l)$                               ▷ Try to align $d$ with $s_{cur}$
4:     $v \leftarrow$ position in $d$ of $l^{(a)}[.]$
5:     if $l \sim s_{cur}$ and $s^{(a)}_{cur}[.] = s_{cur}[.]$ and $l[v+1,.] \neq \emptyset$ then   ▷ An alignment is found ending in $s_{cur}[.]$
6:         $f \leftarrow l[v+1,.]$                                        ▷ Find the candidate recommendation
7:         $F \leftarrow F \cup \{f\}$
8: return $F$
```

$$\gamma_{cand}(v, w) = 1 - \frac{1 - \gamma_{min}}{1 + e^{\frac{-20}{|l|}w + \frac{5}{|s_{cur}|}v + \frac{10}{|s_{cur}|}}} ,$$

where v is a position in s_{cur}, w is a position in d, and γ_{min} is the minimal value desired for γ_{cand} (i.e., the minimal weight given to query alignments considered as irrelevant). The constants have been experimentally tuned in order to answer to the specific desired behavior: $\frac{-20}{|l|}$ defines the proportion of queries in d whose alignment with the last queries of s_{cur} has to be favored; $\frac{5}{|s_{cur}|}$ defines the proportion of queries in s_{cur} whose alignment with the first queries of s has to be favored; $\frac{10}{|s_{cur}|}$ defines a minimal weight to consider between the first queries of s_{cur} and d.

The pseudocode for the whole alignment process is given in Algorithm 4. For each log session d such that $d \sim s_{cur}$ (line 2), its *future* is determined as the subsession $f = d[v+1,.]$ (line 6) where v is the position of the last query aligned (line 4). If the last query aligned in s_{cur} is $s_{cur}[.]$, i.e., the last query in s_{cur}, and d has a non-empty future f (line 5), then f is added to the set F of candidate recommendations (line 7).



	P	M
s_{cur}	Region=Mountain	AvgPerWt, SumIncTot
r	Region=MA	AvgPerWt, AvgCostGas
r'	Region=Mountain	SumIncTot, AvgPerWt

Fig. 9.5 A current session, its aligned log subsessions, its candidate recommendations, its base recommendation, and its recommendation

Example 27 *With reference to the log of Figure 9.2, let the current session s_{cur} be the one whose group-by's, selection predicate, and measure set are shown in Figure 9.5. An alignment is found between s_{cur} and each session in the log. In particular, s_{cur} is aligned with log subsessions $s_1[1,3]$ (with similarity 0.15), $s_2[1,3]$ (similarity 0.21), and $s_3[2,3]$ (similarity 0.29); in the last case, similarity is higher (though the matching subsession is shorter) because the component of query similarity related to measure sets is higher. So, in this example the set of candidate recommendations is $F = \{s_1[4,6], s_2[4,6], s_3[4,6]\}$, obtained by considering the futures of the aligned log subsessions.*

9.4.2 Ranking

The goal of this phase is to examine F to determine the most suitable base recommendation r, which will then be refined in the fitting phase. Consistently with collaborative filtering approaches, we identify the densest areas in F so as to define r as the most relevant subsession in F. More precisely, this phase requires first to compute pairwise alignments between all sessions in F, and to use those alignments to assign a relevance score to each query $q \in f_i$, for each $f_i \in F$. Then, a relevance score is computed for each subsession that has been successfully aligned in each f_i by averaging the scores of its queries. Finally, r is chosen as the subsession with the highest relevance among those that meet the minimum foresight constraint δ set by the user. Note that the ranking methods normally used in collaborative filtering approaches can hardly be adapted to the context of databases [38], and even less in our context because (i) we work with two different levels of aggregation: queries and sessions; and (ii) we compare objects (queries in our case) in terms of similarity and not of identity.

Algorithm 5 Ranking

```
Input $s_{cur}$: the current session, $F$: set of candidate recommendations, $\delta$: minimum foresight
Output $r$: base recommendation
Variables: $A_{ij}$: sets of alignments
 1: for each $f_i \in F, q \in f_i$ do                                    ▷ Initialize query scores
 2:     $q.relevance \leftarrow 0$
 3: $maxRelevance \leftarrow 0$
 4: for each $f_i \in F$ do
 5:     for each $f_j \in F, j > i$ do
 6:         $A_{ij} \leftarrow SW_{rank}(f_i, f_j)$                         ▷ Compute pairwise alignments...
 7:         for each $a \in A_{ij}$ do                                  ▷ ...and update query scores
 8:             for each $q \in f_i^{(a)}$ do
 9:                 $q.relevance \leftarrow q.relevance + \sigma_{ses}(a)$
10:             for each $q \in f_j^{(a)}$ do
11:                 $q.relevance \leftarrow q.relevance + \sigma_{ses}(a)$
12:     for each $j \neq i, a \in A_{ij}$ do
13:         $avgRelevance \leftarrow \frac{\sum_{q \in f_i^{(a)}} q.relevance}{length(f_i^{(a)})}$        ▷ Compute score for $f_i^{(a)}$
14:         $a' \leftarrow SW_{rank}(f_i^{(a)}, s_{cur})$                ▷ Align $f_i^{(a)}$ with $s_{cur}$ to compute foresight
15:         if $avgRelevance > maxRelevance$ and $(1 - \sigma_{que}(f_i^{(a)}[.], s_{cur}[.])) \cdot (1 - \sigma_{ses}(a')) \geq \delta$ then
16:             $maxRelevance \leftarrow avgRelevance$
17:             $r \leftarrow f_i^{(a)}$
18: return $r$                                   ▷ Return the subsession with the highest score
```

The pseudocode for this phase is sketched in Algorithm 5. The *for* loop starting at line 4 aims at finding, for each candidate recommendation f_i, its subsession yielding the highest relevance score. This is done by first computing the pairwise alignments between f_i and all the other sessions in F making use of a different version of the alignment algorithm. In this version, called SW_{rank}, no time-discounting function is used (as we do not need to favor alignments in particular positions), and every possible alignment is returned; so, differently from SW_{cand}, the result of $SW_{rank}(f_i, f_j)$ is not only the best alignment but a set of alignments A_{ij} between f_i and f_j. For each alignment $a \in A_{ij}$, in lines 7-11 we increase the scores of all aligned queries in f_i and f_j by the similarity $\sigma_{ses}(a)$. Eventually, the queries with the highest scores will be marking the densest areas in F, i.e., those that have been traversed the most by the sessions in F. Then, in lines from 12 to 17, the query scores are used to compute a score for each subsession of f_i corresponding to an alignment found with another session in F. In particular, for subsession $f_i^{(a)}$ aligned in a, its relevance score is computed as the average score of its queries. To check that the constraint on δ is met, the foresight of $f_i^{(a)}$ is estimated in line 15 as the distance between its last query and the last query of the current session, weighted on the distance between the whole $f_i^{(a)}$ and s_{cur}.

Example 28 *With reference to our working example, the subsession in F yielding the highest score (0.32) is* $s_2[4, 6]$*, which becomes the base recommendation r (see Figure 9.5).*

9.4.3 Fitting

The goal of this phase is to adapt the base recommendation r to the current session s_{curr}, i.e., to move r closer to s_{curr} while preserving its intrinsic logic. This is achieved by characterizing (i) the differences between s_{cur} and its aligned counterpart in the log session d from which r is extracted and (ii) the user's behavior during s_{cur}. These characterizations adapt the technique of [6], that is itself inspired by *label ranking*, a form of classification that has been shown to be effectively handled by association rules. In [6] we proposed to modify a query using association rules extracted from a query log. Our fitting phase therefore consists of extracting association rules from s_{cur} and d. Two types of rules, called Type-1 rules and Type-2 rules respectively, are extracted and used to transform r as sketched in Algorithm 6. Type-2 rules are those introduced in [6] and have been proved successful in a similar context, while Type-1 rules are novel and ensure that the final recommendation remains focused on the fragments frequently used in the current session. The main difference between the two types is the sessions they are computed from, which impacts the form of the rules.

A Type-1 rule aims at establishing a correlation between a query fragment x (either a measure, a group-by level, or a selection predicate) used in d and a query fragment y of the same type used in s_{cur}, so that r can be transformed by substituting all occurrences of x with y. These rules take the form $x \rightarrow y$ and are extracted from couples formed by a query q_i of s_{cur} and q'_j, the query of d aligned with q_i (line 1 of Algorithm 6). For instance, session d may be characterized by a recurrent selection predicate $\mathsf{Year} = 2013$ and be focused on measure $\mathsf{AvgCostGas}$, while session s_{cur} may be characterized by $\mathsf{Year} = 2011$ and measure $\mathsf{AvgPerWt}$; in this case, two rules $(\mathsf{Year} = 2013) \rightarrow (\mathsf{Year} = 2011)$ and $\mathsf{AvgCostGas} \rightarrow \mathsf{AvgPerWt}$ will be extracted aimed at making the base recommendation r more similar to the current session by focusing r on 2011 and $\mathsf{AvgPerWt}$ rather than on 2013 and $\mathsf{AvgCostGas}$.

Type-2 rules aim at finding query fragments used frequently together in s_{cur} (line 2), and have the form $X \rightarrow y$ with X a set of fragments of s_{cur} and y a fragment of s_{cur}. For instance, rule $\{\mathsf{AvgCostGas}, (\mathsf{Year} = 2011)\} \rightarrow \mathsf{Region}$ captures the fact that the trends of measure $\mathsf{AvgCostGas}$ for 2011 are mainly analyzed in s_{cur} on a region-by-region basis. If in r the same measure for 2011 is analyzed by State instead, this rule can be used to adapt r by changing query group-by's from State to Region. Noticeably, letting both types of rules potentially work with *all* query fragments ensures that the full range of transformations between OLAP sessions are covered.

Rules of both types are ranked according to the geometric mean of the following figures, which have been experimentally selected: (i) the support of the rule; (ii) the confidence of the rule; (iii) the average position in s_{cur} where the head fragment y appears (to favor recent fragments); (iv) the support of the head fragment y in s_{cur} (to favor frequent fragments). Note that, for Type-1 rules, support is given by $supp(x \rightarrow y) = \frac{|\{(q_i,q'_j) \text{ s.t. } q_i \in s_{cur}, q'_j \in d, x \in q'_j, y \in q_i\}|}{|s_{cur}|}$, while for Type-2 rules it is $supp(X \rightarrow y) = \frac{|\{q_i \text{ s.t. } q_i \in s_{cur}, X \cup \{y\} \subseteq q_i\}|}{|s_{cur}|}$. The confidence of a rule $X \rightarrow Y$ (where X and Y are sets of fragments) is $conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)}$.

The rules extracted are used to change the fragments Fr shared by all the queries of the base recommendation r (line 3 of Algorithm 6). This ensures that the fitting process respects the intrinsic logic of r, without producing two identical queries in the recommended session. First, Type-1 rules $x \rightarrow y$ are considered (lines 5 to 11), in descending order (line 6), as follows. If fragment x exists in r and not in s_{cur}, then this fragment is replaced by y in r. Every modified fragment is marked so as not to be adapted any more (line 10). Then, Type-2 rules $X \rightarrow y$ are considered (line 12 to 22), in descending order (line 13), only for changing the queries that include the rule body X (line 14), as follows. If the rule head y is a selection predicate or a level and is not already present in the query (line 15), then it replaces the corresponding fragment (i.e., the one belonging to the same hierarchy) of the query (line 17). If y is a measure that is not already present in r (line 19), then it is added to the measure set of the query (line 21). Like for Type-1 rules, once a fragment is modified, it is marked so it is not modified any more (line 21).

Note that, as an effect of the fitting phase, there is a possibility that the recommendation produced, r', includes some queries that are identical to queries that were part of s_{cur} (i.e., queries that the user has already formulated during the current session). Such a recommendation would be completely useless. So, in this case, we go back to the ranking phase and take as a base recommendation the next most relevant subsession.

Example 29 *In our working example, the only applicable rules are the Type-1 rules* AvgCostGas $\rightarrow$ SumIncTot *and* (Region =MA) $\rightarrow$ (Region =Mountain)*, which transform r into r' by changing its measure set and its selection predicate as shown in Figure 9.5.*

9.5 Experimental Results

This section describes the results of the experimental tests we performed. After explaining the measures we use to assess the quality of a recommendation, we report and discuss the test results from both the effectiveness and efficiency points of view. The benchmark we adopted for our tests is based on a set of synthetic logs over

Algorithm 6 Fitting

```
Input $s_{cur}$: the current session, $r$: the base recommendation, $d$: the log session from which $r$ was extracted
Output $r'$ : the recommended session
 1: $T_1 \leftarrow ExtractType1Rules(s_{cur}, l)$                                   ▷ Extract rules
 2: $T_2 \leftarrow ExtractType2Rules(s_{cur})$
 3: $Fr \leftarrow \bigcap_{q \in r} q$                                              ▷ Set of shared fragments in $r$
 4: $r' \leftarrow r$
 5: while $Fr \neq \emptyset$ and $T_1 \neq \emptyset$ do                            ▷ Apply Type 1 rules
 6:     $(x \to y) \leftarrow TopRank(T_1)$
 7:     if $x \in Fr$ and $x \notin q\, \forall q \in s_{cur}$ then
 8:         for $i = 1$ to $length(r')$ do
 9:             $r'[i] \leftarrow r'[i] \setminus \{x\} \cup \{y\}$
10:         $Fr \leftarrow Fr \setminus \{x\}$
11:     $T_1 \leftarrow T_1 \setminus \{x \to y\}$
12: while $Fr \neq \emptyset$ and $T_2 \neq \emptyset$ do                            ▷ Apply Type-2 rules
13:     $(X \to y) \leftarrow TopRank(T_2)$
14:     if $X \subseteq Fr$ then
15:         if $y$ is a group-by level or a selection predicate and $\exists z \in Fr$ corresponding to $y$ then
16:             for $i = 1$ to $length(r')$ do
17:                 $r'[i] \leftarrow r'[i] \setminus \{z\} \cup \{y\}$
18:             $Fr \leftarrow Fr \setminus \{z\}$
19:         else if $y \notin q\, \forall q \in r'$ then
20:             for $i = 1$ to $length(r')$ do
21:                 $r'[i] \leftarrow r'[i] \cup \{y\}$
22:     $T_2 \leftarrow T_2 \setminus \{X \to y\}$
23: return $r'$
```

the CENSUS schema. The sessions in each log were generated using *CubeLoad*, the benchmark generator presented in Chapter 8.

9.5.1 Assessing the Quality of Recommendations

Consistently with the literature on recommender systems (see Section 9.2), we assess the accuracy and coverage of our recommender system first, and then use more elaborated measures to assess the suitability of recommendations.

Accuracy is measured by extending the classical precision and recall measures to take session similarity into account. Let D be a log and S be a set of current sessions for which recommendations are to be computed. Given session $s \in S$, let f_s be its actual future (i.e., the sequence of queries the users would have formulated after $s[.]$ if she had not been given any recommendation) and r_s be the future recommended by our system. Recommendation r_s is considered to be *correct* when $r_s \sim f_s$, i.e., when it is similar to the actual future of s.

Let $FS = \{f_s \,;\, s \in S\}$ and $RS = \{r_s \,;\, s \in S\}$. The set of true positives is then defined by

$$TP = \{r_s \in RS \,;\, r_s \sim f_s\}$$

i.e., the set of recommended futures similar to their actual counterparts. The set of false positives is $FP = RS \setminus TP$ and the set of false negatives is $FN = \{f_s \in FS\,;\ f_s \not\sim r_s\}$. Then, *recall* is $\frac{|TP|}{|TP|+|FN|}$ and *precision* is $\frac{|TP|}{|TP|+|FP|} = \frac{|TP|}{|RS|}$.

The global accuracy is measured using the *F-measure* [169]:

$$F = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

while the *coverage* is $\frac{|RS|}{|S|}$.

To assess the suitability of a recommendation r_s for a current session s we adopt two more measures. *Foresight*, measured like in Section 9.4.2, indicates how "far" from the current query of s the last query of r_s is, weighted by the distance between the two sessions:

$$Foresight(s) = (1 - \sigma_{que}(s[.], r_s[.])) \cdot (1 - \sigma_{ses}(a))$$

(where a is the best alignment between s and r_s, and $\sigma_{ses} = 0$ if no alignment can be found). *Novelty* indicates how distant r_s is from the sessions in the log:

$$Novelty(s) = min_{l \in L}(1 - \sigma_{ses}(a'))$$

(where a' is the best alignment between d and r_s).

9.5.2 Effectiveness Tests

This section presents the results of the seven groups of tests we conducted to assess the accuracy and coverage of our recommender system, as well as the suitability of the generated recommendations. All tests were conducted on a 64-bits Intel Core i7 quad-core 3.4GHz, with 8GB RAM, running Windows 7 pro SP1.

A set of tests on a log D generated by CubeLoad is executed by iteratively (i) picking one session $d \in D$; (ii) taking subsession $d[1,4]$ as the current session and subsession $f_d = d[5,.]$ as its actual future (except for the third group of tests, where the position of the current query is varied); (iii) finding a recommendation r_d for $d[1,4]$ using the remaining sessions, $D \setminus \{d\}$, as the log sessions. As explained in Section 9.5.1, r_d is considered to be correct when $r_d \sim f_d$, non correct otherwise.

The aim of the first group of tests we carried was to tune our approach, i.e., to determine a good trade-off between accuracy and coverage. To this end we recall from Section 9.4.2 that the base recommendation is chosen, among the candidate recommendations, as the one with the highest relevance score; this suggests to check

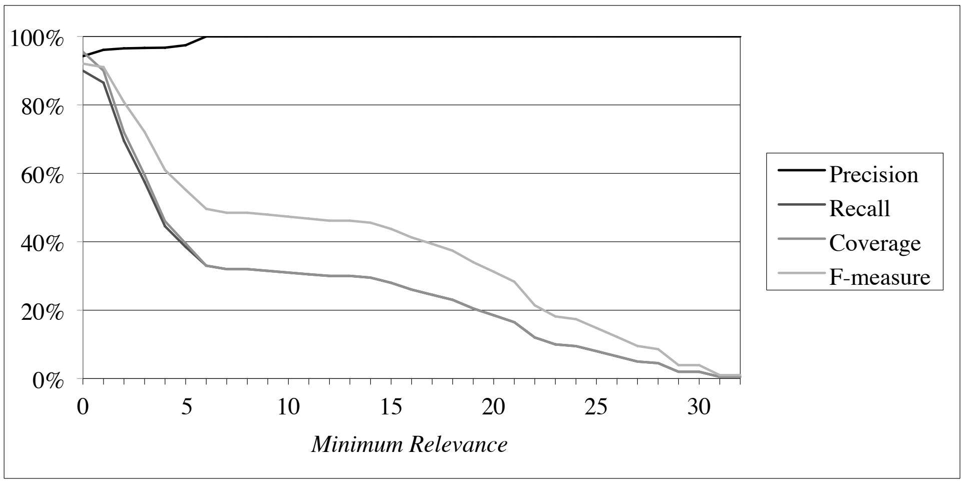


Fig. 9.6 Accuracy vs. coverage trade-off in function of the base recommendation minimum relevance

how often selecting a subsession with low relevance leads to a wrong recommendation. So in this tests we gave a recommendation only if the base recommendation had relevance higher than a *minimum relevance* threshold, otherwise we gave no recommendation. The results in Figure 9.6 show how the accuracy and coverage change when the minimum relevance increases, on a log of 200 sessions. When the minimum relevance is 0 no filtering of base recommendations is made, so coverage is about 90%. Expectedly, precision increases with the minimum relevance, while coverage —and therefore recall— decrease quite rapidly, meaning that base recommendations often have low relevance. However, the curve for precision clearly shows that our approach is well capable of delivering good recommendations even out of base recommendations with low relevance. These facts are well summarized by the F-measure curve, showing that the best overall performance is achieved when the minimum relevance is 0. Therefore, no minimum relevance threshold was posed for all the following tests.

The focus of the second group of tests was to observe how the approach performs on logs with different characteristics. To this purpose, we tuned CubeLoad parameters to create two series of logs: the first one with different densities, the second one with different session lengths. Note that a *clustered* log is meant as one with dense groups of similar sessions (specifically, each session is similar to about 30 other sessions on average), whereas in a *sparse* log all sessions tend to be dissimilar from each other (each session is similar to about 15 other sessions on average). The minimum foresight δ was set to 0. As shown in Figure 9.7 (top left), the coverage increases as sessions get more clustered, while precision is not significantly affected; this behavior

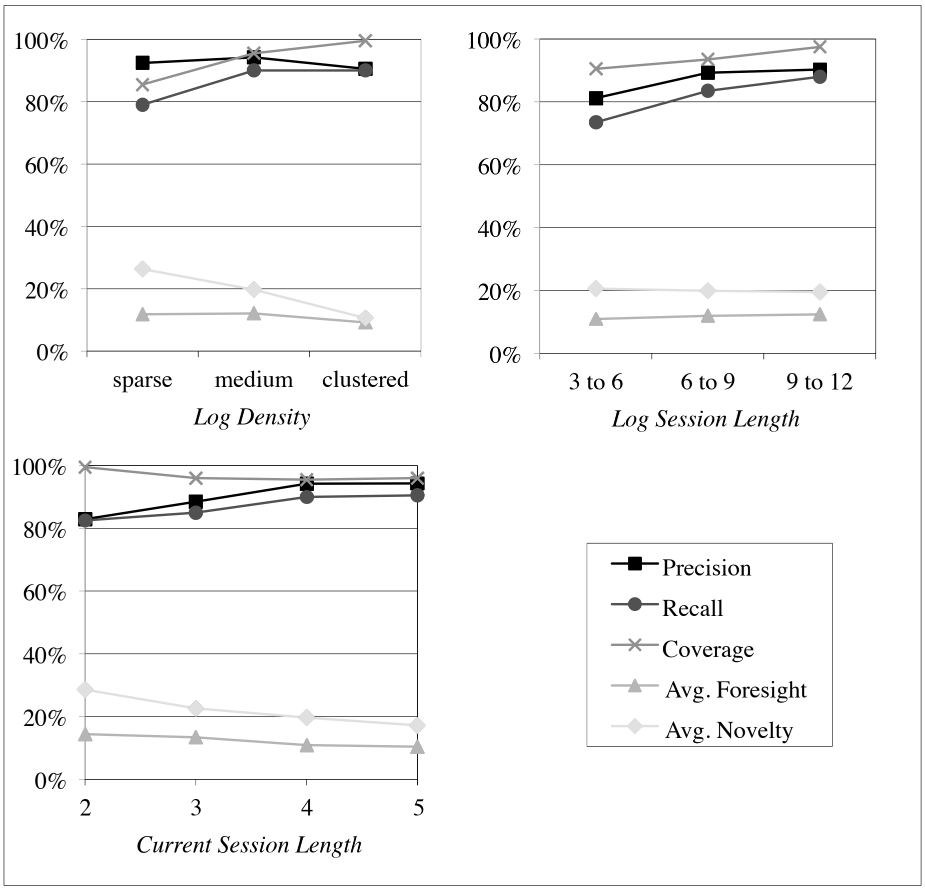


Fig. 9.7 Effectiveness vs. log and session features

is consistent with that of collaborative filtering approaches, where the capability of giving a recommendation depends on the quantity of data that matches the user's query. Also, Figure 9.7 (top right) shows that it is hard to give good recommendations when log sessions are short; indeed, the shorter the log sessions, the less likely for the recommendation to be similar to the (even shorter) actual future —therefore, the lower the precision. As to the average foresight, in these tests it is expectedly low because $\delta = 0$. Finally, the average novelty is relatively higher but still it does not exceed 0.3, again as a consequence of having set $\delta = 0$; the reason for this relationship between novelty and δ will be explained below (see comments to the fourth group of tests).

The core question of the third group of tests was: *which is the best time in the user's OLAP journey to give her a recommendation?* In other words, we analyzed how the recommendation effectiveness changes with the length of the current session on a log with medium density and medium length of sessions (again with $\delta = 0$). The results in Figure 9.7 (bottom) show that increasing the length of current sessions has a

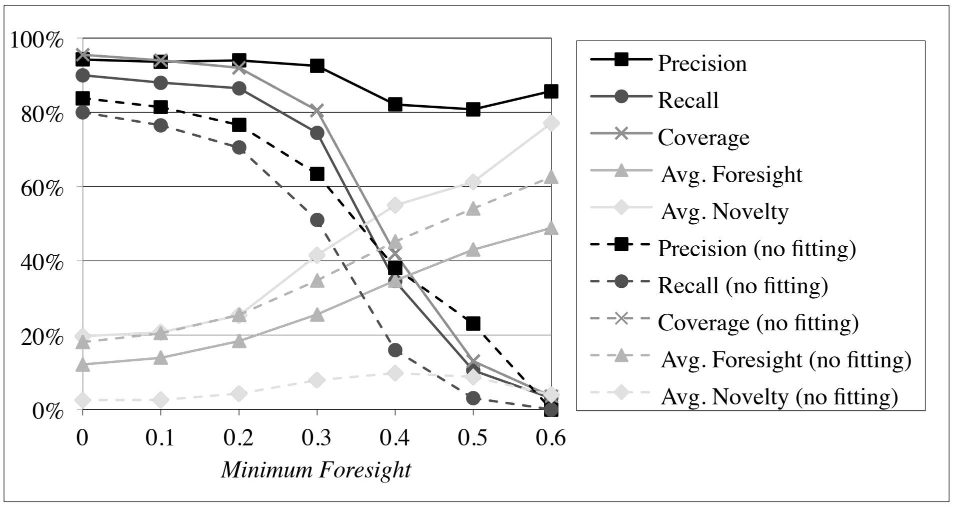


Fig. 9.8 Effectiveness vs. minimum foresight

clear positive effect on accuracy: this is compatible with the intuition that a longer past is more explanatory of the user's behavior and intentions, thus leading to the delivery of better recommendations.

The fourth group of tests was aimed at measuring the effectiveness of the parameter δ set by the user to rule the minimum foresight of the base recommendation. The log used for these tests is again the one with medium density and medium length of sessions, and current sessions have length 4. Figure 9.8 enables a comparison of the accuracy and suitability measures with the fitting phase of our approach switched on (plain lines) and off (dashed lines); in this regard, some interesting observations can be made:

- The average foresight of the base recommendation (no fitting) is always higher than the minimum foresight δ, which confirms the correctness of the approach. However, the average foresight of the final recommendation (with fitting) is slightly lower: in fact, fitting inevitably reduces the foresight by applying modifications that move the base recommendation closer to the current session.
- Fitting has a strong impact on the recommendation correctness. Not only precision is improved by fitting when $\delta = 0$, which indeed is the motivation for the fitting phase, but the increase in precision caused by fitting with higher values of δ is remarkable. The reason is that, when δ is high, the base recommendation tends to be very distant from the current session, so it is probably not simi-

lar to the actual future; however, fitting is quite effective in making the base recommendation compatible with the current session.

- The novelty of the base recommendation is always very low; this is expected, as the base recommendation is just a portion of a log session, hence it will always have a strong similarity with the session from which it was extracted. The novelty of the given recommendation is much higher, indicating that the recommendation is actually something new with respect to what we have in the log. Interestingly, the novelty reaches very high values when δ is also high. This can be explained by considering that the sessions in the log tend to be clustered into some high-density areas; to achieve a high foresight for a current session taken from one of these clusters, base recommendations are mostly chosen from a low-density inter-cluster areas of the log, so fitting transforms them into something very different from the other log sessions.

The fifth group of tests investigates how the recommendation accuracy changes with continued usage of the system. These tests were made on the same three logs used for the second group of tests (a sparse one, a clustered one, and an intermediate one), and the results were averaged. The sessions in each log were then randomly grouped in groups of 20: at the first step, the 20 sessions in the first group were put in the log, and the 20 sessions in the second group were used as current sessions; at the second step, the 20 sessions in the second group were added to the log and the 20 sessions in the third group were used as current sessions; and so on. As a result, the log size is increased by steps of 20 in a way that simulates real usage of the system. Figure 9.9 shows the results. As expected, recall and coverage increase quickly with the log size; precision is quite stable and above 80% even when the log is very small.

The sixth group of tests is focused on the robustness of the approach in terms of stability of the recommendation. Figure 9.10 shows the similarity between the base recommendation and the other candidate recommendations ordered according to their relevance (averaged on groups of 20). The curve smoothly decreases, which means that the most relevant subsessions, as ranked during the ranking phase, are all quite similar to each other. This means that small variations in the current session or in the log features will not change drastically the final recommendation returned.

Finally, we compared our approach with the one proposed in [57] (slightly modified to recommend sessions instead of ordered sets of queries), using a log of 200 sessions with minimum relevance 0 and minimum foresight 0. The approach of [57] is clearly outperformed in terms of accuracy, with a precision of 0.52 (against 0.94) and a recall

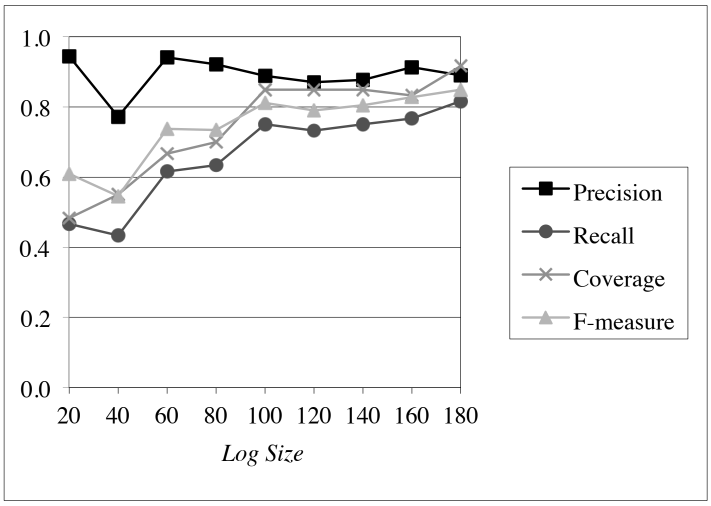


Fig. 9.9 Effectiveness vs. log size

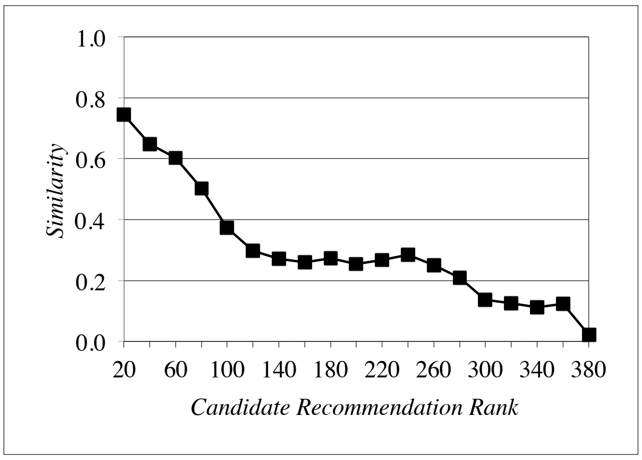


Fig. 9.10 Inter-session similarity for increasing rank of candidate recommendations

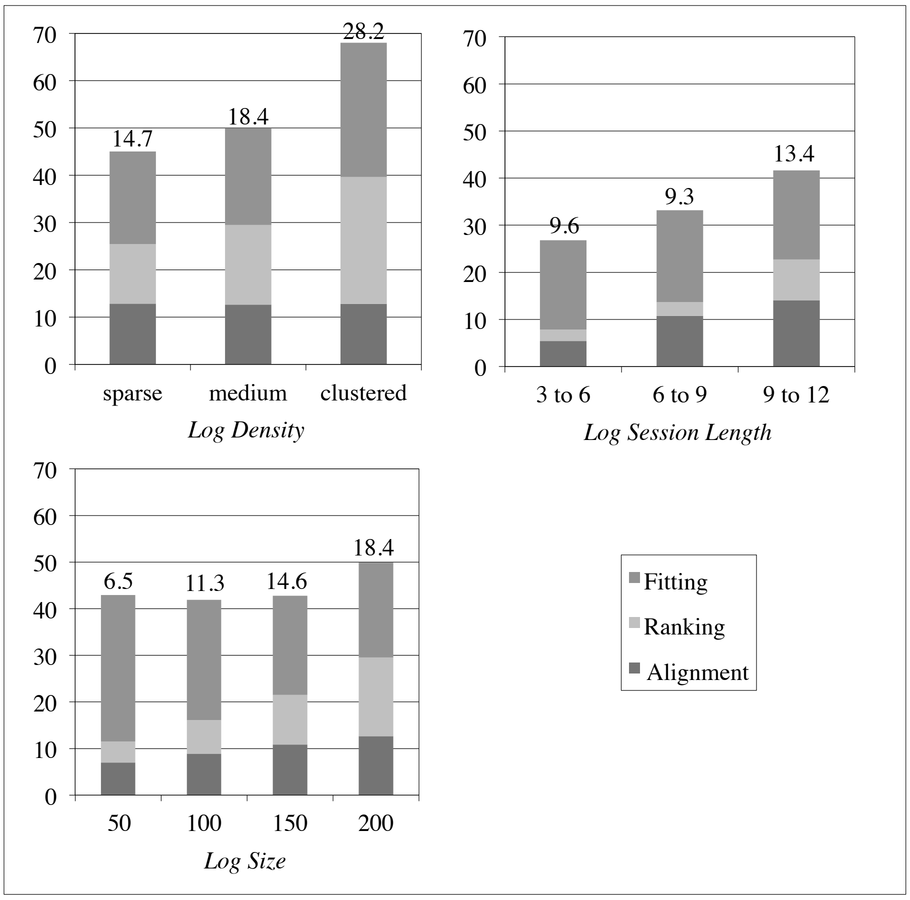


Fig. 9.11 Efficiency vs. log features (all times in msec.; unless otherwise stated, all logs have size 200; labels report the average number of candidate sessions)

of 0.52 (against 0.87). This is explained by the fact that this approach always computes a recommendation (coverage 1 against 0.92) that is chosen from the log (novelty 0 against 0.18, foresight 0.18 against 0.05).

9.5.3 Efficiency Tests

Computing effective recommendations is useless if they are not returned in a time frame compatible with OLAP interactions. Figure 9.11 shows the execution times for our system considering logs with different features. Execution times are split according to the three phases: alignment, ranking, and fitting. Overall, execution times rarely exceed 50 msec., which is perfectly compatible with a human-computer interaction.

Before analyzing in more details the system behavior, we briefly discuss the computational complexity of the three phases. Algorithm 4 looks for the best Smith-Waterman alignment between the current session and those in the log, thus its computational complexity is $O(|D| \times \overline{v}^2)$ where $\overline{v}$ is the average length of the log sessions [7]. Algorithm 5 ranks the candidate sessions computing an all-against-all Smith-Waterman alignment, thus its computational complexity is $O(|F|^2 \times \overline{v}^2)$ where F is the set of candidate recommendations. Finally, Algorithm 6 applies fitting to the base recommendation; its computational complexity is mainly ruled by Type-2 rules extraction, requiring to extract all rules, even infrequent ones, which has an exponential complexity (see e.g., [139]). The time taken remains acceptable though, since the average number of fragments $\overline{f}$ that are common to all the queries of a base recommendation is low. Type-1 rules extraction is polynomial, due to the nature of the rules extracted. However, Type-1 rules extraction takes as input the set of fragments of both the current session and the log session, whose size can be greater than $\overline{f}$, especially in the presence of small logs, where the similarity between the log session and the current session is likely to be low. Clearly, the execution time of each phase depends on the one hand on the cost of the basic operation carried out (i.e., alignment for Algorithms 4 and 5), on the other hand on the number of times such operation is executed. In the light of this premise, the following considerations hold:

- The costs for alignment and ranking increase with the log size $|D|$ and the average session length $|v|$, which jointly determine the number of alignments found with the current session.

- Though fitting works on a single session, its cost is predominant due to the high computational complexity of rule extraction. Unexpectedly, the execution time of fitting increases as the log size decreases; this is due to the fact that the extraction of Type-1 rules is computationally more expensive when the current session is less similar to the log session, which is more likely in the case of smaller logs.

- As to the two remaining phases, as suggested by the complexity estimates reported above, the predominance of either alignment or ranking depends on the relationship between $|F|^2$ and $|D|$: the cost of ranking tends to become higher than that of alignment for large and clustered logs, that determine several candidate recommendations thus making $|F|^2 > |D|$.

As to the comparison with [57], the tests show that our approach is slightly worse in terms of efficiency (50 msec. against 17.8 msec.) due to the extra costs paid for the fitting phase; however, as discussed in Section 9.5.2, the higher effectiveness largely compensates for this lower efficiency.

9.6 Conclusions

In this chapter we have proposed a collaborative filtering approach to recommendation of OLAP sessions that reuses knowledge acquired by other users during previous sessions [5]. Like other collaborative approaches, ours follows a three-phases process, but it is the first that treats sessions as first-class citizens, using brand new techniques for comparing sessions, finding meaningful recommendation candidates, and adapting them to the current session. We have extensively evaluated it by discussing its efficiency and its effectiveness from different points of view. Though our approach ensures that the returned recommendations have several desirable properties (such as novelty, relevance, and foresight), we plan to further improve it under different aspects:

- We have observed that large logs and longer log sessions are necessary to obtain a good coverage (see Figures 9.7 and 9.9). To cope with this well known cold-start problem, extending our approach with non collaborative recommendations (like, e.g., [157]) is a promising research direction.

- As shown in Figure 9.10, several candidate recommendations with high relevance can normally be found. Though we have chosen to recommend only the top-1 session, the approach can be easily reworked to recommend a top-k set of sessions. In this case, the approach effectiveness could benefit from query result diversification [44].

- The user should be enabled to easily understand in which direction a recommendation will guide her through multidimensional data. Since we are recommending sessions rather than single queries, and sessions are complex objects, a visualization problem arises. Solving this problem requires to (i) understand the set of features that describe an OLAP session direction at best, and to (ii) find a good visualization metaphor.

- Our approach has been tested with synthetic, yet realistic workloads. However, given real OLAP logs, characterizing user sessions and analyzing sessions and queries to filter out the irrelevant ones (e.g., those showing an erratic behavior due to a trial-and-error user approach), remain open problems, that should be solved to better adapt our approach to different kinds of users. We are currently working to collect real, user-annotated logs, as well as user feedback on our recommender system.

Chapter 10

Conclusions and future work

In this thesis we presented a series of contributions given in the field of business intelligence, with the goal to enhance the decision-making process by allowing the integration and analysis of *non-conventional data.* The pertinence of these contributions falls within different disciplines of BI 2.0, namely *Social BI*, *Exploratory BI* and *Pervasive BI.*

The field of Social BI revolves around the analysis of *social data*, as they represent an important source to perceive trends and moods from the environment. On one hand, we contributed by presenting an architectural and methodological framework to design a Social BI solution: we analyzed the main factors that impact on the cost-benefit trade-off in choosing the architectural components and we proposed insights and suggestions, based on real experience on a Social BI project (WebPolEU). Eventually, this lead to the release of SABINE, a cross-disciplinary benchmark built on the data collected in WebPolEU. By providing a variety of validated enrichments of the social contents and a user-validated ground truth, with SABINE we enable experimentations and comparisons with reference to the most commonly performed tasks in Social BI.

On the other hand, we proposed meta-stars as an expressive solution to model hierarchies of topics in OLAP cubes. Meta-stars answer the requirements of irregularity and fluidity in the topic hierarchy, integrability with business hierarchies and semantic-aware aggregations. The extensive evaluations also proved that meta-stars grant higher expressiveness and dynamicity than traditional dimension tables without affecting significantly performances and space occupation. Nonetheless, there is still margin for improvement on different aspects. For starters, the static modeling of some portions of the topic hierarchy can be optimized to avoid unnecessary redundancies and to improve the performance of SQL queries. Secondly, we are working to enable the control over the topic hierarchy by means of an ontology: not only would it open to an automatic procedure to build and maintain meta-star, but it would allow to directly query the topic hierarchy on the ontology (by coupling SQL and OWL) without requiring the materialization of a large roll-up table.

As for Exploratory BI, we investigated how the cross-domain knowledge lying within *linked data* can be integrated with corporate cubes. To this end, we proposed iMOLD, an interactive and collaborative approach that enables users to explore many-to-one relationships in the linked data and to translate them into multidimensional hierarchies. At the heart of iMOLD is the identification of five different RDF patterns that go beyond the classical functional dependencies approach for multidimensional design. Though iMOLD is a significant step towards linked data integration, several aspects are open to future developments. In terms of linked data exploration, algorithms for instance-based discovery of approximate functional dependencies can be used to retrieve non-explicit FDs that are hidden in the data (e.g., TANE [78]). Moreover, the approach can be improved with the automatic recognition and management of more complex kinds of modeling (e.g., cycles, convergences and many-to-many relationships). Finally, an interesting development would be to extend the approach to detect entire cubes (and not only hierarchies) from linked data.

In the field of Pervasive BI, we focused on *schemaless data* and addressed the issue of making sense of the heterogeneous wealth of data stored in a corporate data lake. In particular, we proposed an approach to capture the rules that explain the use of different schemata within a collection by means of a decision tree. The distinguishing aspects are the coupling of value-based and schema-based conditions to explain the schema usage, as well as the adoption of a split strategy guided by the requirements elicited from users, who asked for precise, concise, and explicative schema profiles. The effectiveness of the approach is demonstrated in the experimental evaluation, which show a high accuracy in delivering schema profile that match the provided baseline. As for future developments, a first work will consist in incorporating user interaction in the schema profile building algorithm, in order to give users a closer control on the elaboration of the schema profiles; ultimately, this would lead to a more comprehensive understanding of the schema usage and to a finer accommodation of the characteristics of the collection. Furthermore, schema profiling is the first step to enable effective analyses over data lakes in business intelligence contexts. To this end, we will also investigate how schema profiles can be exploited in schema-on-read approaches to improve the effectiveness and flexibility of analytical queries [109, 34].

Finally, a complementary part of this thesis was dedicated to the address the issue of user disorientation in OLAP analyses — an issue that the inclusion of non-conventional data tends to amplify, as it further increases the wealth of information at the user's disposal. In this direction, we entered the field of recommendations and proposed an innovative approach that provides users with OLAP sessions instead of single queries. We presented an algorithm that uses brand new techniques to identify meaningful candidates and to adapt them to the user's current session. The effectiveness of the approach was demonstrated by evaluating several desirable properties in a recommendation, such as novelty, relevance, and foresight. Our plan for future works is oriented to the improvement of different aspects of the algorithm: (i) solving the well known cold-start problem by extending our approach with non collaborative recommendations (like, e.g., [157]); (ii) reworking the approach to propose a series of recommendations, in accordance with the principle of result diversification [44]; (iii) proposing a visualization metaphor to facilitate the user in understanding the directions that different sessions take in the multidimensional space; (iv) collecting real, user-annotated workloads in order to build a characterization of users and allow the proposal of more personalized recommendations.

Contextually to this work, we also proposed a benchmark generator for OLAP sessions, named CubeLoad, which enabled the generation of synthetic workloads for the experimental evaluation of the recommendation algorithm. The benchmark uses several parameters to generate a profile-based workload model and relies on four realistic templates that simulate the typical behaviors of OLAP users. The set of tests carried out showed how it is possible to generate very different workloads by carefully tuning the provided parameters. Nonetheless, several improvements can be made to the benchmark, starting from the adoption of a more sophisticated query model so as to generate more complex queries. Furthermore, the aspect of realism can be enhanced by considering more templates and parameters (e.g., by allowing the distinction of skilled and non-skilled profiles); to this end, an appealing opportunity is to refactor the CubeLoad code according to an open architecture, so that users can collaboratively contribute by proposing different templates in the form of plugins.

References

[1] Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J., Naumann, F., Pedersen, T. B., Rizzi, S., Trujillo, J., Vassiliadis, P., and Vossen, G. (2013). Fusion cubes: Towards self-service business intelligence. *IJDWM*, 9(2):66–88.

[2] Abelló, A., Romero, O., Pedersen, T. B., Llavori, R. B., Nebot, V., Cabo, M. J. A., and Simitsis, A. (2015). Using semantic web technologies for exploratory OLAP: A survey. *IEEE Trans. Knowl. Data Eng.*, 27(2):571–588.

[3] Abelló, A., Samos, J., and Saltor, F. (2001). Understanding analysis dimensions in a multidimensional object-oriented model. In *Proceedings of the 3rd Intl. Workshop on Design and Management of Data Warehouses, DMDW'2001, Interlaken, Switzerland, June 4, 2001*, page 4. CEUR-WS.org.

[4] Adomavicius, G. and Tuzhilin, A. (2005). 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):734–749.

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

[6] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E. (2011). Mining preferences from OLAP query logs for proactive personalization. In *Advances in Databases and Information Systems - 15th International Conference, ADBIS 2011, Vienna, Austria, September 20-23, 2011. Proceedings*, pages 84–97. Springer.

[7] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E. (2014). Similarity measures for OLAP sessions. *Knowl. Inf. Syst.*, 39(2):463–489.

[8] Aligon, J. and Marcel, P. (2012). A framework for user-centric summaries of OLAP sessions. In *Proceedings EDA*, pages 103–117, Bordeaux, France.

[9] Aufaure, M., Kuchmann-Beauger, N., Marcel, P., Rizzi, S., and Vanrompay, Y. (2013). Predicting your next OLAP query based on recent analytical sessions. In *Data Warehousing and Knowledge Discovery - 15th International Conference, DaWaK 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings*, pages 134–145. Springer.

[10] Azabou, M., Khrouf, K., Feki, J., Soulé-Dupuy, C., and Vallès, N. (2014). A novel multidimensional model for the OLAP on documents: Modeling, generation and implementation. In *Model and Data Engineering - 4th International Conference,*

MEDI 2014, Larnaca, Cyprus, September 24-26, 2014. Proceedings, pages 258–272. Springer.

[11] Bernstein, P. A., Madhavan, J., and Rahm, E. (2011). Generic schema matching, ten years later. *PVLDB*, 4(11):695–701.

[12] Bex, G. J., Gelade, W., Neven, F., and Vansummeren, S. (2010). Learning deterministic regular expressions for the inference of schemas from XML data. *TWEB*, 4(4):14:1–14:32.

[13] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22.

[14] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[15] Bogdanov, P., Busch, M., Moehlis, J., Singh, A. K., and Szymanski, B. K. (2014). Modeling individual topic-specific behavior and influence backbone networks in social media. *Social Netw. Analys. Mining*, 4(1):204.

[16] Bohannon, P., Elnahrawy, E., Fan, W., and Flaster, M. (2006). Putting context into schema matching. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 307–318. ACM.

[17] Carpineto, C. and Romano, G. (2012). A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50.

[18] Castano, S., Ferrara, A., Genta, L., and Montanelli, S. (2016). Combining crowd consensus and user trustworthiness for managing collective tasks. *Future Generation Comp. Syst.*, 54:378–388.

[19] Castano, S., Ferrara, A., and Montanelli, S. (2006). Matching ontologies in open networked systems: Techniques and applications. In *Journal on Data Semantics V*, pages 25–63. Springer.

[20] Castano, S., Ferrara, A., and Montanelli, S. (2011). Thematic exploration of linked data. In *Proceedings of the First International Workshop on Searching and Integrating New Web Data Sources - Very Large Data Search, Seattle, WA, USA, September 2, 2011*, pages 11–16. CEUR-WS.org.

[21] Castellanos, M., Dayal, U., Hsu, M., Ghosh, R., Dekhil, M., Lu, Y., Zhang, L., and Schreiman, M. (2011). LCI: a social channel analysis platform for live customer intelligence. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 1049–1058. ACM.

[22] Chang, K. C. and Garcia-Molina, H. (1999). Mind your vocabulary: Query mapping across heterogeneous information sources. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 335–346. ACM Press.

[23] Chu, C., Nakazawa, T., and Kurohashi, S. (2014). Iterative bilingual lexicon extraction from comparable corpora with topical and contextual knowledge. In *Computational Linguistics and Intelligent Text Processing - 15th International Conference, CICLing 2014, Kathmandu, Nepal, April 6-12, 2014, Proceedings, Part II*, pages 296–309. Springer.

[24] Ciaccia, P., Golfarelli, M., and Rizzi, S. (2013). Efficient derivation of numerical dependencies. *Inf. Syst.*, 38(3):410–429.

[25] Colliat, G. (1996). Olap, relational, and multidimensional database systems. *SIGMOD Record*, 25(3):64–69.

[26] Crawford, S. L. (1989). Extensions to the CART algorithm. *International Journal of Man-Machine Studies*, 31(2):197–217.

[27] Dai, W. and Ji, W. (2014). A MapReduce implementation of C4.5 decision tree algorithm. *Int. Jour. of Database Theory and Application*, 7(1):49–60.

[28] Darmont, J., Bentayeb, F., and Boussaid, O. (2007). DWEB: A data warehouse engineering benchmark. *CoRR*, abs/0705.1453.

[29] Dayal, U., Gupta, C., Castellanos, M., Wang, S., and García-Solaco, M. (2012). Of cubes, dags and hierarchical correlations: A novel conceptual model for analyzing social media data. In *Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings*, pages 30–49. Springer.

[30] Deng, H., Runger, G. C., and Tuv, E. (2011). Bias of importance measures for multi-valued attributes and solutions. In *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part II*, pages 293–300. Springer.

[31] Deutsch, T. (2013). Why is schema on read so useful? http://www.ibmbigdatahub.com/blog/why-schema-read-so-useful.

[32] Dhamankar, R., Lee, Y., Doan, A., Halevy, A. Y., and Domingos, P. M. (2004). iMAP: discovering complex semantic matches between database schemas. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 383–394. ACM.

[33] Domingos, P. M. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 71–80. ACM.

[34] Dong, X. L. and Srivastava, D. (2013). Big data integration. *PVLDB*, 6(11):1188–1189.

[35] Dong, Y., Johnson, R. A., and Chawla, N. V. (2015). Will this paper increase your *h*-index?: Scientific impact prediction. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 149–158. ACM.

[36] Drosou, M. and Pitoura, E. (2013). YmalDB: exploring relational databases via result-driven recommendations. *VLDB J.*, 22(6):849–874.

[37] Eberius, J., Thiele, M., Braunschweig, K., and Lehner, W. (2012). Drillbeyond: Enabling business analysts to explore the web of open data. *PVLDB*, 5(12):1978–1981.

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

[39] Etcheverry, L. and Vaisman, A. A. (2012). QB4OLAP: A vocabulary for OLAP cubes on the semantic web. In *Proceedings of the Third International Workshop on Consuming Linked Data, COLD 2012, Boston, MA, USA, November 12, 2012*. CEUR-WS.org.

[40] Fernández, M., Cantador, I., López, V., Vallet, D., Castells, P., and Motta, E. (2011). Semantically enhanced information retrieval: An ontology-based approach. *J. Web Sem.*, 9(4):434–452.

[41] Ferrara, A., Nikolov, A., and Scharffe, F. (2013). Data linking for the semantic web. *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications*, 169.

[42] Francia, M., Gallinucci, E., Golfarelli, M., and Rizzi, S. (2016). Social business intelligence in action. In *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, pages 33–48. Springer.

[43] Francia, M., Golfarelli, M., and Rizzi, S. (2014). A methodology for social BI. In *18th International Database Engineering & Applications Symposium, IDEAS 2014, Porto, Portugal, July 7-9, 2014*, pages 207–216. ACM.

[44] Fraternali, P., Martinenghi, D., and Tagliasacchi, M. (2012). Top-k bounded diversification. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 421–432. ACM.

[45] Fung, P. (1998). A statistical view on bilingual lexicon extraction: From parallel corpora to non-parallel corpora. In *Machine Translation and the Information Soup, Third Conference of the Association for Machine Translation in the Americas, AMTA '98, Langhorne, PA, USA, October 28-31, 1998, Proceedings*, pages 1–17. Springer.

[46] Furche, T., Gottlob, G., Grasso, G., Schallhart, C., and Sellers, A. J. (2013). Oxpath: A language for scalable data extraction, automation, and crawling on the deep web. *VLDB J.*, 22(1):47–72.

[47] Gallinucci, E., Golfarelli, M., and Rizzi, S. (2013). Meta-stars: multidimensional modeling for social business intelligence. In *Proceedings of the sixteenth international workshop on Data warehousing and OLAP, DOLAP 2013, San Francisco, CA, USA, October 28, 2013*, pages 11–18. ACM.

[48] Gallinucci, E., Golfarelli, M., and Rizzi, S. (2015a). Advanced topic modeling for social business intelligence. *Inf. Syst.*, 53:87–106.

[49] Gallinucci, E., Golfarelli, M., and Rizzi, S. (2015b). Meta-stars: Dynamic, schemaless, and semantically-rich topic hierarchies in social BI. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 529–532. OpenProceedings.org.

[50] Gao, W., Li, P., and Darwish, K. (2012). Joint topic modeling for event summarization across news and social media streams. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1173–1182. ACM.

[51] Garcia-Alvarado, C. and Ordonez, C. (2012). Query processing on cubes mapped from ontologies to dimension hierarchies. In *DOLAP 2012, ACM 15th International Workshop on Data Warehousing and OLAP, Maui, HI, USA, November 2, 2012, Proceedings*, pages 57–64. ACM.

[52] Garcia-Molina, H., Ullman, J. D., and Widom, J. (2000). *Database System Implementation.* Prentice-Hall.

[53] García-Moya, L., Kudama, S., Aramburu, M. J., and Llavori, R. B. (2013). Storing and analysing voice of the market data in the corporate data warehouse. *Information Systems Frontiers*, 15(3):331–349.

[54] Garofalakis, M. N., Gionis, A., Rastogi, R., Seshadri, S., and Shim, K. (2000). XTRACT: A system for extracting document type descriptors from XML documents. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 165–176. ACM.

[55] Ge, M., Delgado-Battenfeld, C., and Jannach, D. (2010). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 257–260. ACM.

[56] Gehrke, J., Ganti, V., Ramakrishnan, R., and Loh, W. (1999). Boat-optimistic decision tree construction. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 169–180. ACM Press.

[57] Giacometti, A., Marcel, P., and Negre, E. (2009). Recommending multidimensional queries. In *Data Warehousing and Knowledge Discovery, 11th International Conference, DaWaK 2009, Linz, Austria, August 31 - September 2, 2009, Proceedings*, pages 453–466. Springer.

[58] Giacometti, A., Marcel, P., Negre, E., and Soulet, A. (2011). Query recommendations for OLAP discovery-driven analysis. *IJDWM*, 7(2):1–25.

[59] Glance, N. S., Hurst, M., Nigam, K., Siegler, M., Stockton, R., and Tomokiyo, T. (2005). Deriving marketing intelligence from online discussion. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 419–428. ACM.

[60] Golfarelli, M., Lechtenbörger, J., Rizzi, S., and Vossen, G. (2006). Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. *Data Knowl. Eng.*, 59(2):435–459.

[61] Golfarelli, M., Maio, D., and Rizzi, S. (1998). The dimensional fact model: A conceptual model for data warehouses. *Int. J. Cooperative Inf. Syst.*, 7(2-3):215–247.

[62] Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., and Turricchia, E. (2012). OLAP query reformulation in peer-to-peer data warehousing. *Inf. Syst.*, 37(5):393–411.

[63] Golfarelli, M. and Rizzi, S. (2009). *Data warehouse design: Modern principles and methodologies.* McGraw-Hill, Inc.

[64] Golfarelli, M., Rizzi, S., and Biondi, P. (2011). myolap: An approach to express and evaluate OLAP preferences. *IEEE Trans. Knowl. Data Eng.*, 23(7):1050–1064.

[65] Golfarelli, M. and Saltarelli, E. (2003). The workload you have, the workload you would like. In *DOLAP 2003, ACM Sixth International Workshop on Data Warehousing and OLAP, New Orleans, Louisiana, USA, November 7, 2003, Proceedings*, pages 79–85. ACM.

[66] Guerrini, G., Mesiti, M., and Sanz, I. (2007). An overview of similarity measures for clustering XML documents. In *Web Data Management Practices: Emerging Techniques and Technologies*, pages 56–78. Idea Group.

[67] Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10:2935–2962.

[68] Gündüz, S. and Özsu, M. T. (2003). A web page prediction model based on click-stream tree representation of user behavior. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 535–540. ACM.

[69] Halb, W., Raimond, Y., and Hausenblas, M. (2008). Building linked data for both humans and machines. In *Proceedings of the WWW2008 Workshop on Linked Data on the Web, LDOW 2008, Beijing, China, April 22, 2008.* CEUR-WS.org.

[70] Hegewald, J., Naumann, F., and Weis, M. (2006). Xstruct: Efficient schema extraction from multiple and large XML documents. In *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, page 81. IEEE Computer Society.

[71] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

[72] Hirao, T., Iwata, T., and Nagata, M. (2013). Latent semantic matching: Application to cross-language text categorization without alignment information. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 212–216. The Association for Computer Linguistics.

[73] Hirsch, C., Hosking, J., and Grundy, J. (2009). Interactive visualization tools for exploring the semantic graph of large knowledge spaces. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, volume 443.

[74] Hogan, A., Harth, A., Passant, A., Decker, S., and Polleres, A. (2010). Weaving the pedantic web. In *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010.* CEUR-WS.org.

[75] Hopkins, B. and Skellam, J. G. (1954). A new method for determining the type of distribution of plant individuals. *Annals of Botany*, 18(2):213–227.

[76] Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 168–177. ACM.

[77] Hub, I. B. D. . A. (2013). The four v's of big data. http://www.ibmbigdatahub.com/infographic/four-vs-big-data.

[78] Huhtala, Y., Kärkkäinen, J., Porkka, P., and Toivonen, H. (1999). TANE: an efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111.

[79] Hüllermeier, E. and Rifqi, M. (2009). A fuzzy variant of the rand index for comparing clustering structures. In *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference, Lisbon, Portugal, July 20-24, 2009*, pages 1294–1298.

[80] Hyde, J. (2011). Mondrian documentation. http://mondrian.pentaho.com/documentation/schema.php.

[81] Ibragimov, D., Hose, K., Pedersen, T. B., and Zimányi, E. (2014). Towards exploratory OLAP over linked open data - A case study. In *Enabling Real-Time Business Intelligence - International Workshops, BIRTE 2013, Riva del Garda, Italy, August 26, 2013, and BIRTE 2014, Hangzhou, China, September 1, 2014, Revised Selected Papers*, pages 114–132. Springer.

[82] Inmon, W. H. (2005). *Building the data warehouse.* John wiley & sons.

[83] Izquierdo, J. L. C. and Cabot, J. (2013). Discovering implicit schemas in JSON data. In *Web Engineering - 13th International Conference, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings*, pages 68–83. Springer.

[84] Jensen, M. R., Holmgren, T., and Pedersen, T. B. (2004). Discovering multidimensional structure in relational data. In *Data Warehousing and Knowledge Discovery, 6th International Conference, DaWaK 2004, Zaragoza, Spain, September 1-3, 2004, Proceedings*, pages 138–148. Springer.

[85] Jerbi, H., Ravat, F., Teste, O., and Zurfluh, G. (2009a). Applying recommendation technology in OLAP systems. In *Enterprise Information Systems, 11th International Conference, ICEIS 2009, Milan, Italy, May 6-10, 2009. Proceedings*, pages 220–233. Springer.

[86] Jerbi, H., Ravat, F., Teste, O., and Zurfluh, G. (2009b). Preference-based recommendations for OLAP analysis. In *Data Warehousing and Knowledge Discovery, 11th International Conference, DaWaK 2009, Linz, Austria, August 31 - September 2, 2009, Proceedings*, pages 467–478. Springer.

[87] Jovanovic, P., Romero, O., Simitsis, A., Abelló, A., and Mayorova, D. (2014). A requirement-driven approach to the design and evolution of data warehouses. *Inf. Syst.*, 44:94–119.

[88] Kahan, J., Koivunen, M., Prud'hommeaux, E., and Swick, R. R. (2002). Annotea: an open RDF infrastructure for shared web annotations. *Computer Networks*, 39(5):589–608.

[89] Kalles, D. and Morris, T. (1996). Efficient incremental induction of decision trees. *Machine Learning*, 24(3):231–242.

[90] Kämpgen, B., O'Riain, S., and Harth, A. (2012). Interacting with statistical linked data via OLAP operations. In *The Semantic Web: ESWC 2012 Satellite Events - ESWC 2012 Satellite Events, Heraklion, Crete, Greece, May 27-31, 2012. Revised Selected Papers*, pages 87–101. Springer.

[91] Kämpgen, B., Stadtmüller, S., and Harth, A. (2014). Querying the global cube: Integration of multidimensional datasets from the web. In *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, pages 250–265. Springer.

[92] Kersten, M. L., Idreos, S., Manegold, S., and Liarou, E. (2011). The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB*, 4(12):1474–1477.

[93] Khouri, S., Boukhari, I., Bellatreche, L., Sardet, E., Jean, S., and Baron, M. (2012). Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. *Computers in Industry*, 63(8):799–812.

[94] Khoussainova, N., Balazinska, M., Gatterbauer, W., Kwon, Y., and Suciu, D. (2009). A case for A collaborative query management system. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*. www.cidrdb.org.

[95] Kimball, R. and Ross, M. (2008). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons.

[96] Kivinen, J. and Mannila, H. (1995). Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149.

[97] Klettke, M., Störl, U., and Scherzinger, S. (2015). Schema extraction and structural outlier detection for json-based nosql data stores. In *Datenbanksysteme für Business, Technologie und Web (BTW), 16. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 4.-6.3.2015 in Hamburg, Germany. Proceedings*, pages 425–444. GI.

[98] Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artif. Intell. Rev.*, 39(4):261–283.

[99] Krumm, J., Davies, N., and Narayanaswami, C. (2008). User-generated content. *IEEE Pervasive Computing*, 7(4):10–11.

[100] Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

[101] Lee, J., Grossman, D. A., Frieder, O., and McCabe, M. C. (2000). Integrating structured data and text: A multi-dimensional approach. In *2000 International Symposium on Information Technology (ITCC 2000), 27-29 March 2000, Las Vegas, NV, USA*, pages 264–271. IEEE Computer Society.

[102] Lee, L. H. and Isa, D. (2010). Automatically computed document dependent weighting factor facility for naïve bayes classification. *Expert Syst. Appl.*, 37(12):8471–8478.

[103] Lee, L. H., Wan, C. H., Rajkumar, R., and Isa, D. (2012). An enhanced support vector machine classification framework by using euclidean distance function for text document categorization. *Appl. Intell.*, 37(1):80–99.

[104] Lee, M., Yang, L. H., Hsu, W., and Yang, X. (2002). Xclust: clustering XML schemas for effective integration. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002*, pages 292–299. ACM.

[105] LeFevre, J., Sankaranarayanan, J., Hacigümüs, H., Tatemura, J., and Polyzotis, N. (2013). Towards a workload for evolutionary analytics. *CoRR*, abs/1304.1838.

[106] Lenz, H. and Shoshani, A. (1997). Summarizability in OLAP and statistical data bases. In *Ninth International Conference on Scientific and Statistical Database Management, Proceedings, August 11-13, 1997, Olympia, Washington, USA*, pages 132–143. IEEE Computer Society.

[107] Liu, B. and Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pages 415–463. Springer.

[108] Liu, Y., Cai, J., Yin, J., and Fu, A. W. (2008). Clustering text data streams. *J. Comput. Sci. Technol.*, 23(1):112–128.

[109] Liu, Z. H. and Gawlick, D. (2015). Management of flexible schema data in rdbmss - opportunities and limitations for nosql -. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org.

[110] Malinowski, E. and Zimányi, E. (2006). Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data Knowl. Eng.*, 59(2):348–377.

[111] Markl, V. (2008). Situational business intelligence. In *Informal Proceedings of the Second International Workshop on Business Intelligence for the Real-Time Enterprise, BIRTE 2008, in conjunction with VLDB'08, August 24, 2008, Auckland, New Zealand*.

[112] Maruta, K., Nagai, H., and Nakamura, T. (2015). Document classification with varied viewpoints using matrix decomposition. In *Proceedings of the 2015 IIAI 4th International Congress on Advanced Applied Informatics*, pages 154–159. IEEE Computer Society.

[113] Mathioudakis, M. and Koudas, N. (2010). Twittermonitor: trend detection over the twitter stream. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 1155–1158. ACM.

[114] McCord, M. C. (1990). *Slot grammar.* Springer.

[115] Michelson, M. and Macskassy, S. A. (2010). Discovering users' topics of interest on twitter: a first look. In *Proceedings of the Fourth Workshop on Analytics for Noisy Unstructured Text Data, AND 2010, Toronto, Ontario, Canada, October 26th, 2010 (in conjunction with CIKM 2010)*, pages 73–80. ACM.

[116] Minnesota Population Center (2008). Integrated public use microdata series. http://www.ipums.org.

[117] Mirizzi, R., Ragone, A., Noia, T. D., and Sciascio, E. D. (2010). Semantic wonder cloud: Exploratory search in dbpedia. In *Current Trends in Web Engineering - 10th International Conference on Web Engineering, ICWE 2010 Workshops, Vienna, Austria, July 2010, Revised Selected Papers*, pages 138–149. Springer.

[118] Miyahara, T., Shoudai, T., Uchida, T., Takahashi, K., and Ueda, H. (2001). Discovery of frequent tree structured patterns in semistructured web documents. In *Knowledge Discovery and Data Mining - PAKDD 2001, 5th Pacific-Asia Conference, Hong Kong, China, April 16-18, 2001, Proceedings*, pages 47–52. Springer.

[119] Montgomery, A. L., Li, S., Srinivasan, K., and Liechty, J. C. (2004). Modeling online browsing and path analysis using clickstream data. *Marketing Science*, 23(4):579–595.

[120] Morton, K., Balazinska, M., Grossman, D., and Mackinlay, J. D. (2014). Support the data enthusiast: Challenges for next-generation data-analysis systems. *PVLDB*, 7(6):453–456.

[121] Nagwani, N. (2015). Summarizing large text collection using topic modeling and clustering based on mapreduce framework. *Journal of Big Data*, 2(1):1.

[122] Nayak, R. and Iryadi, W. (2007). XML schema clustering with semantic and hierarchical similarity measures. *Knowl.-Based Syst.*, 20(4):336–349.

[123] Nebot, V. and Llavori, R. B. (2014). Towards analytical MD stars from linked data. In *KDIR 2014 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, Rome, Italy, 21 - 24 October, 2014*, pages 117–125. SciTePress.

[124] Nebot, V., Llavori, R. B., Pérez-Martínez, J. M., Aramburu, M. J., and Pedersen, T. B. (2009). Multidimensional integrated ontologies: A framework for designing semantic data warehouses. *J. Data Semantics*, 13:1–36.

[125] Nelson, G. S. (2010). Business intelligence 2.0: Are we there yet. In *SAS Global Forum*, pages 50–62. Citeseer.

[126] Neri, F., Aliprandi, C., Capeci, F., Cuadros, M., and By, T. (2012). Sentiment analysis on social media. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012, Istanbul, Turkey, 26-29 August 2012*, pages 919–926. IEEE Computer Society.

[127] Nestorov, S., Abiteboul, S., and Motwani, R. (1998). Extracting schema from semistructured data. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 295–306. ACM Press.

[128] Nestorov, S., Ullman, J. D., Wiener, J. L., and Chawathe, S. S. (1997). Representative objects: Concise representations of semistructured, hierarchial data. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K.*, pages 79–90. IEEE Computer Society.

[129] Niemi, T., Nummenmaa, J., and Thanisch, P. (2001). Logical multidimensional database design for ragged and unbalanced aggregation. In *Proceedings of the 3rd Intl. Workshop on Design and Management of Data Warehouses, DMDW'2001, Interlaken, Switzerland, June 4, 2001*, page 7. CEUR-WS.org.

[130] Nottelmann, H. and Straccia, U. (2005). splmap: A probabilistic approach to schema matching. In *Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005, Proceedings*, pages 81–95. Springer.

[131] O'Neil, P. E., O'Neil, E. J., Chen, X., and Revilak, S. (2009). The star schema benchmark and augmented fact table indexing. In *Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers*, pages 237–252. Springer.

[132] Otero, P. G. (2007). Learning bilingual lexicons from comparable English and Spanish corpora. *Proceedings of MT Summit xI*, pages 191–198.

[133] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. *CoRR*, cs.CL/0205070.

[134] Pedersen, T. B., Jensen, C. S., and Dyreson, C. E. (2001). A foundation for capturing and querying complex multidimensional data. *Inf. Syst.*, 26(5):383–423.

[135] Poess, M. (2007). Controlled SQL query evolution for decision support benchmarks. In *Proceedings of the 6th International Workshop on Software and Performance, WOSP 2007, Buenes Aires, Argentina, February 5-8, 2007*, pages 38–41. ACM.

[136] Pöss, M., Smith, B., Kollár, L., and Larson, P. (2002). Tpc-ds, taking decision support benchmarking to the next level. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pages 582–587. ACM.

[137] Pound, J., Mika, P., and Zaragoza, H. (2010). Ad-hoc object retrieval in the web of data. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 771–780. ACM.

[138] Priebe, T. and Pernul, G. (2003). Ontology-based integration of OLAP and information retrieval. In *14th International Workshop on Database and Expert Systems Applications (DEXA'03), September 1-5, 2003, Prague, Czech Republic*, pages 610–614. IEEE Computer Society.

[139] Purdom, P. W., Gucht, D. V., and Groth, D. P. (2004). Average-case performance of the apriori algorithm. *SIAM J. Comput.*, 33(5):1223–1260.

[140] Quadrianto, N., Smola, A. J., Song, L., and Tuytelaars, T. (2010). Kernelized sorting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(10):1809–1821.

[141] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

[142] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

[143] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

[144] Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. (2007). A conceptual model for multidimensional analysis of documents. In *Conceptual Modeling - ER 2007, 26th International Conference on Conceptual Modeling, Auckland, New Zealand, November 5-9, 2007, Proceedings*, pages 550–565. Springer.

[145] Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. (2008). Top_keyword: An aggregation function for textual document OLAP. In *Data Warehousing and Knowledge Discovery, 10th International Conference, DaWaK 2008, Turin, Italy, September 2-5, 2008, Proceedings*, pages 55–64. Springer.

[146] Rehman, N. U., Mansmann, S., Weiler, A., and Scholl, M. H. (2012). Building a data warehouse for twitter stream exploration. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012, Istanbul, Turkey, 26-29 August 2012*, pages 1341–1348. IEEE Computer Society.

[147] Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011). Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John*

McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1524–1534. ACL.

[148] Rizzi, S. and Gallinucci, E. (2014). Cubeload: A parametric generator of realistic OLAP workloads. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, pages 610–624. Springer.

[149] Romero, O. and Abelló, A. (2009). A survey of multidimensional modeling methodologies. *IJDWM*, 5(2):1–23.

[150] Romero, O., Calvanese, D., Abelló, A., and Rodriguez-Muro, M. (2009). Discovering functional dependencies for multidimensional design. In *DOLAP 2009, ACM 12th International Workshop on Data Warehousing and OLAP, Hong Kong, China, November 6, 2009, Proceedings*, pages 1–8. ACM.

[151] Rosenfeld, B. and Feldman, R. (2007). Clustering for unsupervised relation identification. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 411–418. ACM.

[152] Ruggieri, S. (2002). Efficient C4.5. *IEEE Trans. Knowl. Data Eng.*, 14(2):438–444.

[153] Ruiz, D. S., Morales, S. F., and Molina, J. G. (2015). Inferring versioned schemas from nosql databases and its applications. In *Conceptual Modeling - 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings*, pages 467–480. Springer.

[154] Safavian, S. R. and Landgrebe, D. A. (1991). A survey of decision tree classifier methodology. *IEEE Trans. Systems, Man, and Cybernetics*, 21(3):660–674.

[155] Sapia, C. (2000). PROMISE: predicting query behavior to enable predictive caching strategies for OLAP systems. In *Data Warehousing and Knowledge Discovery, Second International Conference, DaWaK 2000, London, UK, September 4-6, 2000, Proceedings*, pages 224–233. Springer.

[156] Sarawagi, S. (2000). User-adaptive exploration of multidimensional data. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 307–316. Morgan Kaufmann.

[157] Sarawagi, S. and Sathe, G. (2000). i^3: Intelligent, interactive investigaton of OLAP data cubes. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, page 589. ACM.

[158] Selkow, S. M. (1977). The tree-to-tree editing problem. *Inf. Process. Lett.*, 6(6):184–186.

[159] Shannon, C. E. (2001). A mathematical theory of communication. *Mobile Computing and Communications Review*, 5(1):3–55.

[160] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

[161] Stavrakantonakis, I., Gagiu, A.-E., Kasper, H., Toma, I., and Thalhammer, A. (2012). An approach for evaluation of social media monitoring tools. *Common Value Management*, 52(1):52–64.

[162] Stillger, M. and Freytag, J. C. (1995). Testing the quality of a query optimizer. *IEEE Data Eng. Bull.*, 18(3):41–48.

[163] Stonebraker, M. (2012). What does 'big data' mean? http://cacm.acm.org/blogs.

[164] Taboada, M., Brooke, J., Tofiloski, M., Voll, K. D., and Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

[165] Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., and Decker, S. (2010). Sig.ma: Live views on the web of data. *J. Web Sem.*, 8(4):355–364.

[166] Turricchia, E. (2013). *Pervasive Business Intelligence.* PhD thesis, University of Bologna.

[167] Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning*, 4:161–186.

[168] Vaisman, A. A. and Zimányi, E. (2014). *Data Warehouse Systems - Design and Implementation.* Data-Centric Systems and Applications. Springer.

[169] van Rijsbergen, C. J. (1979). *Information Retrieval.* Butterworth.

[170] Varga, J., Romero, O., Pedersen, T. B., and Thomsen, C. (2014). SM4AM: A semantic metamodel for analytical metadata. In *Proceedings of the 17th International Workshop on Data Warehousing and OLAP, DOLAP 2014, Shanghai, China, November 3-7, 2014*, pages 57–66. ACM.

[171] Vrdoljak, B., Banek, M., and Rizzi, S. (2003). Designing web warehouses from XML schemas. In *Data Warehousing and Knowledge Discovery, 5th International Conference, DaWaK 2003, Prague, Czech Republic, September 3-5,2003, Proceedings*, pages 89–98. Springer.

[172] Wang, K. and Liu, H. (1997). Schema discovery for semistructured data. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, pages 271–274. AAAI Press.

[173] Wang, L., Hassanzadeh, O., Zhang, S., Shi, J., Jiao, L., Zou, J., and Wang, C. (2015). Schema management for document stores. *PVLDB*, 8(9):922–933.

[174] Wang, Q. Y., Yu, J. X., and Wong, K. (2000). Approximate graph schema extraction for semi-structured data. In *Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings*, pages 302–316. Springer.

[175] Wang, X., McCallum, A., and Wei, X. (2007). Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 697–702. IEEE Computer Society.

[176] Woods, D. (2011). Big data requires a big, new architecture. http://www.forbes.com/sites/ciocentral/2011/07/21/big-data-requires-a-big-new-architecture/#48f109291d75.

[177] Wu, X., Zhu, X., Wu, G., and Ding, W. (2014). Data mining with big data. *IEEE Trans. Knowl. Data Eng.*, 26(1):97–107.

[178] Yi, J., Nasukawa, T., Bunescu, R. C., and Niblack, W. (2003a). Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 427–434. IEEE Computer Society.

[179] Yi, L. and Liu, B. (2003). Web page cleaning for web mining through feature weighting. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 43–50. Morgan Kaufmann.

[180] Yi, L., Liu, B., and Li, X. (2003b). Eliminating noisy information in web pages for data mining. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 296–305. ACM.

[181] Zhang, D., Zhai, C., and Han, J. (2009). Topic cube: Topic modeling for OLAP on multidimensional text databases. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 1124–1135. SIAM.

[182] Zhang, Y., Zhang, G., Chen, H., Porter, A. L., Zhu, D., and Lu, J. (2016). Topic analysis and forecasting for science, technology and innovation: Methodology with a case study focusing on big data research. *Technological Forecasting and Social Change*, 105:179–191.

[183] Zhou, X., Hu, X., Zhang, X., Lin, X., and Song, I. (2006). Context-sensitive semantic smoothing for the language modeling approach to genomic IR. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 170–177. ACM.

Appendix A

iMOLD glossary

Appendix iMOLD

The glossary for the variables used in the SPARQL queries and the IO are shown in Tables A.1 and A.2, respectively.

Table A.1 Variables used in the SPARQL queries for detecting patterns

Name	Type	Pattern	Value
?c	input	ass., gen.	URI of c
?maxCard	input	ass.	parameter $maxCard$
?offset	input	ass.	query offset
?p	output	ass.	a property linked to c
?class	output	ass.	a class to which the instances o belong
?rightCard	output	ass.	right cardinality of a
?leftCard	output	ass.	left cardinality of a
?nO	output	ass.	range cardinality, i.e., number of distinct instances of $class$ involved in p
?nS	output	ass.	domain cardinality, i.e., number of distinct instances of c involved in p
?type	output	gen.	the superclass of c
?s	auxiliary	ass.	an instance of c
?o	auxiliary	ass.	an instance linked to s through p
?nProp	auxiliary	ass.	number of properties p connecting s to o

Table A.2 Glossary for the classes and properties in the Internal Ontology

IO area	URI	Definition
MK	qb4o:DimensionProperty	Defines a dimension (e.g., the Time dimension)
MK	qb4o:Hierarchy	Defines a hierarchy within a dimension; a dimension may contain many hierarchies (e.g., TimeByWeeks, TimeByMonths)
MK	qb4o:LevelProperty	Defines a generic level (e.g., Day)
MK	qb4o:HierarchyStep	Defines a roll-up relationship between two instances of qb4o:LevelProperty
MK	imold:asMembersHasInstancesOf	Links an instance of qb4o:LevelProperty to one or more classes in the EO; it states that the members of the level are the instances of the linked classes.
MK	imold:asMembersHasSubClassesOf	Links an instance of qb4o:LevelProperty to one or more classes in the EO; it states that the members of the level are the subclasses of the linked classes
MK	imold:correspondsTo	Links a qb4o:HierarchyStep to the corresponding property in the EO; it is also used to link a qb4o:LevelMember to the corresponding owl:Thing (either a class or an instance) in the EO
MK	imold:classMetadata	Allows to annotate all necessary metadata regarding the class in the EO referenced by the qb4o:LevelProperty (e.g., the source ontology)
MK	imold:propertyMetadata	Allows to annotate all necessary metadata regarding the property in the EO referenced by the qb4o:HierarchyStep
UK	imold:User	(rdf:type sm4am:User) Lists the users of the company
UK	imold:LevelPreference	(rdf:type sm4am:UserAction) Each instance represents the preference of one or more users towards a specific qb4o:LevelProperty
UK	imold:RollupPreference	(rdf:type sm4am:UserAction) Each instance represents the preference of one or more users towards a specific qb4o:HierarchyStep
UK	imold:isLevelPreferenceOf	(rdf:type sm4am:usesSchemaComponent) Links an imold:LevelPreference to the referred qb4o:LevelProperty
UK	imold:isRollupPreferenceOf	(rdf:type sm4am:usesSchemaComponent) Links an imold:RollupPreference to the referred qb4o:HierarchyStep
UK	imold:byUser	(rdf:type sm4am:byUser) Links the instances of imold:LevelPreference and imold:RollupPreference to the various imold:Users that expressed them
—	sm4am:usesSchemaComponent	Meta-property to link SM4AM instances to QB4OLAP instances
—	sm4am:User	Meta-class to represent users
—	sm4am:UserAction	Meta-class to represent user actions
—	sm4am:byUser	Meta-property to link a sm4am:UserAction to its sm4am:User