Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 33

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

AUGMENTING THE KNOWLEDGE PYRAMID WITH UNCONVENTIONAL DATA AND ADVANCED ANALYTICS

Presentata da: Matteo Francia

Coordinatore Dottorato

Davide Sangiorgi

Supervisore

Matteo Golfarelli

Co-supervisore

Stefano Rizzi

Esame finale anno 2021

“Tell me and I forget, teach me and I
may remember, involve me and I learn”

To my parents.

Acknowledgements

I would like to express my sincere gratitude to my advisors Prof. Matteo Golfarelli and Prof. Stefano Rizzi for their support and inspiration. My Ph.D. has been a joyful journey under their supervision.

My thanks go to Prof. Xiaofang Zhou, who provided me the incredible opportunity to join the Data Science Research Group at The University of Queensland, Australia. I wish to extend my special thanks to Verónika Peralta and Sandro Bimonte since their insightful comments improved the quality of this thesis.

I thank my labmates for the stimulating discussions, especially Dr. Enrico Gallinucci for his guidance and for the great amount of work we did together. Also, I thank my friends at the University of Bologna for all the fun we had in the last years.

I thank my family, a never-ending source of motivation and courage.

Abstract

The volume, variety, and high availability of data backing decision support systems have impacted on business intelligence, the discipline providing strategies to transform raw data into decision-making insights. Such transformation is usually abstracted in the "knowledge pyramid," where data collected from the real world are processed into meaningful patterns. In this context, volume, variety, and data availability have opened for challenges in augmenting the knowledge pyramid. On the one hand, the volume and variety of *unconventional* data (i.e., unstructured non-relational data generated by heterogeneous sources such as sensor networks) demand novel and type-specific data management, integration, and analysis techniques. On the other hand, the high availability of unconventional data is increasingly attracting data scientists with high competence in the business domain but low competence in computer science and data engineering; enabling effective participation requires the investigation of new paradigms to drive and ease knowledge extraction. The goal of this thesis is to augment the knowledge pyramid from two points of view, namely, by including unconventional data and by providing advanced analytics. As to unconventional data, we focus on mobility data and on the privacy issues related to them by providing (de-)anonymization models. As to analytics, we introduce a higher abstraction level than writing formal queries. Specifically, we design advanced techniques that allow data scientists to explore data either by expressing intentions or by interacting with smart assistants in hand-free scenarios.

Table of contents

Chapter 1

The Knowledge Pyramid

Data science extracts actionable insights from raw data [153]. The data transformation process is usually abstracted in the "knowledge pyramid" (also known as the "Data Information Knowledge Wisdom pyramid" or "knowledge hierarchy"; Figure 1.1), where data (i.e., symbols) are collected through measurements taken from the real world; information is processed and linked data that it is meaningful to scientists; knowledge is interpreted, understood, and organized information; and wisdom is knowledge in action. Climbing the pyramid is not a straightforward path and requires the iterative exploration and preparation of data so that patterns can be extracted, evaluated, and later deployed into models supporting effective decisions.

Analytic applications strive to extract knowledge from data fueled by pervasive systems [71], where any device can be turned into a sensor leading to a huge volume and variety of available data [277]. These *unconventional*[1] data (i.e., unstructured and non-relational data) have impacted on business intelligence (BI)—the discipline providing strategies and technologies to transform data into decision-making information—leading to BI 2.0, where decisions are drawn *not only* on the data owned by the organization. Indeed, bigger data volumes lead to a holistic view of historic and current trends; higher data velocities ground decisions in continuously updated data, and broader data varieties provide many nuances of the matter at hand [1]. On the one hand, volume and variety hinder the management, the integration, and the analysis of the collected data (from "World" to "Data" in Figure 1.1), since each instance of unstructured data might have a different structure. This requires ad-hoc techniques for different data types (e.g., social networks [87] or sensor data [92]). On the other hand, high availability has attracted scientists with no expertise in computer science or data engineering, requiring novel paradigms to support the "Data"-to-"Knowledge"

[1] We consider *conventional* the data collected by operational databases, ERP (enterprise resource planning), and CRM (customer relationship management) enterprise systems

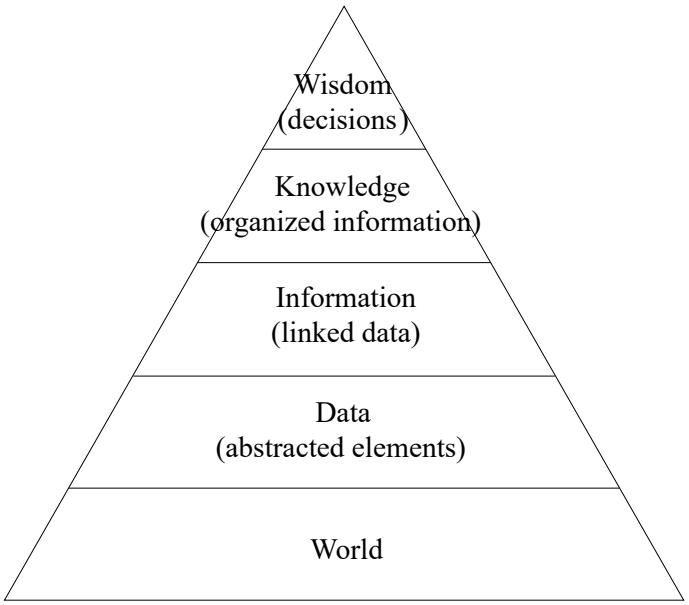


Figure 1.1: The "knowledge pyramid" (also known as the "Data Information Knowledge Wisdom pyramid" or "knowledge hierarchy"; adapted from [259]).

transformation with a higher abstraction level than formal programming languages (e.g., through graphical metaphors, recommender systems, or automatic transformation pipelines). Additionally, availability has enabled pervasive analyses, allowing data scientists to access data in hand-free contexts involving augmented reality and smart assistants.

While investigating these challenges, this thesis evolves into two parts.

Part I: Unconventional Data. Sensing provides real-time data upon which "contextual" decisions—ranging from user-centered to societal problems—are based. This includes data collected from human and technological assets (e.g., sensor or mobile networks), which are analyzed to monitor and manage urban and rural areas. Sensor data are highly available due to the growth of Internet of Things devices, have variable content, and their value comes from historic trends that range from hours to years [153]. Such volume and variety demand for augmenting the knowledge pyramid with novel and scalable techniques for different data types (Figure 1.2). In urban mobility, spatiotemporal data (i.e., temporal sequences of spatial locations traced by moving objects and sampled through global or relative positioning sensors) are collected and processed for the sake of traffic analysis and forecasting, clustering of objects moving in similar paths, and habits profiling. Spatiotemporal data are challenging because of their uncertain, sparse, and multiresolution nature [106]. Also, due to the number of moving objects (e.g., mobile devices, cars, taxis, and public transport in a metropolitan area) and to the sampling rate of positioning sensors (e.g., from minutes to seconds), mining

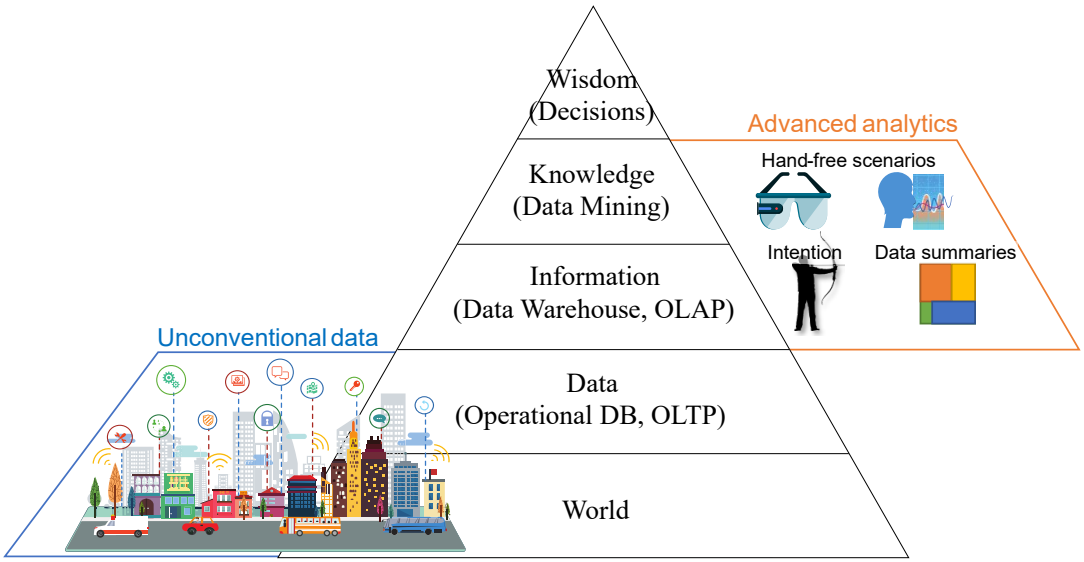


Figure 1.2: Augmenting the knowledge pyramid with unconventional data (left) and advanced analytics (right).

mobility data easily scale up to big-data problems that require big-data solutions, hence introducing new business opportunities that were previously ignored because of technology limitations. Besides the valuable knowledge, due to high uniqueness [64] and sensitivity (e.g., home and work locations), trajectory data expose individuals to privacy violations, demanding for ad-hoc techniques to (de-)anonymize historical trajectory datasets. In this direction, Part I focuses on trajectory data and on the privacy implications of the publication of long-term mobility datasets.

Part II: Advanced Analytics. Since the introduction of the relational model in the '70s, users used relational queries (e.g., SQL queries) to retrieve data collected in operational databases. This requires a good comprehension of programming languages and database management systems. Later, more user-friendly abstractions and tools provided a simpler view of the data, hiding the complexity of the underlying databases [275] and transitioning from static (repetitive) workloads involving a few records (On-Line Transaction Processing, OLTP) to dynamic workloads (On-Line Analytical Processing, OLAP, and On-Line Analytical Mining, OLAM) involving a huge amount of records (Figure 1.2). The spread of data and analytical tools at hand has brought an increasing participation of data scientists with high competence in the business domain but low competence in computer science and data engineering [275]. Indeed, data science emerged as an amalgamation of domain expertise with disciplines such as statistics, data mining, and databases [272]. Such amalgamation has brought challenges not only concerning big data volumes but also in terms of the in-

creasing complexity and interdisciplinarity of the analytic questions. Enabling an effective participation in data science requires the investigation of user-centered paradigms supporting analytical querying and making knowledge extraction more accessible. Data scientists can benefit from proactive systems that "understand" the tasks at hand, make recommendations, and generate effective visualizations [106]. For instance, in the digital twin scenario [264] where physical entities are mapped into a digital world, the synergy of personal assistants and augmented reality lacks analytic capabilities. Additionally, limited attention has been devoted to providing analytical reports that can be useful to let the user compare the current behavior of the visualized objects with their historical behavior. To this end, unconventional data sources, such as smart glasses and wearable devices, can be accessed by personal assistants (e.g., recommender systems) to address users' needs [21]. Data scientists can interact with personal assistants through natural language interfaces which provide a higher abstraction level than formal queries and programming languages. In this direction, Part II focuses on supporting data scientists with higher analytic abstractions than formal queries also in scenarios entailing pervasive data access.

Part I

Unconventional Data

Chapter 2

Introduction

2.1 Motivation and contributions

Following the spread of positioning systems and mobile devices, mobility data are at the core of location-based decision support systems and augment the knowledge pyramid (Figure 1.1) with novel storage, integration, and analysis techniques specific to spatiotemporal data. Examples of valuable applications are urban-mobility analysis, point-of-interest recommendation, and personal navigation systems [157].

Mobility data are described along spatial and temporal dimensions and are also known as trajectory data (or spatiotemporal data). A spatiotemporal entity is a moving object (e.g., an individual, an animal, a car) characterized by its position in space that varies over time. Position can be recorded by positioning sensors (e.g., a traffic monitor collecting passing vehicles), by issuing geolocalized queries to location-based systems (e.g., return the closest restaurant to my location), by social check-ins (e.g., foursquare), or by wearable or smart devices with positioning technologies (e.g., GPS). Such heterogeneity demands for ad-hoc analytic techniques and has driven an intensive research activity [310].

Definition 1 (Trajectory) *The* trajectory *(or trace) of a spatiotemporal entity (or "moving object", interchangeably) represents its entire movement history. It is a sequence of spatiotemporal points, denoted as* $T = \langle p_1, p_2, \ldots, p_n \rangle$ *where each* $p = (x, y, t)$ *consists of a location* (x, y) *(e.g., longitude and latitude) at time* t*. Points in a trace are organized chronologically. The set of trajectories is denoted by* $\mathscr{T}$*.*

Example 1 (Trajectory) *With reference to Figure 2.1,* T_r, T_g, T_b *are examples of trajectories in a two-dimensional space, with* $\mathscr{T} = \{T_r, T_g, T_b\}$*.* □

In this context, we contribute to the following research directions.

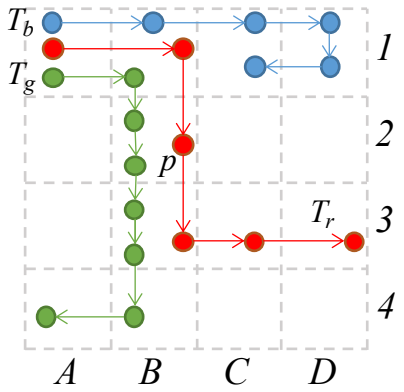


Figure 2.1: Example of trajectories in a two-dimensional space; trajectory points are organized chronologically.

Scaling up to big data analytics. The volume of trajectory datasets is always increasing, demanding big data solutions supporting the processing of trajectory data. In urban mobility, map-matching aims to project GPS points generated by moving objects onto the road segments representing the actual object positions. Up to now, map-matching has found interesting applications in traffic analysis, frequent path extraction, and location prediction. However, state-of-art implementations of map-matching algorithms are either private, sequential, or inefficient. We propose an extension of an existing centralized algorithm of known efficiency by reformulating it in a distributed way, in order to achieve great scalability on real big data scenarios [92]. Furthermore, we enhance the robustness of the algorithm, which is based on a first-order Hidden Markov Model, by introducing a smart strategy to avoid gaps in the matched road segments; indeed, this problem may occur under sparse GPS sampling or in urban areas with highly fragmented road segments.

(De-)Anonymization of trajectory data. The ubiquity of mobility data can benefit various real-world applications such as traffic management and location-based services. However, trajectories may disclose highly sensitive information of an individual including identity [149], personal profiles [91], and social relationships, making it indispensable to consider privacy protection when releasing trajectory data. Ensuring privacy on trajectories demands more than hiding single locations since trajectories are intrinsically sparse and high-dimensional, and requires to protect multi-scale correlations. To this end, extensive research has been conducted to design effective techniques for privacy-preserving trajectory data publishing. Furthermore, protecting privacy requires to carefully balance two metrics: privacy and utility. In other words, it needs to protect privacy as much as possible and meanwhile guarantee the usefulness of the released trajectories for data analysis. We provide a comprehensive study

of the existing protection models. We also conduct experiments in de-anonymization privacy models.

2.2 Structure

This part of the thesis hinges on the above-mentioned contributions and is organized as follows. In Chapter 3, we design a map-matching algorithm in Spark, a big data framework. Chapter 4 introduces the sensitivity of mobility data (e.g., the exposure of individual identity and frequented locations) and surveys the existing approaches addressing the privacy-preserving publication of trajectory data. Chapter 5 introduces DART, an algorithm exploiting personal gazetteers (i.e., the set of relevant places in everyday life such as home and work locations) to de-anonymize individuals while scaling up to big data problems. Finally, Chapter 6 draws the conclusion and future research directions.

Chapter 3

Map-Matching on Big Data

3.1 Introduction

Following the spread of mobile devices and enhancements in positioning systems, trajectory data has gained much attention. Among the plethora of applications suitable for trajectory mining [310], map-matching projects inaccurate Global Positioning System (GPS) points generated by objects moving in urban areas (Figure 3.1) onto the road segments representing the actual object positions (Figure 3.2). Up to now, map-matching algorithms had interesting applications, such as predicting moving object (MO) locations [200], generating traffic analysis models [110, 126], and uncovering frequent routes [164].

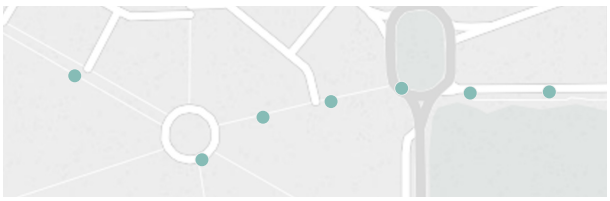


Figure 3.1: GPS points of a trajectory in the road network of Milan, Italy.

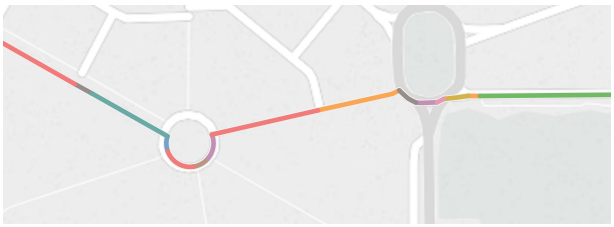


Figure 3.2: Sequence of road segments matched to the trajectory in Figure 3.1; colors represent different segments.

Despite its straightforward applications, map-matching is not easily addressable due the articulated topology of urban networks (parallel roads, roundabouts, road crosses, as seen in Figure 3.1). Over the years, several models capturing increasingly complex matching features have been proposed. Zehng categorizes four map-matching approaches [310]: geometric (e.g., mapping GPS onto the closest road segment), topological (e.g., mapping relies on local connections of adjacent segments), probabilistic (e.g., mapping is about finding the most likely path under noisy conditions), and advanced (e.g., mappings relies on mixing the above techniques).

Among the advanced map-matching methods, Luo et al. [181] propose a promising map-matching algorithm based on first-order Hidden Markov Model (HMM) to find the sequence of roads segments with highest probability of being the exact path traveled by a MO. In their experimental evaluation, the authors demonstrated that the proposed solution achieves good accuracy performance. However, when it comes to real big data applications accuracy is not all. Indeed, Peixoto et al. [222] introduce a further categorization of map-matching algorithms with respect to sequential and distributed computation. Peixoto et al. highlight that although sequential algorithms achieve high accuracies, they are capable to process only few trajectories at a time. Hence, they do not account for scalability.

In this chapter, we propose a distributed extension of the sequential algorithm proposed by Luo et al. [181], overcoming its limitations in terms of: (1) distributed, scalable and open-source implementation; (2) higher robustness in case of highly fragmented urban maps; and (3) the inference of MO movement in trajectories with silence periods (i.e., GPS pings are missing or compromised for short time periods). Although distributed algorithms for map matching do exist, their implementations are either private or less accurate than sequential models. Our implementation is based on the Apache Spark engine [301] and leverages the GeoSpark [298] library. Due to its execution engine and in memory computation, Spark achieves higher performance than MapReduce [65].

The remainder of the chapter is organized as follows. We introduce the formal background on map-matching in Section 3.2 and our distributed implementation in Section 3.3. In Section 3.4, we test our algorithm against a real big data case study in the Italian city of Milan. In Section 3.5, we discuss a summary of the state-of-art related contributions to our work. Finally, in Section 3.6, we summarize future works to extend our contribution.

3.2 Formal foundation

A Hidden Markov Model (HMM) is a temporal and finite model that describes a probability distribution over an infinite number of sequences [78]. The HMM is composed of hidden

states that emit observable symbols; the goal is to find the most likely sequence of hidden states depending on the observed symbols. The *symbol emission probability* (or simply *emission probability*) determines the probability of a certain state emitting a certain symbol, while the *state-transition probability* (or simply *transition probability*) determines the probability of moving from one state to another. Given an initial state, a *state sequence* is generated by moving from state to state. The sequence of states is a Markov chain, i.e. the transition to the next state only depends on the current state. The state sequence is hidden (i.e., it is not directly observable), while the sequence of symbols generated by the hidden states is observed. One of the applications of HMMs is to uncover the most likely state sequence that generates the observed symbols. Map-matching can be modeled as an HMM, where: states correspond to road segments and symbols correspond to GPS points; the emission probabilities depend on the geometrical relationships between GPS points (i.e., the symbols) and close road segments (i.e., the states), while the transition probabilities reflect topological relationships between consequent segments. Finally, the HMM identifies the optimal sequence of road segments that most likely emits the sequence of observed GPS points. To define emission and transition probabilities of the HMM, we recall the definition of (GPS) trajectory from Definition 1 and formalize the concept of road network.

Definition 2 (Road Network) *A road network is a directed graph $G(V,S)$, where S is the set of road segments connecting the set of terminal points V. A route is a sequence of consequent road segments.*

Following the notation provided in [237], let us denote with X_t the index of the road segment in which a MO is possibly located at time t (i.e., $X_t = [1, |S|]$). With a slight abuse of notation, we denote with s_i the i-th segment in S. Then, we define the emission and transition probabilities as follows.

Definition 3 (Emission Probability) *Given the set of road segments S, and the MO position p_t at time t, the emission probability $P(p_t|X_t = i)$ is the probability of a state i to generate the observation p_t:*

$$P(p_t|X_t = i) = \frac{d_H(p_t, s_i)^{-1}}{\sum_{s_j \in N(p_t,S,\alpha,\tau)} d_H(p_t, prj(p_t, s_j))^{-1}}$$

$N(p_t, S, \alpha, \tau)$ is the operation that returns the *neighborhood* of a point p_t, i.e., the set of α closest segments in S whose distance to p_t is less than a spatial threshold τ; d_H is the great-circle distance[1] between the point p_t and its projection $prj()$ to the road segment s_i.

[1]The great-circle distance is the shortest distance between two points on the surface of a sphere, measured along the surface of the sphere.

Since road segments are quite dense in urban areas, the road network topology provides constraints to the map-matching process (i.e., a MO cannot "jump" from road one segment to another if they are not somehow connected), thus it should be taken into consideration to define the transition probability. In [181], the authors estimate the (global) average road segment length (i.e., 119 meters) and the (global) average distance between successive GPS pings (i.e., 21 meters) to estimate the transition probability on same road segments (estimated as 3/5), on adjacent road segments (estimated as 2/5), segments separated by one road segment (estimated as 1/5), and 0 otherwise. However, this probability distribution performs poorly on road networks with high variance on segment length (e.g., a road network including highly segmented urban roads and long highway segments) and/or on high variance on GPS sampling rates. Additionally, the higher the resolution of the road network, the worst the transition probability represents the network topology. For instance, a roundabout with 3 exits is modeled with at least 6 segments (i.e., 3 segments for the roundabout and 3 for the outgoing roads). As a consequence, if the sampling between consequent GPS pings is not high enough (e.g., 10 to 20 seconds), then the transition probability assigned in [181] is 0 even if the two points are close but farther than 3 segments in the road network. Thus, while Definition 3 is compliant to [181], we adopt the transition probability from [212].

Definition 4 (Transition probability) *Given the set of road segments S, the state-to-state transition $P(X_t|X_{t-1})$ is a $S \times S$ matrix, where $P(X_t = j|X_{t-1} = i)$ is the transition probability from segment i (i.e., the state at time $t-1$) to segment j (i.e., the state at time t)*

$$P(X_t = j|X_{t-1} = i) = \frac{1}{\beta} e^{-d/\beta}$$

where

$$d = |d_H(p_t, p_{t-1}) - d_R(prj(p_t, s_j), prj(p_{t-1}, s_i))|$$

prj projects p_t and p_{t-1} to road segments s_j and s_i, d_R is the length of the shortest-path between two points, and β is a smoothing parameter [212]. In other words, given two GPS points p_t and p_{t-1}, the transition probability is defined as the difference between the great circle distance between the two points $d_H(p_t, p_{t-1})$ and the road-network based distance between their projections onto segments of the road network $d_R(prj(p_t, s_j), prj(p_{t-1}, s_i))$. In fact, in the actual routes there is typically a small difference between d_H and d_R [212].

The formal definition of the map-matching process is summarized in Algorithm 1.

Algorithm 1 Map-Matching Algorithm

Require: T: trajectory, $G(V,S)$: road network, β: prob. smoothing, α: nearest neighbor card., τ: spatial thr., γ: max. route distance, θ: max. route depth
Ensure: M: map-matched trajectory
$routes \leftarrow$ routes of max. distance γ and max. depth θ ▷ pre-computed to determine d_R in transition prob.
$I \leftarrow 1/|S|, \forall s_i \in S$ ▷ All segments are equally probable to p_0
$O \leftarrow P(p_t|X_t = i), \forall p_t \in T, \forall s_i \in N(p_t, S, \alpha, \tau)$ ▷ emission prob. vector
$R \leftarrow P(X_t = j|X_{t-1} = i), \forall p_t, p_{t-1} \in T, \forall s_j \in N(p_t, S, \alpha, \tau), \forall s_i \in N(p_{t-1}, S, \alpha, \tau)$ ▷ transition prob. matrix
$M \leftarrow Viterbi(S, T, I, R, O)$ ▷ Viterbi (state space; trajectory; transition, initial, and emission prob.)
return M

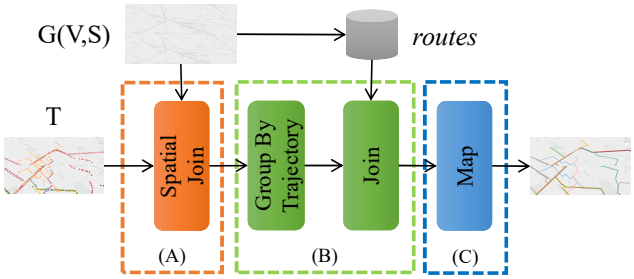


Figure 3.3: Main transformations involved in the Map Matching process with respect to steps (A), (B) and (C).

3.3 Algorithm implementation

The algorithm consists of a distributed implementation of [181], which is depicted in Figure 3.3. The algorithm undergoes the following steps for each trajectory in the dataset: (A) computation of the emission probability, (B) computation of the transition probability, and (C) computation of the result by means of the Viterbi algorithm [84] to infer the optimal sequence of hidden states that generates a sequence of observable symbols. As the Viterbi algorithm map trajectories onto possibly fragmented routes, we embed in this step an aiding process similar to [181]. With respect to the sequential implementation provided in [181], we design the map-matching algorithm in terms of Spark's and GeoSpark's Resilient Distributed Datasets (RDDs) in order to distribute and parallelize the computation of the result for each trajectory.

Our implementation is publicly available at http://www.github.com/big-unibo/map-matching. The adopted versions of Spark and GeoSpark are 2.1 and 1.1.3, respectively.

3.3.1 Emission probability

Given a GPS point p_t (i.e., an observable symbol in the HMM), the estimation of its emission probability requires the computation of the neighborhood $N(p_t, S, \alpha, \tau)$ for each point in $\mathscr{T}$. This requires a spatial *range-join* (step (A) in Figure 3.3), a spatial operation that combines two datasets with respect to their element-wise spatial distance (e.g., finding, for each observed GPS point, the nearby road segments). Note that this transformation is applied to all GPS points in $\mathscr{T}$ independently from the trajectory they belong to, in order to achieve the maximum degree of parallelism. We rely on GeoSpark's spatial RDDs (SRDDs) to distribute and range-join the sets of GPS points and road networks; the result is the set of α road segments closer to p_t than a spatial threshold τ. The efficiency of the spatial operation is affected by τ, as it sensibly reduces the algorithmic search space, whereas α simply determines the size of the output. The setting of these parameters depends on the cluster specifics: the higher τ and α, the greater the number of candidate segments to be considered (and finally returned) for each point.

3.3.2 Transition probability

Among S there exist unlikely segment transitions (e.g., segments too far for a correct match, or connected by multiple paths with sensibly different lengths). To lighten the workload, we prevent the computation of transition probabilities that are close to zero (according to Definition 4): given two GPS points p_t and p_{t-1}, we consider as candidate states of the respective points only the road segments that are closer than τ; also, from the list of candidate pairs of segments, we prune those connected by a shortest-route with depth (i.e., the number of segments) higher than θ or a distance on the road network (i.e., d_R) higher than γ. While γ bounds the routes potentially producing transition probabilities close to 0, θ lighten the process of the route creation by preventing the computation of long sequences of segments. Note that the computation of deep routes is necessary in case of highly fragmented urban areas (Figure 3.2).

To distribute the computation of the transition probability (step (B) in Figure 3.3), given the result of step (A), we group GPS points by trajectory, we create a sliding window with size 2 on each trajectory (basically, we build pairs of p_t and p_{t-1}), and we compute the transition probability $P(X_t = j | X_{t-1} = i)$ for each pair of candidate segments s_j and s_i, with $s_j \in N(p_t, S, \alpha, \tau)$ and $s_i \in N(p_{t-1}, S, \alpha, \tau)$; note that the information about the neighborhood is kept from step (A). While the estimation of d_H in Definition 4 is straightforward, d_R requires to compute point-wise distance in the road network. To do so, all routes (i.e., sequences of road segments) in $G(V,S)$ with a maximum distance of γ and a maximum depth of θ

must be computed; then, d_R is obtained by finding the shortest route for each s_j and s_i pair. Interestingly, as the routes only depend on $G(V,S)$, they can be computed once, persisted in the distributed file system, and then retrieved on demand.

3.3.3 Viterbi computation

For a trajectory T with $|T|$ GPS points, we could naively compute $S^{|T|}$ sequences to find the optimal route generating the observed GPS points. However, this solution has exponential complexity. The Viterbi algorithm is a recursive (optimal) estimation of the state sequence of a time-discrete Markov process [84] (i.e., a sequence of events in which the probability of an event depends only on the state of the previous event) with a polynomial complexity $O(|S|^2|T|)$. In other words, the Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states of a finite-state machine that results in a sequence of observed events.

When applied to map-matching, the algorithm returns the sequence of road segments (i.e., a route) that, with the highest likelihood, generates the observed GPS trajectory (step (C) in Figure 3.3). Intuitively, the Viterbi algorithm detects the most likely transition that leads into each state at time t, it discards the other transitions into that state, and, when the algorithm reaches the last timestamp, it determines the most-likely path by concatenating the states connected by the most-likely transitions. The input parameters required by the algorithm are: the trajectory T; the segment (i.e., state) space S; the emission probability matrix O (computed for each point as in Definition 3); the initial probability vector I, where $|I| = |S|$ and $I_i = 1/|S|$ for each segment in S (we consider all the segments equally probable with respect to the initial GPS point); and the transition probability matrix R (computed for each pair of consequent trajectory points as in Definition 4).

Intuitively, applying the Viterbi algorithm to T returns the route onto which T is map-matched by associating to each GPS point an optimal road segment. However, there is no insurance of consequent GPS points being mapped to adjacent road segments, possibly resulting in fragmented optimal routes. We embed in the Viterbi algorithm an aiding process to fill the gaps. In each iteration over two consequent points p_t and p_{t-1}, if the shortest route between the mapped segments s and s' is shorter than γ and has a smaller depth than θ, we map p_{t-1} also to all the segments included in the shortest path between s and s'. To do so, we leverage the possible route that has been pre-computed as in (B). With respect to [181], the advantage of embedding the aiding process in this step of computation (rather than after computing Viterbi) prevents an unnecessary further iteration on the map-matched trajectories.

The computational complexity of the Viterbi algorithm can be tackled down by addressing both $|T|$ and $|S|$. As to $|T|$, reducing the trajectory length requires a simplification process that is applicable before the map-matching process (hence is out of the scope of this chapter). As to $|S|$, the segment space can be sensibly pruned assuming that, among all the roads segments in S, only the segments close to the observed points contribute to the matching process with a non-null probability. For each point p_t, we only consider the candidate segments in its neighborhood $N(p_t, S, \alpha, \tau)$, where α (i.e., the number of retrieved neighbors) constrains the Viterbi search space).

Finally, rethinking the Viterbi algorithm is a distributed environment is not straightforward due to its recursive nature. In this chapter, we limit to apply Viterbi in a *bag-of-task* fashion, i.e., executing the algorithm to concurrently map each trajectory onto its road network projection (step (C) in Figure 3.3).

3.4 Evaluation

In our contribution, we engineer the sequential algorithm proposed in [181] in terms of a distributed paradigm. Note that, while the distributed implementations described in Section 3.5 approximate the sequential map-matching algorithms, we entirely preserve the accuracy of the implementation proposed in [181]. Although the setting of the parameters does have an impact on the accuracy, the lower bounds that we consider (with reference to Table 3.2) are higher than the values defined in [181], thus guaranteeing the same minimum accuracy. For this reason, the evaluation of our implementation is focused on the efficiency on big data in a real case study. We run our test on a cluster composed by 11 nodes, each equipped with an 8-core i7 CPU @ 3.60GHz and 32 GB of RAM and interconnected by Gigabit ethernet.

3.4.1 Dataset

We test our implementation against a real-case study scenario in the Italian city of Milan. The dataset includes 120,000 trajectories ($\mathscr{T}$) that contains an average number of 65 GPS points (resulting in an overall number of 7.8 million of GPS points). The distance between consequent GPS points is on average of 99 meters, and the average sampling rate is of 0.14Hz (i.e., 7 seconds). These values, with their respect variances are summarized in Table 3.1.

Our dataset includes ground truth on 50 trajectories. Ground truth data is leveraged to test the accuracy of the algorithm with respect to α, τ, γ, θ. In addition, we used the 50 trajectories to estimate $\beta = 10$ (as required by [212]).

Table 3.1: Summary of the dataset features

Feature	**Value**
GPS points	$7.8 \cdot 10^6$
Trajectories	$12 \cdot 10^4$
Avg. trajectory points	65 ± 42
Avg. ping distance (meters)	99 ± 55
Avg. sampling rate (seconds)	7 ± 6

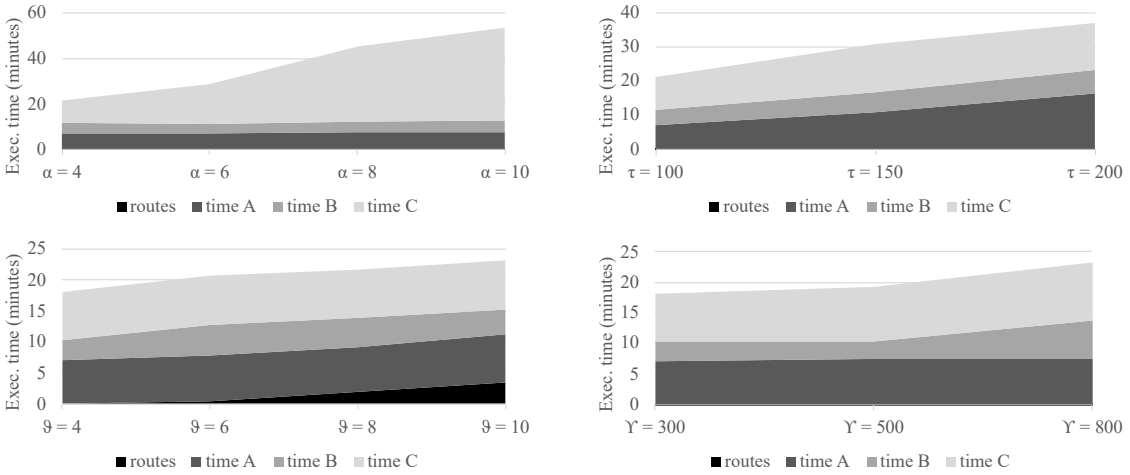


Figure 3.4: Execution times of the different steps of the algorithm under different parameter settings; when not specified, it is $\alpha = 4$, $\tau = 100$, $\theta = 4$, and $\gamma = 300$.

3.4.2 Efficiency

To evaluate our implementation in terms of efficiency, we first measure the impact of each parameter on the execution times of the different steps of the algorithm. We exploit the metrics of Spark's web GUI to obtain the execution times of the pre-computation of the routes and the computation of steps (A), (B) in (C). We start from a base configuration where all parameters are valued at their minimum and, for each parameter, we progressively scale its value up to its maximum. The results (shown in Figure 3.4) reveal that each parameter impacts only on one of the algorithm's steps. Increasing α means producing more candidate segments for each GPS points, thus increasing the complexity of Viterbi (step (C)). A higher value of τ augments the search space of the spatial join to calculate the emission probability (step (A)). Parameter θ increases the complexity of the pre-computation of *routes*; however, since γ is limited to 300, the depth of the *routes* does not actually increase, thus not producing on effect on step (B). Conversely, a higher value of γ produces a greater number of *routes*, which increases the computation of the transition probability (step (B)).

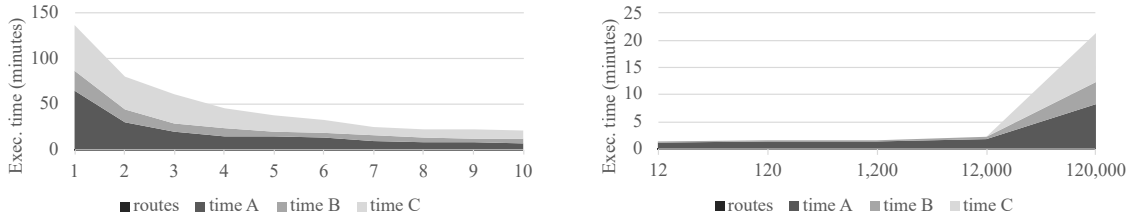


Figure 3.5: Execution times of the different steps by scaling the number of worker nodes (left) and the number of trajectories (right).

Table 3.2: Input parameters; range values are empirically determined

Parameter	Value	Meaning
β	10	Probability smoothing
α	$[4,10]$	Candidate segments
τ	$[100,200]$	Neighborhood thr. (meters)
θ	$[4,10]$	Route max. length
γ	$[300,800]$	Road network distance thr. (meters)

Finally, we evaluate the performance of the algorithm with respect to the computational power of the cluster and the number of trajectories in the dataset; the results are shown in Figure 3.5. In the first case, we scale the number of worker nodes from 1 to 10. The results demonstrate that the algorithm effectively takes advantage of the higher levels of parallelism, reducing the execution times from more than 2 hours to as low as 20 minutes. Consistently with Amdahl's law [17], the speedup is bounded by the serial part of the program and converges to 7x with respect to the sequential execution[2]. Lastly, scaling on the number of trajectories shows that execution times increase linearly with number of trajectories in the last step. The same behavior is not observable in the previous steps due to the overhead of the parallel execution framework, which hides the actual workload times.

3.5 Related works

In this section, we categorize map-matching related contributions with respect to their sequential and distributed natures.

As to sequential implementations, in [179], Lou et al. introduce "ST-Matching" to handle GPS trajectory data at low sampling rates. The idea is to generate all the possible candidate routes on the map and to identify the-most-likely one by calculating spatial and temporal probabilities. However, this approach is particularly sensitive to noise, as common errors

[2]For the fairest comparison, the original sequential implementation should be considered; unfortunately, however, the authors of [181] have not made the code publicly available.

in positioning systems lead to the identification of erroneous paths. Newson et Al. [212] address this problem by adopting a first-order HMM that models the connectivity of the road segments and considers multiple path hypotheses. Further accuracy enhancements are obtained by Lou et al. [181] by leveraging a formulation based on HMM in which emission and transition probabilities depend respectively on geometrical and topological information. The authors leverage the Viterbi algorithm to extract the route that most likely generates a sequence of observed GPS positions. When estimating the state transition probability, the topological information on road segments enhance the matching with a sampling rate up to 0.1Hz (10 seconds).

With the rise of big data, distributed implementations have aimed to optimize the performance of existing map-matching algorithms. However, the sequential nature of these algorithms hinders a trivial reformulation on a distributed environment. For this reason, a recurring problem in distributed implementations is the trade-off between accuracy and efficiency, which is usually in favor of the latter. Almeida et al. [16] leverage the spatial probability from the ST-Matching algorithm [179] and provide a map-matching implementation in the MapReduce paradigm [53]. However, they do not take into account the latest enhancements in terms of HMM and road topology. Zeidan et al. [304] and Peixoto et al. [222] introduce new algorithms that best exploit the characteristics of a distributed environment. In particular, both contributions rely on the construction of spatial indexes that enable an effective partitioning of road segments and trajectories, with the goal to minimize the amount of shuffled data between the cluster nodes. Although these contributions focus on the scalability to big data applications, they usually approximate exact sequential algorithms and lack a sound accuracy comparison against sequential approaches to map matching (whose good accuracies are well known in the literature). Finally, several spatial libraries (e.g., GeoSpark [298], SpatialSpark [296], etc.) have been built on top of Spark's RDD [301]; however, none of these directly addresses map-matching, as they only provide spatial indexing and querying.

3.6 Conclusion

We engineered a Spark-based distributed map-matching algorithm inspired by the sequential implementation provided by Luo et al. [181]. Our algorithm entirely preserves the accuracy of the sequential implementation (which, to the best of our knowledge, achieves a higher accuracy than the existing map-matching algorithms) and it has been tested on a real dataset of 120,000 trajectories (for a total of 7.8 millions GPS points). Also, we overcome the limitations of [181] in case of highly fragmented urban networks.

Chapter 4

Trajectory Privacy

4.1 Introduction

Privacy is usually referred to as the *"ability of an individual to control the terms under which personal information is acquired and used"* [281]. Privacy entails the protection of several data aspects such as collection [293], mining [195], querying [214], and publication [100]. Each of these aspects involves its own privacy protection models as well as measures to evaluate privacy level. We focus on data publication in this chapter, i.e., releasing datasets without leaking any sensitive information. Privacy-preserving data publishing has been extensively studied in the database community, and well-known techniques have been proposed to anonymize tabular records stored in the database including k-anonymity [242, 262] (introduced in 1998), l-diversity [184] (2006), t-closeness [170] (2007), differential privacy [76] (2006), and plausible deniability [30] (2017).

With the increasing popularity of GPS-enabled devices, a wide range of location-based services keeps track of moving objects, resulting in massive available spatial trajectory data. Nowadays, trajectory data analysis has become ubiquitous, as evidenced by a huge amount of trajectory-related techniques, which can benefit various real-world applications including urban planning, traffic management, personalized recommendation. However, the analysis of trajectory data can disclose sensitive information of an individual, making it essential to design techniques for privacy protection. In general, the protection of trajectory privacy is based on two major directions: *location-based services* (LBSs) and *privacy preserving trajectory publication* (PPTD). On the one hand, privacy protection in LBSs requires that a sufficient quality-of-service is ensured while preventing an adversary to learn the exact locations of an individual [54]. On the other hand, privacy concerns hinder data-holders in the publication of private trajectories which, therefore, has spawned extensive research on privacy-preserving trajectory data publishing. These directions are orthogonal and can be

distinguished according to the amount of adversary's knowledge (i.e., a sequence of real-time locations for LBSs; the entire movement history for PPTD) and the protection scope (i.e., the current location for LBSs; the entire trajectory for PPTD). We focus on PPTD in this chapter, considering the proliferation of applications relying on the availability of trajectory data.

Formally, the trajectory of an individual is recorded as a sequence of (geo-position, time) ordered chronologically. Although trajectory data is representable in a tabular format (e.g., organizing each individual and the whole moving history as a record), trajectories cannot be easily anonymized as "classic" tabular data due to the following reasons:

- Trajectory data fulfills spatial constraints (e.g., mobility in an urban area).
- Trajectory locations are not independent (e.g., there is spatiotemporal continuity between adjacent locations; it is impossible to jump from a road to another).
- Although trajectory data is highly sparse, only a few locations can link 95% of individuals [64]. The longer the trajectory, the easier to break individuals' privacy.
- Trajectory locations represent geographical features mappable into semantics (e.g., POIs) that can directly reflect individuals' interests and demographics.
- Trajectories do not have fixed quasi-identifiers [295, 54]. Sensitivity depends on both single locations and arbitrary spatiotemporal patterns (e.g., day and nighttime mobility).

The sensitivity, uniqueness, and low anonymizability of trajectory data raise many issues and concerns, and hence extensive research has been conducted to develop effective techniques for privacy-preserving trajectory data publishing. In 2000s, two main approaches *ad-hoc* for spatiotemporal data were introduced to protected individual locations either by producing dummies locations indistinguishable from the real ones [155] (2005) or by mixing identifiers of individuals entering/leaving mix-zones [217] (2008). Due to the need for publishing trajectories, both dummy and mix-zone models have been adapted to trajectory data. Additionally, since 2008, *generic* privacy models for sequential patterns [262, 184, 170, 76, 30] were also specialized to protect trajectory data, with trajectory k-anonymity being implemented first [210] by making a trajectory indistinguishable in an anonymity group including k-1 other trajectories. Differential privacy has been introduced for trajectory data in [48] (2012), where, rather than generalizing/suppressing locations to achieve k-anonymity, the authors release synthetic trajectories resembling the original ones. Recently, l-diversity, t-closeness [270] and plausible deniability [30] have also been applied to trajectory data to protect semantic locations (e.g., home, work, and frequently-visited pubs).

We analyze and organize the articulated spectrum of threat and anonymization models on the publication of trajectory data. Understanding which anonymization models fulfill the publication requirements is hard especially because ad-hoc privacy and utility metrics are usually leveraged in these papers, requiring exhaustive comparison which is missing in these works. Our goal is to provide a comprehensive and clear overview of the privacy issues related to trajectory data as well as the privacy models countering these issues. Overall, the main contributions are the following.

- We provide a detailed overview of the sensitive information and privacy attacks on trajectory data to highlight the privacy issues related to the publication of trajectory data.
- We conduct a systematic analysis of models applied to trajectory data publishing and the ways to quantify their privacy and utility, based on highly-cited papers found in DBLP, Scopus, and Google Scholar.

The remaining of the chapter is organized as follows: we summarize the privacy threats in Section 4.2 and the state-of-the-art privacy protection models for trajectory in Section 4.3. Finally, we summarize future research direction in Section 4.4.

4.2 Sensitive data and attacks

Sensitive data is personal data (i.e., *any information* related to an identifiable natural person) which, by its nature, is particularly sensitive in relation to fundamental rights and freedoms [2] (e.g., which might cause forms of discrimination or undesired profiling). In this section, we categorize *what* sensitive data are exposed by spatiotemporal patterns, and *how* (technically) attack models exploit these patterns on published trajectory data. At the end of the section, Table 4.1 connects and summarizes these two directions.

4.2.1 Sensitive data

By taking inspiration from GDPR [2], we distinguish three categories of sensitive data: *identity* (i.e., any data that directly identifies an individual; e.g., fiscal code and social security number), *personal data* (i.e., any information related to an identifiable person; e.g., religion and ethnicity), and *social relationships* (i.e., any relationship between natural people; e.g., friendship or partnership). Despite the value of trajectory data is out of question, its peculiar spatiotemporal, sequential, and recurrent natures threaten the protection of sensitive data.

Identity: As human mobility is highly unique [64], individual trajectories act as fingerprints, since individuals in trajectory datasets are likely to be re-identified using only on a few *known locations*. For instance, a trajectory in a rural area generates outlier locations that are easily exposed [300], and the identity of an individual might be uncovered by linking *shared paths* (i.e., connecting individuals with high trajectory similarity). Additionally to single trajectory locations, individual moving history unveils personal routines and idiosyncratic behaviors that are easily linkable to individual identities. For instance, the *personal gazetteer* identifies recurrent locations in everyday life, such as home, work, and favorite restaurant. Similarly, the *location probability distribution* identifies how likely an individual is in a given location at a given time. Although some spatiotemporal patterns are extracted for the good purposes such as *destination prediction* [287], *point-of-interest (POI) recommendation* [312, 311], and *personalized navigation* [41, 59], acquiring these distinguishable knowledge dramatically enhance attackers' capability of identifying a specific individual.

Personal data: Beside identification, personal gazetteers (e.g., frequented locations, check-ins, POIs) and individual mobility also unveil personal data. The semantic information on locations contained in personal gazetteers (e.g., Mosque or Catholic church, 5^* hotel or B&B) expose individual habits (e.g., religion, wage) to user profiling [97]. Similarly, *mobility preferences* or *recurrent mobility patterns* (e.g., how likely an individual rides a bicycle instead driving of a car, or knowing her preferred routes or frequent stops) vary from person to person [203, 202], exposing even religion [86]. Indeed, by the analysis and prediction of individual trajectories, it is possible to infer demographics, lifestyle, and previously-unknown locations [101, 266]. Interestingly, also from *aggregated location statistics* (e.g., the number of individuals covered by a GSM cell) it is possible to infer the presence of an individual in certain dataset, allowing the inference of her personal data related to the dataset (e.g., her health condition if the dataset is about the movement of hospitalized people).

Social relationships: Social relationships affect user mobility [51]. Following the ever-increasing amount of geo-tagged contents (e.g., check-ins or geo-localized games), individuals not only expose themselves through personal gazetteer, but also give the chance of inferring their social relationships [132]. Additionally, the spreading of positioning systems (e.g., GPS and wireless access points) exposes aggregated patterns such as the *encounter* of people in area of interest (i.e., a continuous time interval in which individuals are close in space; e.g., concerts and manifestations). For instance, as individuals tend to group in communities (e.g., family and colleagues) the encounter and proximity of people in restricted areas unveils social ties based on co-located trajectories [28].

4.2.2 Attack models

An adversary can gather sensitive data on individuals within or across the datasets. The research community is paying much attention to the issue of privacy. While some researchers continuously invent new algorithms to effectively anonymize sensitive data without compromising its analytical power [314], others keep discovering ways to break such anonymity [226]. Among the most famous *de-anonymization* attacks is [207], where more than 80% of the anonymous users published by Netflix for a competition had been re-identified by matching them with those on the publicly available Internet Movie Database (IMDB). De-anonymization techniques can be applied to very different kinds of data. For instance, some works study social network relationships to infer users' identities [208, 256], while others rely on profile data (e.g., username, gender, age) [109, 204].

The sensitivity of trajectory data opened new opportunities to specialize generic attacks to the spatiotemporal domain. We classify existing attack models in two orthogonal categories: *linkage* and *probabilistic*. Linkage models refer to *what* sensitive data is inferred and are categorized depending on such information, while the probabilistic attack model quantifies *how much* knowledge is revealed by accessing the dataset.

Linkage

Depending on the target, linkage attack models are categorized in *record* (i.e., inferring individual identity), *attribute* (i.e., inferring personal data such as heal condition), *table* (i.e., inferring personal data through the presence of a *known* individual in the dataset), and *group* (i.e., inferring social relationships).

Record linkage: An adversary with some background knowledge (e.g., exposed locations [225, 174], origin and destination locations [193], and social relationships [42]) can try to identify the record of a known victim (i.e., perform an identification attack). Record linkage is the principal attack addressed by the state-of-art contributions.

Record linkage is formalized as a k-nearest-neighbor search (i.e., finding the most similar k individuals to the given one) in [148]. While in [230], the authors model a linkage attack as a bipartite graph in which individuals are modeled as two disjoint vertex sets connected by edges weighted by the similarity between the two individuals (e.g., the number co-occurrences at a certain spatiotemporal bin). The maximal match within the bipartite graph [162] identifies the optimal linkage. Linkage attacks differentiate by how individual similarity is computed (i.e., what spatiotemporal patterns are exploited to link two individuals). We distinguish *behavioral similarity* and *point-wise matching*.

Behavioral similarity approaches measure the similarity between two trajectories by extracting a behavioral model from both of them and by computing the similarity between the models. One of the most common models is based on the extraction of stay points [205, 206, 189, 172, 98]. In [285], the authors exploit the uniqueness and regularity of human mobility [254] (e.g., night and daytime mobility behaviors) to recover individual trajectories from aggregated mobility data without any prior knowledge; given a dataset representing the number of trajectories in a cell at a given time, the authors iteratively estimate the probability for a single individual to move from a cell to another in its neighborhood and link adjacent locations by maximizing such probability. However, the issue with behavioral similarity approaches is that they require the adversary's knowledge to be sufficiently rich to build a model that can be compared with the one built from the anonymous source — which is not the typical case. For instance, in the case of social networks, only a few points per user are known to the adversary with respect to the length of the anonymous trajectories. Indeed, these works are all verified by relying on the same dataset to extract both training and test sets of the approach.

Point-wise matching approaches measure the similarity between two trajectories by evaluating the match of the single points of one trajectory against the other; each match is assigned a *score* or a *probability*, which is then used to compute the final similarity [38, 183, 230]. Differently from behavioral similarity approaches, point-wise matching ones are usually applied to data coming from different sources (e.g., anonymous GPS trajectories, and public social network data). For instance, [230] measures the probability of encountering a certain user in a certain location and point in time. Similarly, in [234], the authors use the frequency of user login in different locations to approximate the probability of visiting these locations. In [190], the authors discretize a map in a uniform grid, define the similarity between two individuals as the Jensen-Shannon divergence between their two location probability distributions, and finally link users minimizing the divergence. In [74], the authors link datasets through a spatiotemporal join on co-occurring locations and time periods (known locations or areas can be leveraged to prune the join space). In [148], the authors map trajectories into road network locations, build compressed spatiotemporal signatures by selecting the locations with the highest TF-IDF score, and formalize linkage as k-nearest neighbor problem. While these attacks are based on trajectory micro-data, i.e., raw trajectory locations, aggregated trajectory data (e.g., the count of users in different areas) also pose privacy issues.

Attribute linkage: If sensitive values frequently occur within similar trajectories, an adversary can uncover sensitive information even though cannot unequivocally isolate single trajectories (i.e., she performs an attribute linkage attack but not a linkage attack). Despite

value diversity can be ensured through l-diversity, if distinct sensitive values sharing a semantic similarity occur frequently within trajectories, an adversary can still cause a privacy breach (i.e., perform a similarity attack).

POIs and personal gazetteer easily expose personal data, since they characterize the individual interests [60]. Examples of POIs are home, work, religion or political parties locations [101]. Revealing the POIs can cause a privacy breach as this data may be sensitive (e.g., frequent visits to a hospital suggest potential diseases). In [101], the authors introduce a Markov model that represents the mobility behavior of an individual. POIs are states and transitions correspond to movements from one POI (i.e., state) to another. Then, the authors leverage such model to infer home locations (i.e., where individuals usually spend their night) and regular patterns emerging from circles in the mobility models. In [303], for each individual in a dataset of call records, the authors extract her "top N" locations (i.e., locations with high frequency) and join them with census data. In [160], the authors introduce an algorithm to classify the POI semantic. Given two government diary studies (i.e., logs of two-day individual locations), a multi-class classifier [99] is trained to assign semantic labels based on individual demographics, time of visits, and nearby businesses. Furthermore, by extracting and predicting individual movement patterns (either short-term [287] or even up to a year [238]), it is possible to infer sensitive information such as the mode of transport, demographics and lifestyle [101]. In [313], given a dataset of locations check-ins, the authors use spatiotemporal knowledge and the regularity of human mobility to classify demographics attributes such as gender, age, education, and marital status based on the individual's POIs extracted from check-in dataset. In [86], a Reddit user identified Muslim taxi drivers in New York City by integrating anonymized taxi trips data to the to the daily praying times. By uncovering which taxi drivers were inactive at such times, it is possible to infer sensitive information such as religion. In [97], the authors collected and integrated GPS locations with open data to profile the income, home and working locations of individuals frequenting a specific mall by summarizing frequent location patterns.

Table linkage: The inference of an individual presence in a private dataset set can leak sensitive information. For instance, learning that a victim is part of a dataset of hospital patients implies learning that she suffers some disease [227]. Membership disclosure attacks determine the presence of target individuals within a dataset. In [227], the authors train a classification model to infer whether an individual is part of the aggregated released data. Given a trajectory dataset, the authors extract features for each region of interest (e.g., variance and sum of values of each location over time) to train a classifier. A peculiar case of disclosure—not directly related to individual privacy—is the identification of military bases

from the publication of a visual map representing sport activities using the Strava mobile application [135].

Group linkage: The analysis of trajectory data can leak social relationships between individuals with the published dataset. For instance, individuals in the vicinity of each other on a frequent basis can share home or work place, or share the same religious and political orientation [101]. In [51], the authors investigate the influence of social relationships on human mobility, showing that social relationships can explain about 10% to 30% of all human movement. In [28], the authors exploits the ubiquity of Wi-Fi access points to infer social ties based on co-located trajectories. Individuals tends to group in communities (e.g., family and colleagues) where community members share some traits stronger than with non-members [85]. Relationships can be represented by an undirected weighted graph where vertexes are individuals, edges are relationships, and edge weights quantify the relationship intensity. Communities are represented as sub-graphs. In [28], the authors characterize relationship types: friends, classmates, and others. To construct the ground truth data, each relationship is assigned one (or more) labels based on survey questionnaires. The authors define an *encounter* as a continuous time interval in which individuals are close in space, then extracts spatiotemporal features to train a classifier to label social relationships.

Probabilistic

A probabilistic attack quantifies *how much* information an adversary can gather by accessing the dataset rather than focusing on exactly what records, attributes, and tables the adversary can link to a target victim [100]. Intuitively, the access to a trajectory dataset should not reveal *too much* additional information to what is already known to the adversary. Probabilistic attacks can be understood as a generalization of attribute linkage [83], since their goal is not to infer a specific sensitive attribute, but rather to increase the *generic* knowledge of an adversary. For instance, given some locations known by an adversary, while linkage attacks focus on specific sensitive data, a successful probabilistic attack can reveal the entire trajectory of an individual (as in record linkage) as well as the sensitive attributes related to that trajectory (as in attribute linkage).

Recently, probabilistic attack has been formalized in [119], where given τ known locations, an adversary is limited to learn only additional ε locations. The adversary knowledge can be any continued sequence of spatiotemporal samples, and the maximum additional knowledge that she can learn is called leakage. Similarly, [266] formalizes a probabilistic attack as the probability to learn a location previously unknown, and produces a privacy model to remove all the privacy breaches given some known locations. Intuitively, such

Table 4.1: Categorization of sensitive information, sensitive spatiotemporal patterns, and attack models

Sensitive data	**Attack Model**	**Exploited spatiotemporal pattern**	**Attack models**
Identity	Record linkage	Known locations	[225, 174, 193]
		Location probability distribution	[74, 230, 190, 285, 148]
		POI / Personal gazetteer	[91]
		Shared path	[278]
Personal data	Attribute linkage	Recurrent mobility pattern	[101, 287, 86]
		POI / Personal gazetteer	[101, 303, 160, 313, 60, 97]
	Probabilistic attack	Known locations	[119, 266]
	Table linkage	Aggregate statistics	[227]
Social relationship	Group linkage	Encounter	[28]

probability is related to the uniqueness of unknown locations belonging to the trajectories containing the known locations.

4.3 Protection of trajectory privacy

We categorize the privacy models for the release of anonymized trajectory data as *formal* or *ad-hoc*. Formal privacy models are independent from the data type, and adapt the ones introduced in Section 4.1 to spatiotemporal data (e.g., k-anonymity to produce groups of k trajectories). Ad-hoc models are specific to spatiotemporal data and mobility features (e.g., road networks constraints). We organize the content of the section at an increasing level of details. We first introduce each privacy model and its related works, then we describe the main contributions in detail. Privacy models and countered attacks are summarized in Table 4.2.

Note that many privacy models are built atop *generalization* and *suppression* primitives. Generalization [103] coarsen sensitive locations to reduce their sensitivity. However, generalization achieves anonymization for very coarse trajectories (e.g., at city level) [64]. Suppression deletes sensitive locations to prevent de-anonymization through idiosyncratic trajectories. Additionally, as long trajectories are rarer than shorter ones (e.g., a two-week trace reveals 50% population's top two locations [303]), splitting a trajectory to anonymize its subparts retains higher utility [266].

4.3.1 Formal models

These privacy models define privacy on formal requirements usually expressed as parameters of the anonymization process. Some of the following models address quasi-identifier *QI* attributes, i.e., attributes enabling to breach privacy after the anonymization process. For instance, even after removing direct identifiers (e.g., social security number) from census data, it is possible to recover the identity of 87% of the US population through birth date, zip code, and gender attributes [61, 261]. Given a dataset D, $D(QI)$ denotes the records containing *QI* (e.g., the trajectories containing the locations *QI*).

k-anonymity

Among the anonymity models, k-anonymity counts the highest number of contributions due to its intuitive anonymization approach. A dataset D satisfies k-anonymity if each value in $D(QI)$ appears in at least k records. k-anonymity counters record linkage by ensuring the indistinguishability of an individual within an anonymity group that minimizes information loss (intuitively, how much distortion is required to hide the individual). Since optimal k-anonymity has been proved to be NP-hard [196], k-anonymity models share a two-step *greedy* procedure: building groups of at least k trajectories and anonymizing the trajectories within each group.

(k,δ)-anonymity is the first application of k-anonymity to trajectory data. NWA [4] models trajectories as a cylindrical volumes; if two trajectories move within the same cylinder (i.e., are closer than δ), they are indistinguishable. Since NWA measures trajectory distance with the Euclidean distance, W4M [5] extends NWA to overcome the Euclidean distance limitations (e.g., it is only applicable to trajectories with equal length) by introducing an edit distance between two trajectories. Since their introduction, NWA and W4M inspired further contributions to further reduce information loss, such as applying the minimum description length principle to a distance metric [167], coarsening timestamps to increase the anonymized trajectories [50], and enabling custom k for specific trajectories and time intervals [185, 158, 138].

Rather than a cylindrical volume, GLOVE [118] represents a location as a rectangle in space and time, and iteratively merges the two trajectories at minimum stretch effort; intuitively, the stretch necessary to produce a new generalized rectangle including two locations. If generalization requires a big stretch, locations are suppressed. Similarly, KAM [199] merges locations into density-based clusters, transforms trajectories to sequences of cluster centroids, and prunes all the trajectories whose path is shared by less than k other trajectories. Rather than locations, TOPF [73] groups trajectories with the same start/end

timestamps in anonymity groups of size k, and iterates over the remaining trajectories to add subtrajectories to existing groups with the same start/end timestamps. In [141], after grouping trajectories by start/end timestamps, the authors build a weighted graph for each group where vertexes are trajectories, and trajectories overlapping in time are connected by edges weighted with their Euclidean distance. The authors partition the trajectory graph connected components until no connected component with more than k vertexes exists. SwapLocations [72] clusters trajectories overlapping in time, and, for each cluster, swaps close spatiotemporal locations among trajectories.

While k-anonymity makes *no assumption* on the apriori adversary knowledge, k^m-anonymity [225] ensures indistinguishability against an adversary knowing any sequences of m locations by computing all the subtrajectories with length up-to m, and by iteratively generalizing locations with support $\leq k$. $k^{\tau,\varepsilon}$-anonymity [119] bounds the amount of knowledge an adversary can get, ensuring that any adversary with a knowledge on locations over a time period τ can only discover additional locations over a time period ε. $k^{\tau,\varepsilon}$-anonymity is a variation of k^m-anonymity where m=τ.

Detailed description of the main contributions. NWA [4] and its extension W4M [5], as well as GLOVE [118], are well-known implementations of trajectory k-anonymity, and are often taken as baselines in privacy-model comparisons.

NWA models a trajectory as a cylindrical volume where radius δ represents the location imprecision. Two individuals moving within the same cylinder are indistinguishable, leading to (k,δ)-anonymity. NWA coarsens the timestamps of begin/end trajectory locations within an interval of length τ and groups trajectories by begin/end timestamps. Then, NWA clusters the trajectories in each group by selecting the cluster centers (first the trajectory farther from the dataset center, then the farthest trajectory from the previous center), adding to each cluster the k-1 closest trajectories and assigning the remaining trajectories to the closest cluster within a given radius. Clusters with less than k elements are discarded. Finally, NWA maps each cluster into a (k,δ)-anonymity group by moving the original locations and minimizing distortion (i.e., the Euclidean distance between trajectories). First, NWA obtains cluster centers for $\delta = 0$ (i.e., the mean location), then it moves the points outside a disk with radius δ along the center-point direction. As the Euclidean distance is defined for trajectories with the same number of points and does not capture subtrajectories slightly shifted in time, W4M [5] introduces a similarity function based on the edit distance [271] between two trajectories. Intuitively, an edit operation occurs when two trajectory locations are far in space/time. This time-tolerant distance removes the need to group trajectories into equivalence classes.

GLOVE [118] represents a location as a rectangle in space and time (at the beginning, each rectangle collapse to the location point), and achieves k-anonymity through generalization

and suppression. GLOVE measures the stretch effort (i.e., the smallest loss of accuracy necessary to generalize two locations to an indistinguishable one). Given two locations, the stretch effort measures how much the spatial (and temporal) rectangle should be extended to cover the minimum area including both spatial (or temporal) rectangles. As in hierarchical clustering, GLOVE iteratively merges the two trajectories at the minimum stretch effort by merging each location in the longer trajectory to the location in the shorter trajectory at the minimum sample stretch (and vice-versa). If generalization requires a big spatial stretch, the two locations are suppressed.

l-diversity and t-closeness

If individuals within an anonymity group share similar sensitive values, the adversary infers sensitive information. l-diversity [184] ensures that an anonymity group contains at least l *well-represented* values for each sensitive attribute. Several definitions of well-represented values exist. For instance, a dataset D satisfies *distinct* l-diversity if the number of values for the sensitive attribute in $D(QI)$ is greater than or equal to l. Additional definitions of well-represented values are based on entropy and frequency [184]. When applied atop k-anonymity, l-diversity counters both attribute and record linkage. However, assuming that the distribution of the sensitive attribute "Sex" is 99% female and 1% male, distinct 2-diversity creates at most $|D| \cdot 1\%$ groups (each group contains 1 "male" value) for which an adversary knows that 99% of the individuals in each group are females (skewness attack). Additionally, when sensitive values in a group are distinct but semantically similar, an adversary can learn sensitive information (similarity attack). t-closeness overcomes the limitations of l-diversity in the protection of attribute linkage threats. t-closeness [170] ensures that the distance (e.g., Earth Mover's Distance [236]) between the distribution of sensitive attributes within a group and their global distribution is smaller than t.

Given raw trajectories labeled with semantic data (e.g., home, work, pub, cinema), KLT [270] applies l-diversity and t-closeness to ensure that each location in a trajectory is mapped to heterogeneous semantic classes of POIs. Only KLT adopts l-diversity and t-closeness formulations, but others provide models inspired by l-diversity. $(K,C)_L$-privacy [49] guarantees that any subsequence q of any known L locations is shared by at least K records and that the confidence to infer any sensitive value in S from q is at most C. Any subsequence q (with $0 < |q| \leq L$) such that $D(q)$ does not satisfy KCL conditions is suppressed. Similarly, PPTD [156] suppresses a critical subtrajectory τ if the possibility to link a MO in the private dataset is higher than a given threshold. (α,K,L)-privacy [175] guarantees that any subtrajectory τ is contained in a group of at least k elements, that the probability of inferring a sequence of L sensitive locations from τ is lower than α, and that

the probability to infer a sensitive value v is lower than α. (l,α,β)-privacy [294] ensures distinct l-diversity, α-sensitivity (i.e., the probability to infer sensitive value is below α), and β-similarity (i.e., the probability to infer a value within a sensitive group is below β). The authors identify critical sequences of maximum length m (upper bound to the adversary knowledge) and modify/drop them to enforce l-diversity, α-sensitivity and β-similarity. c-safety [201] protects semantic trajectories based on the generalization of visited places within a POI taxonomy. This is similar to l-diversity, but the number of sensitive places is not fixed.

Detailed description of the main contribution. KLT [270] is the only approach implementing both l-diversity and t-closeness. Given raw trajectories labeled with semantic data (e.g., home, work, pub), the authors leverage l-diversity to ensure that each location in a trajectory is mapped to heterogeneous semantic classes of POIs. The authors partition space into regions containing POIs from different classes. Each region is regarded as a vertex in graph connecting neighboring regions via bidirected edges. Given two locations in regions l_a, l_b the authors apply Dijkstra to find the shortest path between l_a and l_b and create a new connected region l_{c_1}. As to time, each location has a time slot $[t_a, t_a + d_a]$ and $[t_b, t_b + d_b]$ that is generalized to $[t_c, t_c + d_c] = [min\{t_a, t_b\}, max\{t_a + d_a, t_b + d_b\}]$. Then, the authors extend l_{c_1} to l_{c_2} to satisfy l-diversity. If l_{c_1} has at least δl categories of POIs, l_{c_2} corresponds to l_{c_1}. Otherwise, the authors calculate the categories of POIs and the area for each region, then pick the neighboring region with minimum area among the ones that, if merged to l_{c_1}, satisfy l-diversity. Finally, the authors find a region l_c that extends region l_{c_2} to satisfy t-closeness. t-closeness is guaranteed by constraining the Kullback-Leibler divergence between the POI distribution of l_{c_2} and the POI distribution of the whole dataset. If t-closeness is already satisfied, l_c corresponds to l_{c_2}, else l_{c_2} is expanded until the requirement is satisfied.

Differential privacy

Differential privacy [76] ensures that the presence of a record in a dataset leaks a controlled amount of information ε. An algorithm f satisfies ε-privacy if for any two datasets D_1 and D_2 that differs of one record, and all set S of values in the image of the algorithm (i.e., $S \subseteq Range(f)$) it is

$$Pr(f(D_1) \in S) \geq e^{\varepsilon} \cdot Pr(f(D_2) \in S)$$

where Pr denotes the probability to observe a specific output.

Several randomized mechanisms achieve differential privacy for trajectory data. The *Laplace mechanism* adds a controlled amount of noise drawn from the Laplacian distribution

$Lap(\frac{\delta f}{\varepsilon})$ [76] to the function f to compute. Tuning ε depends on the *sensitivity* δf, i.e., the largest effect on f's output that the presence of a single individual has. For instance, if f returns the number of rows in the dataset, $f : D \rightarrow R = |D|$, it is $\delta f = 1$. The Laplace mechanism requires the image of f to be real numbers, and is mainly applied to anonymize aggregate queries (e.g., count queries). Several algorithms adapted it to generate *synthetic* trajectories (i.e., data not preserving truthfulness at record level) from the original data distribution. To this end, trajectories are commonly represented in a prefix tree to which Laplacian noise is added (e.g., [133]), and/or statistics on the original trajectory dataset are extracted to generate synthetic trajectories representing such statistics (e.g., [125]).

DPT [133] adapts the Laplacian mechanism to publish *synthetic* trajectories. The authors discretize space in hierarchical grids and build a prefix tree for each grid. The tree nodes represent grid cells, and count the trajectories moving through these cells. To achieve differential privacy, the authors add noise from the Laplacian distribution [76] to trajectory counts, and finally join paths with increasing length to synthesize anonymized trajectories. While DPT [133] synthesizes spatial trajectories, SafePath [12] synthesizes spatiotemporal trajectories by adding the timestamp location to the prefix tree, and [308] also synthesizes trajectories with sensitive attributes. DP-STAR [125] synthesizes trajectories by injecting noise to the utility features introduced in DPT [133]: density grid, mobility model, trip distribution, route length). As for the density grid, note that the probability to transition from a location to another depends on the grid structure (e.g., a fine-grained grid has many cells—some with very few visits—and requires more noise addition). DP-STAR samples trajectory's start/end points and length, and estimates intermediate locations using a random walk on the mobility model and the probability to reach the final location from the current one. Similarly, DP-WHERE [198] adopts different statistics to synthesize trajectories from a dataset of geo-localized phone calls (e.g., calls per day, call time, and call location).

The Laplacian mechanism requires a differentially-private f that maps datasets into real numbers. However, f can map datasets to data structures (e.g., clustering partitions). The exponential mechanism [194] samples the synthesized output from a probability distribution. Intuitively, the exponential mechanism generates artificial samples that preserve the same distribution of the original dataset (i.e., records which represent the input data). Each *possible* record gets a score based on its likelihood to be derived from the input data, and records with high scores are drawn. However, the sampling complexity grows exponentially in the probability space. For instance, in [139], the authors formalize anonymity groups as the ones with the highest *utility* (i.e., intra-group similarity) among the groups in all the possible partitions, but, being the number of partitions exponential, the authors provide a sub-optimal solution which leverages a single partitioning instance. Similarly, in [169],

the authors assign a utility to k-means clustering in terms of intra-cluster distance and sample a clustering partition from an exponential distribution. In [147], generate synthetic trajectories by incrementally sampling the next trajectory location distance and direction from exponential distributions.

Finally, differential privacy can be achieved via randomized response, i.e., deciding by chance whether to return the *actual* outcome or a randomized one. In [249], the authors sample trajectory locations and interpolate the missing ones. Since locations adjacent to sensitive ones may leak sensitive information, Lclean [131] determines the correlation between sensitive and adjacent points. For each sensitive region, Lclean finds sequences close in space/time that either do not contain sensitive information or that show strong correlations. Given the sequences, Lclean substitutes trajectory subsequences via randomized response, making it impossible for an adversary to predict sensitive regions.

Detailed description of the main contribution. DPT [133] is a well-known model for differential trajectory privacy often referred to as a baseline for comparison. The authors leverage the n-gram model introduced in [47] to represent the correlation between trajectory subsequences, and discretize space using a hierarchical reference system, i.e., a set of uniform grids with increasing resolutions. Different reference systems capture movements (i.e., consecutive locations) at different speeds. Within each reference system, individuals are constrained to move from a cell to a neighboring cell. If a movement is too fast to be captured within the same reference system, individuals jump to a coarser one. To each reference system corresponds a prefix tree of depth n, whose nodes count the trajectories moving from the root cell to the current one. The authors add Laplacian noise [76] to the node *true* counts, and decide whether to prune a node by comparing its *noisy* count with a threshold Θ. The exploration ends when either the depth of the tree reaches n or no node can be further expanded (i.e., the noisy counts do not pass Θ or the privacy budget is exhausted). The authors leverage an adaptive allocation of the privacy budget ε: for a shorter path, each node in the path should receive more privacy budget. The authors estimate the length of a path based on known noisy counts and then distribute the remaining privacy budget. Then, the authors drop the trees with high noise and low utility. As trajectory directionality could be lost due to noise addition, the authors weight the sampling of synthetic trajectories with the likelihood for an individual to move from a cell to another. Based on the 1-st order Markov assumption, given the noisy prefix tree, synthetic trajectories are built by joining n-grams sharing $n-1$ elements.

Plausible deniability

Plausible deniability [30] generates synthetic data with differential privacy guarantees. For any dataset D and any output $y = M(d_1)$ with $d_1 \in D$, y is releasable with (k, γ)-deniability if there exist at least k-1 distinct records such that for any $i, j \in \{1, ..., k\}$

$$\gamma^{-1} \leq \frac{Pr(y = M(d_i))}{Pr(y = M(d_j))} \leq \gamma$$

where Pr denotes the probability to observe a specific output. In other words, an adversary cannot tell if a synthetic record has been generated by an individual in the original dataset.

Detailed description of the main contribution. The only implementation of plausible deniability for trajectory data is [29]. SPLT [29] generates synthetic trajectories with high *semantic similarity* and low *geographic similarity* to the original ones. Semantic similarity sim_S reflects if pairs of locations have the same semantic, while geographic similarity sim_G correlates movements through locations. Given two individuals, while sim_G is usually low (i.e., the two do not frequent the *same* place), sim_S can be high (i.e., the frequented places are semantically similar, for instance "home" and "work").

Firstly, the authors sample real traces (i.e., seed dataset). For each seed, the authors compute a 1st-order Markov model representing the probability to visit and to transition between locations. Individual mobility models are averaged into an aggregate one. Secondly, the authors build a location semantic graph in which vertexes are locations, and edges are weighted with the semantic similarity between locations. Such graph is clustered in classes, such that locations in the same class are visited in same way regardless their geographic positions. Thirdly, the authors replace the geographic locations in the mobility with all the locations from the same semantic class. Any random walk is a valid trace that is semantically similar to the seed. To enforce geographical consistency with the general mobility model, the authors encode traces into an HMM (symbols are locations, observations are the semantic classes, transition probability matrix follows the aggregate mobility model) to find the sequence of locations that most likely generated the semantic one by using the Viterbi algorithm. Finally, the authors run a privacy test to decide whether to release each trace under *statistical dissimilarity* and *plausible deniability*. Statistical dissimilarity ensures independence by measuring the EDR distance between the synthetic/seed probability distributions, and by measuring the distance between the two traces. For all seeds s and respective synthetic trajectories f, the trace is rejected if either $sim_G(f, s) \leq \delta$ or $|f \cup s| \leq \delta_i$. Plausible deniability implies that a synthetic trace is generated by multiple seeds. Given the dataset of trajectories discarded as seeds, the synthetic trace satisfies (k, δ)-

plausible deniability if there are at least $k \geq 1$ alternative traces $a \in A$ such that: $|sim_S(s,f) - sim_S(a,f)| \geq \delta$.

4.3.2 Ad-hoc models

Ad-hoc models address privacy-preserving publication *specific* to trajectory data, such as mix zones (i.e., geographical areas in which individuals swap pseudonyms) and dummy (i.e., synthetic trajectories resembling the original).

Mix zone

Mix-zone models prevent long-term tracking of trajectories [26]. Once entered a mix zone, individuals swap their pseudonyms and exit the mix zone after an unknown length of time. An adversary observes the pseudonyms of all ingress/egress events in order to reconstruct mappings between pseudonyms (i.e., record linkage). Given n individuals, there is a total of $n!$ mappings. Three conditions bound mix-zone areas: *location accuracy* (i.e., the lower the accuracy, the bigger the area), *sampling accuracy* (i.e., the higher the rate, the more accurate the linking), and *computational cost* (i.e., the more mix-zones, the higher the cost).

MobiMix [218] models a mix zone as a k-anonymity region where k individuals enter in some order, swap pseudonyms, and none leaves before k individuals have entered. The placement, geometry, and time spent inside mix zones affect the privacy level. If permanence time is constant, it easy to perform a first-in first-out attack. Randomness ensures reordering, however, individuals are not always able to spend random time inside a road network, and do not follow uniform transition probabilities when entering/exiting the mix zone (e.g., in case of trafficked routes [26]). MobiMix introduces the *time window bounded non-rectangular* mix-zone model: for each branch of a road junction, a mix-zone region starts from the center of the junction and expands to the outgoing road segment. The length of the region is proportional to the average road-segment speed. In [193], the authors analyzed the effectiveness of mix-zones under the assumption that MOs follow the shortest path between origin/destination locations (e.g., to reduce the cost of taxi trips). An adversary can compare the minimum path between known origin/destination locations using the Dijkstra algorithm in a road network and the minimum DTW distance between the anonymized trajectories. Given eight mix zones positioned in road intersections with the highest traffic flows, the authors linked more than 90% trajectories back to the original path.

Detailed description of the main contribution. The protection/utility achieved by mix zones depends on their placement. NUTMP [176] formalizes placement by minimizing the number pairwise-associated vertexes in a road-network graph. If a individual can travel

from a vertex to another without going through a mix zone, the two vertexes are pairwise associated. The total number of pairwise associations represents the privacy level. The authors combine this objective function and constraints (e.g., number of mix zones and traffic density) to derive an NP-hard integer-linear-programming formulation and an heuristic solution. The road network is partitioned into disconnected components by looking for articulation points, i.e., vertexes whose removal disconnect the graph. If all vertexes that are not in an independent set are selected as mix zones (i.e., in graph theory, an independent set refers to a set of vertices that are not adjacent to each other), there will be no pairwise association between the vertexes in the independent set. The authors iteratively remove the vertexes introducing the least number of pairwise associations from the mix zone candidate set until all constraints are satisfied. To consider the impact of traffic conditions, NUTMP only considers the articulation points with high uncertainty in determining if a user has traveled through such vertex, and vertexes ensuring a minimum pairwise entropy (intuitively, high uncertainty in connecting two pseudonyms).

Dummy

Dummy anonymization release original trajectories along with fake ones. The probability to disclose true trajectories is minimized by dummy-privacy models.

In [297], the authors generate dummy trajectories resembling individuals moving in free space given three privacy parameters: *short-term disclosure* (i.e., the probability of successfully identifying a true individual location); *long-term disclosure* (i.e., the probability to identify a trajectory depending on its intersection with others); and *distance deviation* (i.e., the distance between dummy and real trajectories for a given individual). The authors introduce the *random pattern* strategy, which selects dummy start/end points and intermediate movements as random moves towards the end point. However, this strategy requires the synthetization of many dummy trajectories to meet privacy requirements. Several contributions aim at reducing the number of generated dummies by applying different strategies. The *rotation* strategy, given an individual trajectory, generates a new dummy trajectory by rotating the original one. In [165], the authors introduce the *K-intersected* strategy, where, given K intersection points as input, a dummy trajectory is generated by composing two sub-dummy trajectory sets: one between two intersection points (a sub-dummy is obtained by performing random moves from the start to the end point), and one of the sub-dummies that do not contain intersection points (as for the rotation strategy, the authors determine the solution space based on distance derivation first, and then pick the locally optimal sub-dummy). The authors also introduce the *multiple rotation* strategy. Individual trajectories usually overlap with a given frequency (i.e., probability). A point within the overlapping area with the high-

Table 4.2: Privacy models and countered attacks

Attack Model	**Privacy Models**	**Papers**
Record linkage	k-anonymity	W4M [5], GLOVE [118]
	Differential privacy	DPT [133]
	Plausible deniability	SPLT [29]
	Dummy	DTPP [174]
	Mix-zone	NUTMP [176]
Attribute linkage	l-diversity, t-closeness	KLT [270]
Table linkage	Plausible deniability	SPLT [29]
	Differential privacy	DPT [133]
Probabilistic	Differential privacy	DPT [133]

est probability is used as a rotation point, and new dummies are devised using the rotation strategy. In [282], the authors introduce the *adaptive generation* strategy. For each given rotation angle and location in directory T, the authors synthesize a new candidate dummy trajectory satisfying the distance distortion, then perturb trajectory locations to achieve more uniformly distributed trajectories by moving these locations in sparse areas. In [151], the authors generate dummies resembling known individual movements between known stop locations. So far, these contributions do not consider external knowledge.

Detailed description of the main contribution. The effectiveness of dummy-privacy models relates to the capability to rule out fake trajectories. This highly depends on the apriori adversary knowledge: if a true spatiotemporal location is known (i.e., exposed), an adversary can easily rule out all the trajectories not containing such location. DTPP [174] generates dummy trajectories based on exposed locations. The authors divide the location space into a uniform grid. If a trajectory location is exposed, the authors get dummy candidates from the neighboring cells. Otherwise, if the region where the location belongs cannot generate enough dummy locations to anonymize it the location is suppressed. DTPP *generates* k-1 dummy trajectories to form an anonymous trajectory set including the real trajectory (whereas in k-anonymity no synthetic trajectory is generated). The authors construct dummy trajectories by connecting dummy locations. Since a trajectory generates m dummy locations at each sensitive location, the number of dummy trajectories is exponential. DTPP leverages spatiotemporal reachability to drop such a number.

4.4 Conclusion and research directions

In this chapter, we provided an overview of the sensitivity and privacy issues of trajectory data. Since trajectory locations follow idiosyncratic spatiotemporal patterns, only a few locations can link 95% of individuals; the more detailed/longer the trajectory, the easier to break individuals' privacy. As a consequence, many anonymization models have been introduced to protect privacy issues such as personal gazetteers and information as well as the linkability of trajectories from different datasets. Since there are no common frameworks and datasets to compare the existing privacy models, given some privacy requirements, choosing the appropriate privacy model is non-trivial and requires an extended empirical evaluation. This is especially true when quantitative distortion/utility criteria have to be met.

In this context, further research directions are the following.

- Identifying common privacy/utility metrics that enable the comparison of existing models. Utility metrics measure the usefulness of data with respect to a specific goal; in other words, they assess that data are not *too much* distorted. Privacy metrics measure *how much* sensitive information can be disclosed/protected.

- Studying the scalability of the proposed methods. Indeed, the usefulness of a privacy model is related to its capability to scale up to big datasets. Such metrics are orthogonal to privacy/utility metrics, but ensure the feasibility of anonymization models [226]. A typical performance metric is the execution time.

- Performing the empirical comparison of the existing privacy approaches in a unifying framework involving shared *public* datasets. Indeed, while many attacks and anonymization models have been introduced in this chapter, they are usually evaluated against private datasets and custom metrics.

- Identifying new attacks that expose the limits of the existing privacy (anonymization) models since, with the increasing availability of big data, new privacy issues arise by integrating data from different sources (e.g., mobility and social data; see Chapter 5).

Chapter 5

De-Anonymization of Personal Gazetteers

5.1 Introduction

The interest towards trajectory data has sensibly increased since the dissemination of mobile devices; indeed, the latter allows to constantly trace the location of the people by registering their connections to cell towers, GPS satellites, and WLAN devices. There is enormous value behind trajectory data, as they enable a wide range of *location based services* (LBS), ranging from navigation applications to crowdsensing applications and target marketing [226]. Their collection and ownership are not limited to infrastructure service providers, but it extends to any application which has been granted access to the geolocalization services of a device.

When trajectory data is ascribed to people, *personal gazetteers* [315] are among the most interesting pieces of information that can be extracted. The personal gazetteer of a user is intended as the set of places mostly visited by such user (e.g., the home and work places, restaurants, shops), possibly enriched with the set of time instants of each visit; we refer to each visited place as a *stay point*. Several research efforts have proven that the discovery of users' personal gazetteers is possible by applying a clustering algorithm on users' trajectories [315]. Due to their sensitiveness, personal gazetteers are usually anonymized; nonetheless, researchers have shown that they present unique patterns (much like fingerprints [148]) that expose them to the risk of being de-anonymized, i.e., of being associated with the individual's true identity. The literature shows that the knowledge of few spatiotemporal points of a user are enough to uniquely identify his/her anonymous profile [64, 235].

A de-anonymization attack is carried out by an *adversary* to uncover the true identities in an anonymous dataset by exploiting some *external knowledge* where the users' identities are

known. To this end, social data is often used as external knowledge due to its relatively easy accessibility [38, 230, 278]. Users of social networks often overlook their privacy settings in the platform, leaving their identities and their posts (including the time of posting and the possibly associated georeferenced information, e.g., the location a tweet was posted from) accessible by the general public. A sequence of georeferenced posts on social media still represents trajectory data, namely a *social trajectory*. Although the frequency of points in social trajectories is not very high (and depends on the users' level of interaction with the social network), they can be leveraged by the adversary for the de-anonymization task by measuring its overlapping with the mobility ones.

In this chapter, we present DART (*De-Anonymization of peRsonal gazeTteers*), an innovative approach for the de-anonymization of personal gazetteers through social trajectories. The goal is to exploit the uniqueness properties of personal gazetteers to measure their similarities with social trajectories. State-of-the-art proposals for de-anonymization of trajectory data follow two main philosophies. The first one is to extract a more-or-less complex behavioral model from both sets of trajectories and to compare them. This approach is hardly applicable to social trajectories, as the sporadic nature of the contained points hinders the extraction of a behavioral model. The second one is to carry out a point-wise comparison, i.e., to calculate a *matching score* between the single points of two given trajectories and to measure their similarity by aggregating these scores. The issue with this approach is that it fails to capture the unique patterns of trajectories, risking random/irrelevant matches to impact significantly on the similarity score. In this context, the contribution of this chapter can be summarized as follows.

- We propose DART as an innovative approach that combines the two main philosophies proposed in existing works to properly address the scenario in which social trajectories are used as the external source to de-anonymize personal gazetteers. In particular, it carries out a point-wise comparison to identify the portion of a behavioral model (i.e., the personal gazetteer itself) that is matched by the social trajectory. With respect to the literature discussed in Section 4.2.2, this contribution is categorized as point-wise record linkage attack.

- The point-wise matching strategy relies on the computations of a spatiotemporal model from the personal gazetteer, allowing to estimate the location of the user at a certain moment in time. This makes the approach robust to (i) potentially missing data in the mobility trajectories, and (ii) potentially misaligned datasets, i.e., when the mobility and social trajectories are collected over different periods.

- We rely on a big data implementation that guarantees the scalability of the approach to large volumes of data. Indeed, the literature often emphasizes the importance of designing scalable and efficient algorithms in order to enable their applicability on large-scale datasets [104, 195].
- We measure the effectiveness of the approach on two real-world datasets (a proprietary one and the Mobile Data Challenge dataset [163]) and we compare with state-of-the-art algorithms [38, 148, 278].

Following the extensive literature survey from Chapter 4, this chapter is structured as follows. Section 5.2 introduces personal gazetteers and explains how they are computed, their uniqueness properties and the spatiotemporal model that can be derived; Section 5.3 presents the details of the de-anonymization approach; the experimental evaluation is discussed in Section 5.4; finally, the conclusions are drawn in Section 5.5.

5.2 Stay points as personal signatures

The *personal gazetteer* of a user is the set of places that are meaningful to her (e.g., the home, the workplace, a shop frequently visited). These places are also commonly referred to as *stay points* and are typically retrieved by applying some clustering functions on the user's trajectory. We use S to refer to the personal gazetteer obtained from a trajectory T (Definition 1), i.e., $S = \{s_1, ..., s_m\}$, where s_i is a stay point obtained from T. Each stay point s_i is a triple $(latitude, longitude, Timestamps)$ where $latitude$ and $longitude$ identify the spatial location of the stay point (typically the centroid of the cluster), while $Timestamps$ is the set of timestamps for which there exists a point in T that occurs at s_i.

In this Section:

- we discuss how stay points are retrieved (Section 5.2.1);
- we introduce a measure to evaluate the uniqueness of a set of stay points (Section 5.2.2), which will be used by DART to determine the effectiveness of a match between a social trajectory and a personal gazetteer;
- we present our spatiotemporal model based on the user's personal gazetteer (Section 5.2.3), which will be used by DART to estimate the user's location in the absence of punctual data.

5.2.1 Computing stay points

There exist several methods for computing stay points out of trajectory data. Except for some attempts based on exploratory approaches [192] (where the loss of the GPS signal is interpreted as a clue of the user's stopping in a stay point) or *k*-means clustering [18], most proposals consist of variations of density-based clustering algorithms like DBSCAN [81]. Among the most notable ones, we mention DJ-Cluster [315], a density-joinable clustering technique that improves DBSCAN by addressing some performance issues by applying some pre-processing rules to the available data. For instance, it focuses only on the points where the user is known to not be moving by discarding every point associated with a speed higher than a certain threshold. Otherwise, the risk would be to include in the identified clusters her most common walking/driving paths.

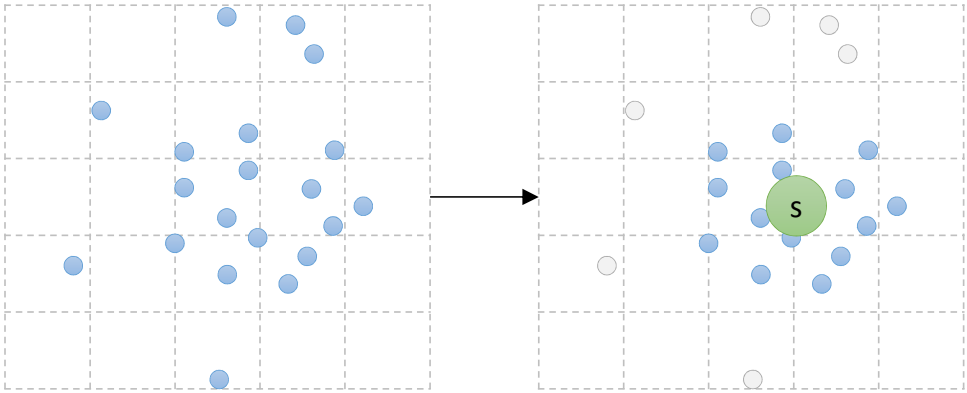


Figure 5.1: Intuition of the algorithm for stay point extraction.

In this chapter, we rely on Patchwork [117], i.e., a density-based algorithm that improves DBSCAN by providing a parallel big data implementation that guarantees linear scalability. Patchwork discretizes the spatial domain into a uniform grid made of 50x50 m cells; the cells with less than 20 points are discarded, the others are aggregated into clusters. Figure 5.1 provides the intuition of the algorithm. Noticeably, given a cluster of points, the stay point is chosen as the centroid of the cluster. As the original algorithm is meant for a single trajectory, our implementation is further parallelized to analyze every trajectory in the dataset as a *bag-of-task*; the clustering algorithm is applied in parallel by mapping each trajectory to its extracted stay points. Additionally, we extend Patchwork to include the pre-processing rules introduced by DJ-Cluster. More specifically:

- trajectories with less than 100 points are discarded;
- points with accuracy greater than 30 m are discarded;

- points with an associated speed higher than 2 km/h are discarded;
- each trajectory is sampled down to 1 point per minute to avoid the generation of clusters on random stopping points (e.g., due to traffic lights).

5.2.2 Uniqueness of stay points

Stay points alone are often enough to uniquely identify a user among the others [116, 303, 64]. Consider $\mathscr{S} = \{S_1, \ldots, S_n\}$ the set of available personal gazetteers and $W = \{s \mid s \in S, S \in \mathscr{S}\}$ the universe of all stay points. Then, let us define the concept of adjacency between stay points as follows.

Definition 5 (Stay points adjacency) *A stay point $s_1 \in W$ is* adjacent *to a stay point $s_2 \in W$ (i.e., $s_1 \cong_{adj} s_2, s_1 \neq s_2$) if the spatial distance between the two stay points is lower than a spatial threshold ω, i.e., $dist_{space}(s_1, s_2) \leq \omega$.*

Clearly, we are interested in evaluating the adjacency between stay points belonging to different personal gazetteers. In particular, the adjacency definition is necessary to introduce a concept of uniqueness: we say that $s_1 \in W$ is *unique* if there is no adjacent stay point $s_2 \in W$ that belongs to a different personal gazetteer, i.e., $s_1 \in S', s_2 \in S'', S' \neq S'$.

Consider the example in Figure 5.2, where the stay points of three personal gazetteers are plotted; it is $S' = \{s_1, s_2, s_3\}$, $S'' = \{s_4, s_5\}$, $S''' = \{s_6, s_7\}$. The stay points of S' are not individually unique, because each of them is adjacent to stay points of other personal gazetteers. However, their combination represents a unique signature. More specifically, the subsets $\{s_1, s_3\}$ and $\{s_2, s_3\}$ are also unique, because S' is the only personal gazetteer that can be traced back to both places in both cases.

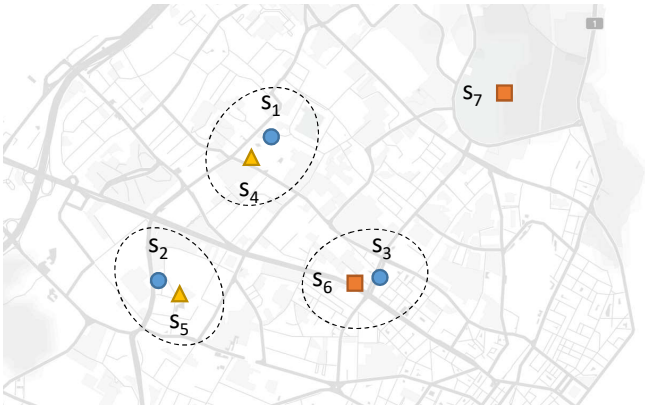


Figure 5.2: Example of stay points of three personal gazetteers

Definition 6 (Set of stay point uniqueness) *Consider a set of stay points $S^* \subset W$; we define the* uniqueness *of S^* in $\mathscr{S}$ as an inverse function of the number of personal gazetteers whose stay points are adjacent to S^*.*

$$unq(S^*) = \frac{1}{|\{S \in \mathscr{S} \mid \forall s_1 \in S^* \; \exists s_2 \in S, s_1 \cong_{adj} s_2\}|}$$

This concept of uniqueness is used in DART to effectively verify the de-anonymization of a user: the best-case scenario is when a social trajectory matches a unique subset of a personal gazetteer. To this end, the adjacency threshold ω plays an important role: on the one side, the higher the value, the higher is the importance given to stay points in remote locations; on the other side, a low-value risks to consider all stay points as unique. Based on an empirical evaluation of our datasets, we find that a good compromise is to set ω to the average diameter of the clusters from which we obtain the stay points in our datasets (the process is explained in Section 5.2.1), which is 200 meters in our case.

5.2.3 Stay point spatiotemporal model

We exploit stay points to define a spatiotemporal model that enables the estimation of a user's location at a certain moment. This is necessary to "fill-in the blanks", as it is not uncommon for anonymous trajectories to be incomplete (e.g., the GPS trace may be recorded only under certain conditions) or for social trajectories to be misaligned with anonymous ones (e.g., because the recording period is different). The granularity we consider is the one of the *hour of the day*, i.e., we split time into 24 time bins[1]. Let S be the personal gazetteer of a certain user, $s \in S$ a stay point of the personal gazetteer, and $b \in [0,23]$ a time bin. Then, we define $stp(s,b)$ as the empirical probability of the user being in stay point s in time bin b.

Let $timeBin : \mathbb{N} \to [0,23]$ be the function that returns the time bin corresponding to a certain timestamp. Also, let $days(s,b) \in [0,\mathbb{N}]$ be number of distinct days along the monitoring period for which there exists a timestamp ts in s such that $timeBin(ts) = b$. Then, the empirical *spatiotemporal probability* of the anonymous user being in s during time bin b is:

$$stp(s,b) = \frac{days(s,b)}{\sum_{s_n \in S} days(s_n,b)}$$

Intuitively, $stp(s,b) \in [0,1]$ indicates the percentage of days for which the user has been recorded being in s and not in other stay points. For instance, in the typical case, $stp(s,b) \to 1$

[1]Time bin 0 refers to the time slot between 00:00 and 00:59; time bin 23 refers to the time slot between 23:00 and 23:59.

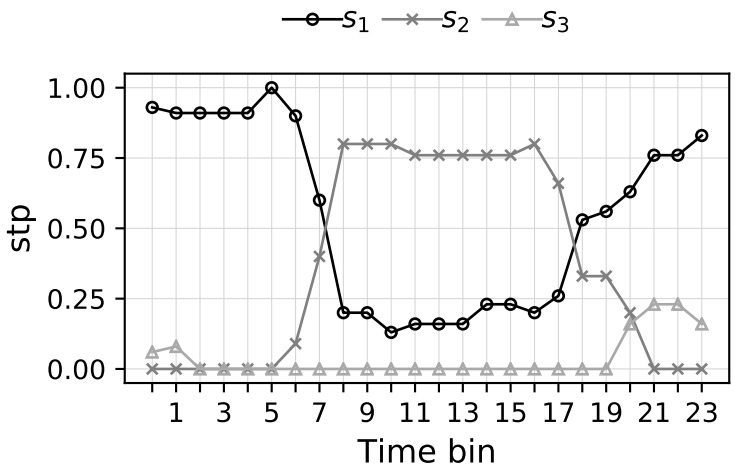


Figure 5.3: Example values of stp on three stay points s_1 (home), s_2 (work), and s_3 (pub/restaurant) in the different time bins.

for the stay point representing the home place during night hours (i.e., $b \in [1,5]$). Figure 5.3 shows some examples for the values of stp on three stay points s_1 (resembling a home place), s_2 (resembling a work location), and s_3 (resembling a pub or restaurant) in the different time bins.

5.3 The de-anonymization approach

The de-anonymization approach relies on anonymous personal gazetteers, each of which represents a *signature* of a user's most frequented places. Indeed, a personal gazetteer (or even a subset of it) is often unique within a population of users [116, 303, 98]. Let us define with $\mathscr{S}$ the set of anonymous personal gazetteers and with $\mathscr{K}$ the set of social trajectories (e.g., geo-localizated tweet histories)[2]; the goal of the approach is to associate each social trajectory $K \in \mathscr{K}$ to the most probable personal gazetteer $S \in \mathscr{S}$. To this end, we proceed in two steps.

1. The first step consists in calculating a *global score* between each social trajectory K and personal gazetteer S. The global scoring function relies on *local scores* that are calculated by measuring the spatiotemporal distance between the single points $k \in K$ and the single stay points $s \in S$. The details of this step are discussed in Section 5.3.1 and Section 5.3.2.

[2]The letter "K" refers to the social trajectories being *known* instead of anonymous.

2. The second step consists in assigning each K to the most probable S based on the computed global scores. Its details are discussed in Section 5.3.3.

Figure 5.4 shows an intuition of the two steps. In the scoring step (i.e., the left figure), each point k is associated with the spatiotemporally closest stay point s; the edges represent these associations. Then, the global score is based on the maximum scores for each stay point (i.e., the bold edges); indeed, the more stay points in S are matched by the points in K, the more significant is the correspondence between K and S. In the de-anonymization step (i.e., the right figure), the rationale is similar: among all the possible matches between personal gazetteers and social trajectories (i.e., the edges), we find the best matches (i.e., the bold edges) such that (a) each K is matched to up to one S, and (b) each S is matched to up to one K. This is necessary to avoid having the same social trajectory being assigned to multiple personal gazetteers, and vice versa.

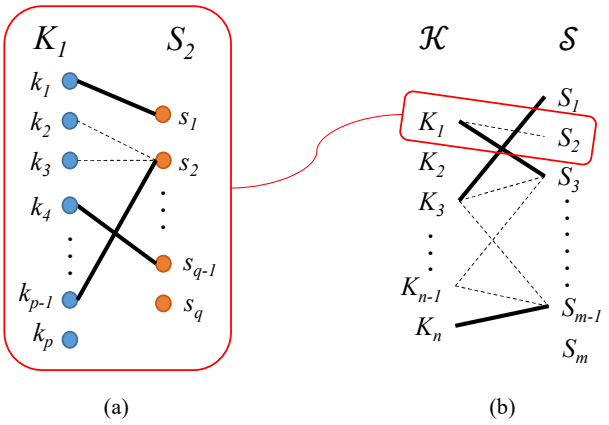


Figure 5.4: On the left (a) the scoring step between a social trajectory K_1 and a personal gazetteer S_2; edges represent local scores, and the bold ones are those used to compute the global score. On the right (b) the de-anonymization step to assign to each social trajectory $K \in \mathscr{K}$ its most probable personal gazetteer $S \in \mathscr{S}$; here, edges represent global scores, and the bold ones represent the chosen assignments.

5.3.1 The local scoring function

In order to evaluate the match between a social trajectory and a personal gazetteer, we must first define a local scoring function to evaluate the single matches between the points $k \in K$ and the stay points $s \in S$. Given a couple (k,s), the local score builds on two principles: (1) the *closeness* between k and s from both spatial and temporal perspective; (2) the *uniqueness* of k with respect to S, which depends on whether k is close to other stay points belonging to different personal gazetteers (i.e., other than S).

Let us start from the spatiotemporal closeness between a point k and a stay point s. Measuring the spatial correspondence is trivial, as both are associated with a spatial location in terms of latitude and longitude. Conversely, to measure the temporal correspondence we rely on the set of known timestamps for s, which indicate the moments in time when the anonymous user has been recorded in s. However, due to the potential incompleteness of trajectory data, it may be hard to find a punctual correspondence. In the absence of the latter, we rely on the spatiotemporal model of the personal gazetteer (Section 5.2.3) to measure the probability of the anonymous user being in s around the time of k. Thus, we define the closeness measure as a function of three components, namely Δ_{space}, Δ_{time}, and Δ_{habit}. Consider $I = \{k \mid k \in K, K \in \mathscr{K}\}$ the universe of all known points and $W = \{s \mid s \in S, S \in \mathscr{S}\}$ the universe of all stay points.

- Let $\Delta_{space} : (I, W) \rightarrow [0,1]$ be the function defined by

$$\Delta_{space}(k,s) = g(dist_{space}(k,s)/200)$$

  to calculate the *spatial closeness* between k and s. Here, $dist_{space}(k,s)$ is the spatial distance in meters between k and s, and g is a Gaussian distribution with mean $\mu = 0$ and standard deviation $\delta = 0.4$. This ensures that $\Delta_{space}(k,s) = 1$ when $dist_{space}(k,s) = 0$ and that $\Delta_{space}(k,s) \approx 0.5$ when $dist_{space}(k,s) = 100$ meters. Note that the normalizing factor of 200 meters is consistent with the threshold for stay point adjacency ω set in Section 5.2.2.

- Let $\Delta_{time} : (I, W) \rightarrow [0,1]$ be the function defined by

$$\Delta_{time}(k,s) = g(dist_{time}(k,ts)/3600)$$

  to calculate the *temporal punctual closeness* between k and s. Here, ts is the timestamp in s that is closest to the timestamp in k, $dist_{time}(k,ts)$ is the temporal distance in seconds between k and ts, and g is defined as above. This ensures that $\Delta_{time}(k,s) = 1$ when $dist_{time}(k,ts) = 0$ and that $\Delta_{time}(k,s) \approx 0.5$ when $dist_{time}(k,ts) = 30$ minutes. The normalizing factor of 1 hour (i.e., 3600 seconds) is consistent with the granularity of time bins in our spatiotemporal model (Section 5.2.3) and with the granularity considered by several related approaches (e.g., [148, 278]).

- Let $\Delta_{habit} : (I, W) \rightarrow [0,1]$ be the function defined by

$$\Delta_{habit}(k,s) = \sum_{(x,y) \in \gamma} stp(s, b+x) * y$$

to calculate the *temporal behavioral closeness* between k and s. In practice, Δ_{habit} is the weighted sum of the empirical probabilities of the user being in s at different time bins: b is the time bin of k, and $\gamma = \{(-2,0.1),(-1,0.2),(0,0.4),(+1,0.2),(+2,0.1)\}$ is a set of (x,y) couples where x is a time bin index relative to b, and y is a weight to be applied to the stp function[3]. This means that $stp(s,b)$ is weighted 0.4, $stp(s,b+1)$ is weighted 0.2, and so on. The values of γ have been obtained from empirical evaluations, but they can be tuned as long as $\sum_{(x,y)\in\gamma} y = 1$.

The intuition is to accommodate behavioral evolution by considering not only the empirical probability at the exact time bin b (i.e., $stp(s,b)$) but also the probabilities in the closest time bins. For instance, with reference to Figure 5.3, consider a point k made in time bin $b = 5$; here, $\Delta_{habit}(k,sp_2) = stp(sp_2,3)*0.1 + stp(sp_2,4)*0.2 + stp(sp_2,5)*0.4 + stp(sp_2,6)*0.2 + stp(sp_2,7)*0.1 = 0+0+0+0.02+0.04 = 0.06$. Thus, instead of completely refusing the possibility of the user being in sp_2 at the time bin of k (as $stp(sp_2,5) = 0$), we admit a small probability (as the empirical evidence shows the user being in sp_2 in the immediately following time bins).

Given the aforementioned components, we define the spatiotemporal closeness as follows.

Definition 7 (Closeness) *The closeness between a point k and a stay point s is:*

$$closeness(k,s) = \Delta_{space}(k,s) \cdot (\Delta_{time}(k,s) + (1 - \Delta_{time}(k,s)) \cdot \Delta_{habit}(k,s))$$

Definition 7 considers Δ_{space} as the spatial component and the rest as the temporal one. The temporal component gives priority to the punctual closeness, i.e., Δ_{habit} is considered only in the absence of a punctual correspondence. Indeed, timely correspondences convey more precise information than those based on the users' habits.

Example 2 *The mechanism of the temporal component is further explained by the following scenarios.*

- $\Delta_{time} = 1$*: there is a punctual correspondence of the anonymous user being in s at the time of k; thus, independently of the user's habits* Δ_{habit}*, the temporal component measures* 1.
- $\Delta_{time} = 0$*: there is no punctual correspondence of the anonymous user being in s at the time of k; thus, the temporal component depends on whether there is any probability of the user's being in s during k's time bin based on its habits* Δ_{habit}.

[3]Notice that we assume circularity in the time bins: e.g., $23+2 = 1$, and $0-2 = 22$.

The closeness measure defines how well a point k matches a stay point s, but it does not capture the potential crowding that is contextual to this match. Indeed, if different anonymous users are spatiotemporally close to k, the latter will obtain a high closeness to the stay points of the former's personal gazetteers. To this end, we introduce the point uniqueness[4] as a measure of the actual discriminating power of a point k with respect to S.

Definition 8 (Point uniqueness) *The uniqueness of a point k with respect to S is:*

$$unq(k,S) = \frac{\sum_{s \in S} closeness(k,s)}{\sum_{S' \in \mathscr{S}} \sum_{s' \in S'} closeness(k,s')}$$

The intuition is that the closer k is to stay points of different personal gazetteers, the lower will be its discriminating power with respect to S. Notice that k may be close to more than one stay point belonging to the same personal gazetteer; this is why the point uniqueness with respect to S positively considers the closeness with all the stay points in S.

Ultimately, we define the local score of a couple (k,s) by combining the measures of closeness and point uniqueness.

Definition 9 (Local score) *The local score of a couple* (k,s) *is:*

$$localScore(k,s) = closeness(k,s) \cdot unq(k,S)$$

5.3.2 The global scoring function

The global score quantifies how well a social trajectory K matches a personal gazetteer S. Let us start by defining the mapping of K into S by means of a *mapping signature*, which associates each point $k \in K$ to the spatiotemporally closest stay point $s \in S$.

Definition 10 (Mapping signature) *Given a social trajectory K and a personal gazetteer S, the mapping signature represents the mapping of K into S as set of couples* $M = \{(k_1,s_1),...,(k_n,s_n)\}$. *It is obtained through a partial function* $\varphi : K \to M$ *that maps each point* $k \in K$ *to the stay point* $s \in S$ *that maximizes the local scoring function* $localScore(k,s)$ *(Section 5.3.1). Formally,* $\varphi(k) = \text{argmax}_{s \in S} localScore(k,s) \mid localScore(k,s) > 0$.

The mapping function φ is partial as a point $k \in K$ may be too spatiotemporally far from any of the stay points in S — and the same holds the other way around. The global score between K and S is calculated on the obtained mapping signature M.

[4]We refer to this measure as *point uniqueness* in order to differentiate it from the *uniqueness* introduced in Section 5.2.2, which applies the same principle to sets of stay points.

Definition 11 (Global score) *Let us refer to K_M and S_M as the set of points and of stay points that belong to the mapping signature M, respectively. Also, for a given $s \in S_M$, consider $k^* \in K_M$ the point mapped to s with the highest local score, i.e., $k^* = \text{argmax}_{k \in K_M} localScore(k,s)$. Then, the* global score *of the mapping signature M is:*

$$globalscore(M) = unq(S_M) \cdot \sum_{s \in S_M} localScore(k^*, s)$$

Whereas other works (e.g., [38]) simply aggregate the local scores, our global scoring function is based on a qualitative and a quantitative principle. The first part of the equation (i.e., the multiplication by the uniqueness of the matched stay points, which is defined in Section 5.2.2) evaluates the quality of the mapping in terms of the discriminating power of the matched stay points: the higher their uniqueness, the higher is the final score. The second part of the equation (i.e., the sum of closeness values) evaluates the quantitative aspect of the mapping by summing the local scores. More specifically, we consider only the maximum local score for each matched stay point in order to favor the mappings that cover a higher number of stay points and, therefore, are more representative of the personal gazetteer.

5.3.3 The assignment step

Following the calculation of the global score between each couple in $(\mathscr{K}, \mathscr{S})$, we are left with the problem of assigning to each K its most probable match S. As anticipated, we want to avoid the assignment of the same K to multiple personal gazetteers (and vice versa) and to maximize the chosen global scores. This problem is well known in the literature as the assignment problem [162]. Let $G = (\mathscr{K}, \mathscr{S}, E)$ be a weighted bipartite graph where the edges E correspond to the global scores between the social trajectories and the personal gazetteers. Then, the assignment simply consists in finding the maximum-weight matching.

Finally, DART's approach is summarized in Algorithm 2. Notice that the computation of the global score between a known trajectory K and a personal gazetteer S cannot be modeled as a simple function of K and S; this is because the computation of point uniqueness (which is a component of the local score) requires a global knowledge of the closeness values between every point k and every stay point s. Thus, we start by computing closeness values between every point k and every stay point s (function *computeCloseness*, Line Algorithm 2), then we compute point uniqueness values between every point k and every personal gazetteer S (function *computePointUnq*, Line 2). At this point, we can iterate through the (K,S) couples to determine local scores, mappings and global scores (Lines 4 to 7). The *computeAssignment* function (Line 8) finally computes the maximum-weight matching on the weighted bipartite graph.

Algorithm 2 DART algorithm

INPUT $\mathscr{S}$: personal gazetteers; $\mathscr{K}$: social trajectories; φ: mapping function.
OUTPUT matches between personal gazetteers and social trajectories.
1: $C \leftarrow computeCloseness(k,s)\ \forall\ (k,s) \in (\mathscr{K},\mathscr{S})$
2: $U \leftarrow computePointUnq(k,S,C)\ \forall\ (k,S) \in (\mathscr{K},\mathscr{S})$
3: $E \leftarrow \varnothing$
4: **for all** $(K,S) \in (\mathscr{K},\mathscr{S})$ **do**
5: $\quad L \leftarrow computeLocalScore(k,s,C,U)\ \forall\ (k,s) \in (K,S)$
6: $\quad M \leftarrow computeMapping(K,S,L,\varphi)$
7: $\quad E \leftarrow E \cup \{K,S,computeGlobalScore(M,L)\}$
8: **return** $computeAssignment(\mathscr{K},\mathscr{S},E)$

5.4 Evaluation

Two anonymous datasets are used, whose statistics are reported in Table 5.1. The first one is a proprietary, real-world dataset based on a case study scenario centered on (but not limited to) the Italian city of Milan; for brevity, we will refer to this dataset as MIL. It spans from September 2017 to December 2017 and includes nearly 500,000 anonymous GPS trajectories. For the sake of simplicity, we extract from this dataset a sample of 100 trajectories of users whose personal gazetteers indicate that they live and work in Milan. As we do not possess ground-truth data of social trajectories, we simulate the latter by generating them from the anonymous ones. Each generated trajectory contains a random 1% sample of the original one[5], where each point is injected with a spatial and a temporal jitter, misplacing it of up to 100 meters and up to 30 minutes. Different scenarios are simulated by generating different sets of social trajectories. First, we vary the percentage of points located at a stay point (i.e., whether people interact more on social networks while commuting or while at a stay point); we consider a point p to be located at a stay point s if $dist_{space}(p,s) \leq 100$ meters. Also, we vary the temporal misalignment between the social and the anonymous trajectories. The second dataset is the Mobile Data Challenge (MDC) dataset [163], which provides real-world data of nearly 200 individuals from the Lake Geneva region (Switzerland) in 2010-2011. For each user, it provides a sample of his/her personal gazetteer (which we will use as ground truth to verify the stay points inferred by DART), as well as the trajectories captured by GPS satellites and WLAN devices in a complementary way (i.e., either one of them is active for a single user at the same time, not both). In this case, we use the GPS satellite as the source of anonymous trajectories and the WLAN as the source of the social ones.

MIL is by far the most challenging dataset due to the higher spatiotemporal density of the trajectories. In particular, GPS trajectories in MDC are often disjoint from the temporal perspective. For this reason, our evaluation mostly focuses on the former dataset. Nonetheless,

[5]The number of social interaction is usually quite low when compared to the frequency of GPS signals.

Table 5.1: Statistics about the MIL and MDC datasets

Parameter	MIL	MDC
Number of users	100	190
Avg number of pings per user (anonymous)	2040	13170
Avg number of pings per user (social)	22	9123
Avg number of stay points per user	5	21
Avg number of days per user	28	185

the unusual characteristic of MDC (i.e., the complementarity of GPS and WLAN sources) allows us to further evaluate the robustness of the approach in a different scenario.

A prototypical implementation of DART has been realized on a big data infrastructure, consisting of 18 Ubuntu machines mounting the Cloudera Distribution for Hadoop (CDH) 6.2.0; the source code is available on GitHub[6]. The datasets are stored as Parquet tables on Apache Hive, while the approach is implemented on Apache Spark.

5.4.1 Stay points validation

In Section 5.2 we have briefly discussed the Patchwork algorithm that we adopt to infer stay points from GPS trajectories. In this Section, we aim to verify the correspondence between the stay points that we infer and the ground truth provided by the MDC dataset — even though it is only a sample of the user's personal gazetteers. In MDC, stay points are simply called *places*; some of them have semantic labels attached (e.g., distinguishing home places from working locations and other kinds of stay points), but none of them contains spatial information. However, since the dataset provides the starting and ending timestamps of each visit to these places, we are able to estimate their location by averaging the latitude and longitude recorded for the user between such timestamps.

The MDC sample contains 493 places (i.e., 2.6 for each user on average), whereas our stay point extraction algorithm infers 3714 (i.e., 19.5 for each user on average). We have verified that 84% of MDC places have a stay point in its close vicinity (i.e., no more than 200 meters apart). Our implementation of Patchwork fails to recognize the remaining places due to either the low volume or the sparsity of geolocated records, which hinders the recognition of a cluster. Conversely, only 11% of our stay points can be traced back to MDC places, meaning that 89% are new with respect to the data provided by MDC.

[6]https://github.com/big-unibo/dart

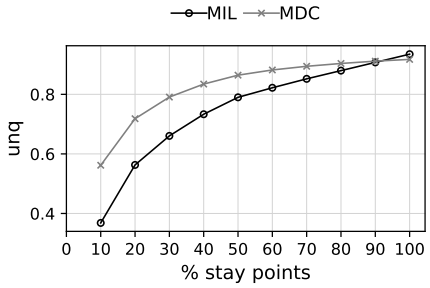


Figure 5.5: Uniqueness probability of subsets of stay points from personal gazetteers.

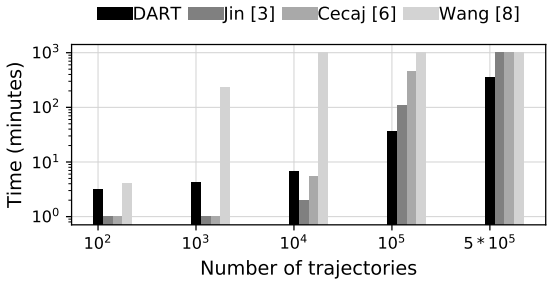


Figure 5.6: Efficiency of DART and other de-anonymization approaches on the MIL dataset considering a progressively increasing number of trajectories.

5.4.2 Stay points uniqueness

The fundamental assumptions in DART are that (1) users' social interactions mainly happen on stay points, and (2) personal gazetteers are often unique and enable the de-anonymization of users. Figure 5.5 verifies the latter statement in the MIL and MDC datasets: given a certain percentage of stay points from each user's personal gazetteer, it shows the average probability that such subset is unique within the population (Definition 6 in Section 5.2.2). The Figure not only confirms that the personal gazetteer is indeed a unique signature, but it also indicates that even half of it is enough to identify about 80% of the population in MIL. The easier de-anonymizability of MDC is also confirmed, as only one third of stay points are enough to identify 80% of the population. Ultimately, we conclude that we can effectively rely on stay points to de-anonymize trajectories.

5.4.3 Efficiency of DART

State-of-the-art de-anonymization approaches usually focus on the effectiveness of the algorithm and overlook the evaluation of its efficiency, even though the importance of

designing scalable and efficient algorithms in order to enable their applicability on large-scale datasets is often emphasized in the literature [104, 195]. We implemented DART on Apache Spark in order to guarantee the scalability of the approach to large volumes of data. To evaluate DART's efficiency, we measure its execution time on MIL by progressively increasing the number of users considered in the dataset (from 100 to the whole 500,000) and compare it with the sequential implementations (in Python) of Cecaj et al. [38], Jin et al. [148], and Wang et al. [278][7].

The configuration of Spark for DART's execution consists of 10 executors with 3 cores and 10GB of RAM for each executor. For the sake of a fair comparison, we parallelized the algorithms of [38] and [278] as bags-of-tasks and ran 30 concurrent instances that independently worked on 1/30th of the dataset[8]. Conversely, the algorithm of [148] cannot be parallelized (in fact, it requires the whole dataset to run its TF-IDF strategy to compute the signatures of trajectories), thus it has been executed on one of the available machines.

The results are shown in Figure 5.6; we remark that both axis are logarithmic and that executions that took more than a day have been halted and are shown as full-sized bars. The parallel nature of DART's implementation entails a slight disadvantage when working on small datasets (due to the overhead for managing distributed computations), but it greatly rewards when the dataset's size becomes significantly large. [278] immediately reaches an exaggerate execution time, even with a medium-sized dataset. This is probably due to its complex algorithm, which relies on a Hidden Markov Model to fill incomplete trajectories by enumerating all possible complete trajectories; although the algorithm allows [278] to achieve fairly good results (as we will show in Section 5.4.4), it sensibly hinders its application on large-scale datasets. As for [38], it reaches high execution times as well, with a linear increase with respect to the increase of trajectories. In particular, the execution time in [38] is mainly given by the pairwise comparison of the point of every couple of trajectories, whereas we use spatial joins to compute local scores only between points that are close enough. The execution time of [148] follows a pattern similar to [38]. Considering that [148] uses $1/30$th of the computation power, it is actually considerably fast; indeed, its computation time is mainly due to the generation of trajectories' signatures, which are easily compared once they have been computed. However, as we will show in Section 5.4.4, [148] trades a faster computation with a low accuracy in almost every scenario.

[7] To the best of our knowledge, [148] and [278] are the most recent related approaches in the literature.

[8] Differently from DART, the computation of the score between every (K,S) in [38] and [278] is independent of the others.

5.4.4 Effectiveness of DART

In this section, we evaluate the effectiveness of the de-anonymization approach on both datasets by means of the *accuracy*, defined as the fraction of correctly guessed assignments over the actual number of users. The results are compared with [38, 148, 278].

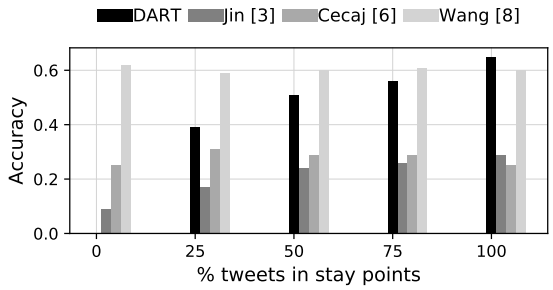
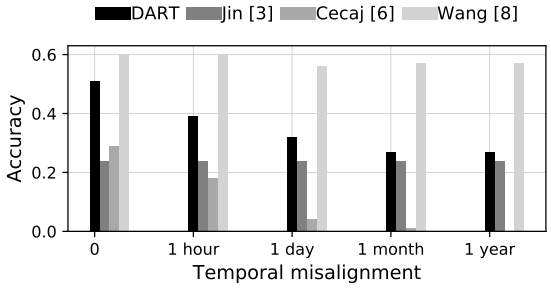


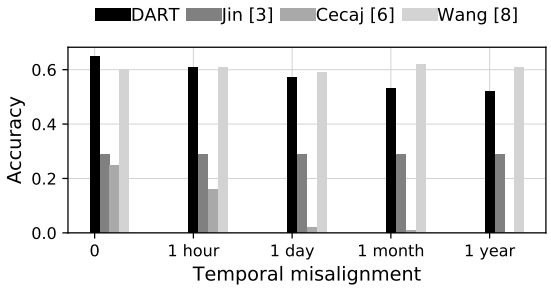
Figure 5.7: Accuracy of DART and other de-anonymization approaches on the MIL dataset.

In the first experiment, we have created several sets of social trajectories on MIL by varying the percentage of points located at a stay point. Considering the scenario of social trajectories being Twitter histories, the extremes are that users always tweet while traveling (i.e., 0% of points within a stay point) and that users always tweet from home, work, or other stay points (i.e., 100% of points within a stay point). The results are shown in Figure 5.7. Except for the case where points in K always fall outside of stay point (i.e., what we assume to be a very unlikely scenario), DART performs better than [38] and [148]. In the former scenario, the poor performance of DART is physiological, as points outside of stay points are not considered. Indeed, the results also show that DART's accuracy (as well as the accuracy of [148]) increases as the percentage of points in K located at stay points also increases (whereas [38] is unaffected by such variation). Interestingly, the fact that the accuracy of [148] (which uses a TF-IDF strategy to summarize each trajectory) follows the same trend of DART confirms that stay points are signatures traceable to unique users; therefore, we conclude that the more data is located at stay points, the easier it is to de-anonymize it. Ultimately, the comparison with [278] shows that the latter achieves consistently good accuracy, comparable to the one of DART in the latest scenarios. Indeed, [278] adopts a complex technique based on a Hidden Markov Model to estimate the user's location when the original trajectory is not complete. However, as shown in Section 5.4.3, the good accuracy of this approach comes at the expense of computational complexity, which hinders its adoption on large-scale datasets.

A second experiment verifies the robustness of DART to the temporal misalignment between social trajectories and personal gazetteers. Starting from the 50% and 100% con-



(a) The temporal misalignment starts from the 50% configuration in Figure 5.7.



(b) The temporal misalignment starts from the 100% configuration in Figure 5.7.

Figure 5.8: Accuracy of DART and other de-anonymization approaches on the MIL dataset considering a progressive temporal misalignment between personal gazetteers and social trajectories.

figurations of the previous experiment, we simulate the misalignment by adding a certain temporal delta to every point in the social dataset. The results are shown in Figure 5.8. As in the previous experiment, [278] is able to maintain high accuracy, even though its execution times remain severely high. DART's accuracy is actually affected by the temporal misalignment, but the behavioral component of DART is able to maintain the former always above 27% and 52% in the respective scenarios. This demonstrates the robustness of DART, as stay points are proven as a solid bedrock for the de-anonymization. The accuracy of [148] is constant because their algorithm does not consider the temporal dimension; although this characteristic makes the approach robust to temporal misalignment as well, it prevents [148] from achieving better accuracy when the temporal dimension is more relevant. Ultimately, the accuracy of [38] progressively decreases down to zero; this is inevitable, as [38] only looks for punctual temporal matches.

Finally, we evaluate the accuracy of DART on MDC. On this dataset we observe a significant difference between the four approaches: as shown in Figure 5.9, DART is able to achieve an optimum result (i.e., a 91% accuracy) and performs far better than

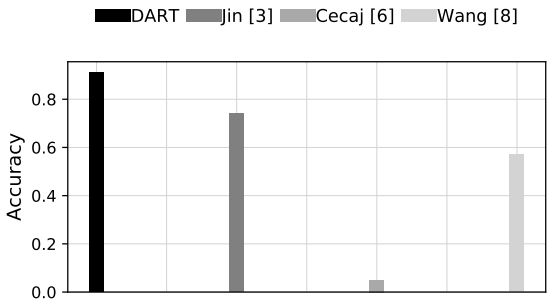


Figure 5.9: Accuracy of DART and other de-anonymization approaches on the MDC dataset.

the other algorithms.The poor performance of [38] is due to its algorithm, which abruptly excludes a match between K and A if there exists a record of A being too far from K (i.e., $dist_{space}(a,k) > 2 \cdot \tau_s$) in a close time windows (i.e., $dist_{time}(a,k) \leq \tau_t$). This is called the *exclusion rule* in [38], and it is based on the principle that a person cannot be in two places at the same time. However, the complementary nature of the original GPS and WLAN trajectories in MDC makes it easier for the exclusion rule to be activated. Conversely, the good performance of DART and [148] (which is much higher than the one on MIL) is due to the trajectories in MDC being less overlapped; in fact, all trajectories in MIL are bounded within the city of Milan over one month, whereas in MDC they are centered on the Geneva lake and can span up to Zurich over one year (i.e., a much larger spatiotemporal area). Interestingly, [278] is unable to achieve a better accuracy than the one reached in MIL. This is probably due to the complementarity of GPS and WLAN sources (mentioned at the beginning of this Section): the behavior of GPS and WLAN trajectories in MDC is quite different, because the former mostly captures movements between stay points, while the latter captures visits to stay point. To this end, MDC is similar to the last scenario in MIL where tweets are mostly made within stay points (Figure 5.7). We conclude that [278] achieves a consistent accuracy in different scenarios, but it is unable to exploit the one in which the concentration of social interactions is mostly around stay points (which, as we recall, is one of our main assumptions).

5.5 Conclusion

We proposed an original approach to trajectory de-anonymization that relies on personal gazetteers to model users' behaviors and to effectively capture and recognize each user's spatiotemporal signature. Our experimental evaluation has proven that DART improves recent state-of-the-art algorithms by effectively coupling an efficient implementation with

high effectiveness on different datasets and scenarios. Our algorithm is also robustness to temporal misalignment between the anonymous and known trajectories. The only related approach that achieves better results in some scenarios is [278], whose elevated complexity severely hinders its applicability on large-scale datasets. At this stage we are currently working in different directions to further improve the effectiveness of the approach by: enhancing the behavioral model to improve its effectiveness in case of significant temporal misalignment, e.g., by considering not only the hour of the day but also the day of the week; and by comparing it with other state-of-the-art algorithms, to improve our understanding as to which scenarios are better handled by different algorithms, with particular focus on those scenarios where [278] achieves higher accuracy.

Chapter 6

Conclusion and Research Directions

In the first part of the thesis, we investigated how the inclusion of unconventional data affects the knowledge pyramid by requiring specific techniques for different data types. We focused on mobility data (i.e., spatiotemporal trajectories) since they fuel location-based services at the core of pervasive applications such as traffic analysis, route and point-of-interest recommendation, and even smart-phone videogames. In particular, we addressed two of the issues posed by mobility data: scalability and privacy.

As for scalability, the ever-increasing number of positioning devices (e.g., smart-phones and geolocalized networks) is challenging traditional trajectory mining approaches (e.g., clustering and map matching) demanding for big data solutions. While we proposed an approach to rethink map matching in a distributed fashion, our broader goal is to address trajectory mining applications in big-data scenarios with high variance in means of transportation (e.g., car, walk, bicycle), sampling rates (e.g., from seconds to minutes), and road networks (e.g., extending the map-matching from Milan to the entire country). On the one hand, this requires the introduction of an innovative approach capable to integrate and analyze spatiotemporal positions at different resolutions (e.g., a GPS point with a limited accuracy or a wide area traced by GSM antennas). The adoption of auto-machine learning techniques [134] for the pre-processing and integration of spatiotemporal data is an interesting research direction (e.g., by automating transformations such as noise filtering, trajectory segmentation or uncertainty reduction). On the other hand, smarter indexing and distributed strategies are needed to scale, since the existing big data frameworks are "limited to" to low-level spatiotemporal operations (e.g., topological operations or k-nearest neighbors rather than clustering or frequent pattern mining).

As for privacy, the sensitive nature of mobility data (e.g., home and work locations) easily identifies the individuals (i.e., moving objects) producing the mobility traces. We investigated how trajectory data can be anonymized for privacy protection, and how the frequently visited

locations can be used to de-anonymize individuals within a trajectory dataset. Most of the (de-)anonymization approaches hardly scale up to big data volumes. Our goal is to investigate whether it is possible to further improve (de-)anonymization accuracy without burdening the efficiency of the involved algorithms. In this direction, an empirical survey on the effectiveness and efficiency of the existing anonymization algorithms is necessary to put a common ground for comparison and to highlight the existing bottlenecks. We plan to collect—possibly with the help of volunteers—a large-scale ground-truth dataset to validate the goodness of the studied anonymization approaches, and to measure the robustness of DART against such anonymization algorithms. Additionally, while the majority of anonymization algorithms focus on static protection (i.e., anonymization requires the elaboration of the *whole* dataset), dynamic trajectory protection is a promising research direction.

Part II

Advanced Analytics

Chapter 7

Introduction

7.1 Motivation and contributions

Business Intelligence (BI) includes a wide range of techniques and tools supporting data transformation in the decision-making process (through the knowledge pyramid in Figure 1.1). Until the '80s, enterprise databases stored operational data. Since then, the increasing data variety (e.g., social networks, IoT, and mobility data) has made clear that enterprise data alone are not sufficient to back strategic decisions, demanding the integration of unconventional data. However, the exponential increase in data volume affects the access to strategic knowledge hidden in the data (Figure 7.1), requiring advanced analytic techniques and tools supporting data scientists. Additionally, BI is becoming more and more pervasive [275], entailing increasing participation of data scientists with high competence in the business domain but low competence in computer science and data engineering skills. Enabling effective participation requires the investigation of user-centered paradigms supporting analytical querying, making knowledge extraction and narration accessible to everyone [216, 215]. So far, "classical" analytic tools required data scientists to write (formal) queries to access data. However, this is unfeasible in scenarios where data scientists have no keyboard at hand (e.g., in the case of analytics via augmented reality). Additionally, there is a need to increase the abstraction level in defining analytical queries. Rather than demanding formal queries, analytic systems should be capable (i) of intercepting and interpreting data scientists' intentions, and (ii) of providing concise representations of the results.

In this context, we contribute to the following research directions.

Augmented OLAP. Augmented reality allows users to superimpose digital information (typically, of operational type) upon real-world objects. The synergy of analytical frameworks and augmented reality opens the door to a new wave of situated analytics, in which users

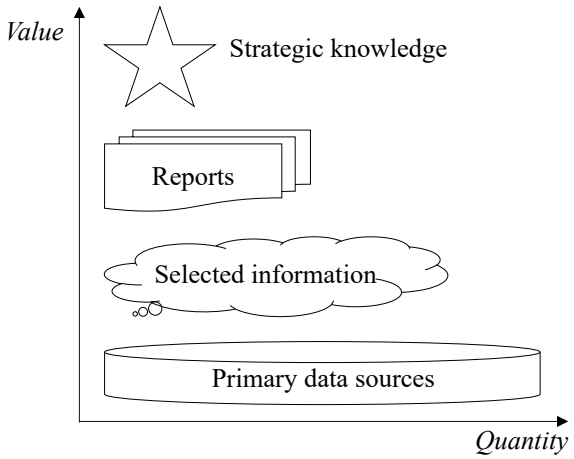


Figure 7.1: Information value pyramid: the higher the quantity, the lower the strategic value (a variation of Figure 1.1 highlighting the relationship between quantity and value; adapted from [113]).

within a physical environment are provided with immersive analyses of local contextual data. We introduce a novel approach named *Augmented OLAP* that, based on the sensed augmented context (provided by wearable and smart devices), proposes a set of relevant analytical queries to the user [95, 96]. This is done by relying on a mapping between the objects that can be recognized by the devices and the elements of the enterprise multidimensional cubes, and also by taking into account the queries preferred by users during previous interactions that occurred in similar contexts. A set of experimental tests evaluates the proposed approach in terms of efficiency, effectiveness, and user satisfaction.

Conversational OLAP. The democratization of data access and the adoption of OLAP in scenarios requiring hand-free interfaces push towards the creation of smart OLAP interfaces. We introduce COOL, a framework devised for conversational OLAP applications [88–90]. COOL interprets and translates a natural language dialogue into an OLAP session that starts with a GPSJ (Generalized Projection, Selection and Join) query and continues with the application of OLAP operators. The interpretation relies on a formal grammar and a knowledge base storing metadata and values from a multidimensional cube. In case of ambiguous or incomplete text description, COOL can obtain the correct query either through automatic inference or through interactions with the user to disambiguate the text. Our tests show very promising results in terms of effectiveness, efficiency, and user experience. Besides adding novel support to the interpretation and translation of complete analytical OLAP sessions, COOL achieves state-of-the-art accuracy in the interpretation of GPSJ queries.

Intentional OLAP. The Intentional Analytics Model (IAM) has been recently envisioned as a new paradigm to couple OLAP and analytics [275, 93, 44]. It relies on two basic ideas: (i) letting the user explore data by expressing her analysis intentions rather than the data she needs, and (ii) returning enhanced cubes, i.e., multidimensional data annotated with knowledge insights in the form of model components (e.g., clusters). We provide a proof-of-concept for the IAM vision by delivering an end-to-end implementation of describe, one of the five intention operators introduced by IAM [44]. Among the research challenges left open in IAM, those we address are (i) automatically tuning the size of models (e.g., the number of clusters), (ii) selecting the most effective chart or graph for visualizing each cube depending on its features, and (iii) devising a visual metaphor to display enhanced cubes and interact with them.

Multidimensional summarization. Multi-level and multi-dimensional data provide a rich description of events through multiple features each at different levels of detail. However, the analysis effectiveness is a compromise between the precision and the size of the information being displayed while analyzing multi-dimensional cubes [231]. Since multidimensional cubes are represented in analytic dashboards with pivot tables, the comprehensibility of such tables becomes is inversely proportional to the quantity of the returned cells (i.e., information flooding problem). To cope with information flooding, through summarization, we help data scientists in getting succinct data representation, especially when multidimensional hierarchies are involved [94, 97].

7.2 Structure

This part of the thesis hinges on the above-mentioned contributions and is organized as follows. In Chapter 8, we introduce the foundational concepts of BI and multidimensional analysis necessary for the comprehension of the subsequent chapters. Chapter 9 introduces Augmented OLAP, a query recommender system based on contextual data. Chapter 10 describes Conversational OLAP, a framework enabling the interpretation of OLAP queries expressed in natural language. Chapter 11 makes a step towards the IAM by designing the describe OLAP intention. Chapter 12 introduces a novel summarization approach that leverages the multidimensional nature of spatial data to group similar patterns. Finally, Chapter 13 draws the conclusion and future research directions.

Chapter 8

Background

In this chapter, we introduce the foundational concepts of BI. We present the data warehouse, a data repository at the core of decision support systems. Then, we introduce the OLAP operators to query a data warehouse. Finally, we provide a formal definition of the multidimensional model.

8.1 Data warehouse

Data warehousing is a collection of methods, techniques, and tools supporting knowledge workers in data analysis. The main actor is the *data warehouse*, a data collection supporting the decision-making process. A data warehouse is subject-oriented (i.e., focuses on specific analysis concepts), integrated and consistent (i.e., brings multiple data sources in a unified view), and not volatile (i.e., covers historic data) [144]. In [115], the authors identify the following requirements for a data warehouse: *accessibility* to data scientists unfamiliar with IT and data structures; *integration* of data on the basis of a standard model; *query flexibility* to maximize the knowledge obtained from data; *information conciseness* for effective analyses; *multidimensional* and intuitive representation; and *correctness and completeness* of the integrated data. A data warehouse differs from operational databases. Operational databases store operational data, i.e. data created by business operations involved in daily management processes (e.g., purchase and sales management). The management of operational data is characterized by a transactional workload, i.e. a high number of users interacting with the database to carry out read/write operations.

The query workload involving a data warehouse is known as OLAP (On-Line Analytic Processing), a dynamic and multidimensional analysis of aggregated data. An OLAP workload is characterized by a few users interacting with the data warehouse to carry out complex read operations (e.g., identifying the products maximizing profits). OLAP queries operate

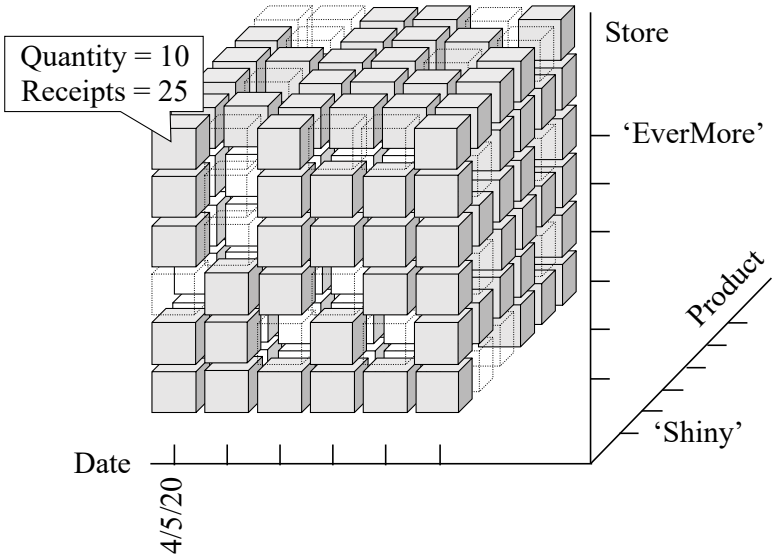


Figure 8.1: Example of a multidimensional Sales cube, where analysis dimensions are Date, Product and Store, and measures are Quantity and Receipts (adapted from [113]).

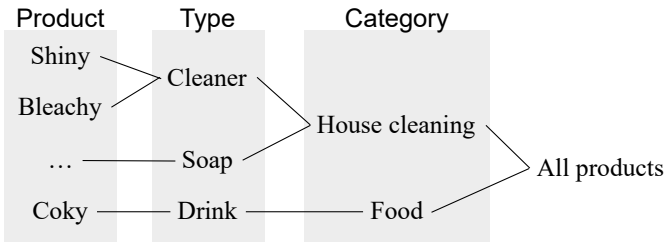


Figure 8.2: Example of hierarchy on Product. Product, Type, Category are hierarchy levels also called dimensional attributes (adapted from [113]).

on a multidimensional representation of a data warehouse, where a data item is a point in a space with many analysis dimensions and is described by numerical measures. These notions recall the visual metaphor of a hyper cube (henceforth, cube) to represent multidimensional data. In this metaphor, cube edges are analysis dimensions, data items are cube cells, and each cell is given a value for each measure (Figure 8.1). Dimensions are associated with (roll-up) *hierarchies* that aggregate *levels* (called dimensional attributes; Figure 8.2). The actual values of each level (e.g., *Shiny* is a specific Product) are referred to as *members*. The dimensional fact model (DFM) [112] conceptualizes the design of a data warehouse and its cubes (Figure 8.3).

Multidimensional cubes are implemented atop different abstractions: MOLAP (Multidimensional OLAP) uses multidimensional structures (e.g., multidimensional vectors), ROLAP (Relational OLAP) uses relations from the standard relational model, and HOLAP (Hybrid OLAP) adopts a hybrid model that relies on both multidimensional and relational models.

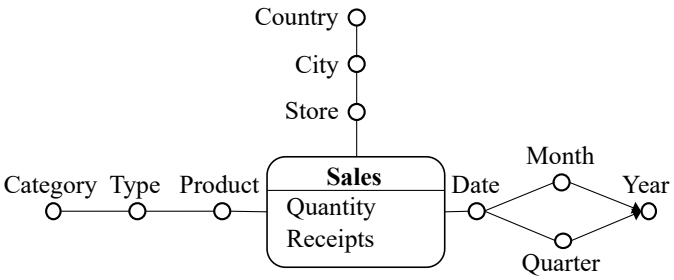


Figure 8.3: Conceptualization of the Sales cube in the Dimensional Fact Model (DFM) [112].

FT_Sales(Date,Product, Store , Quantity, Receipts)
DT_Time(Date,Month, Quarter, Year)
DT_Store(Store,City, Country)
DT_Product(Product, Type, Category)

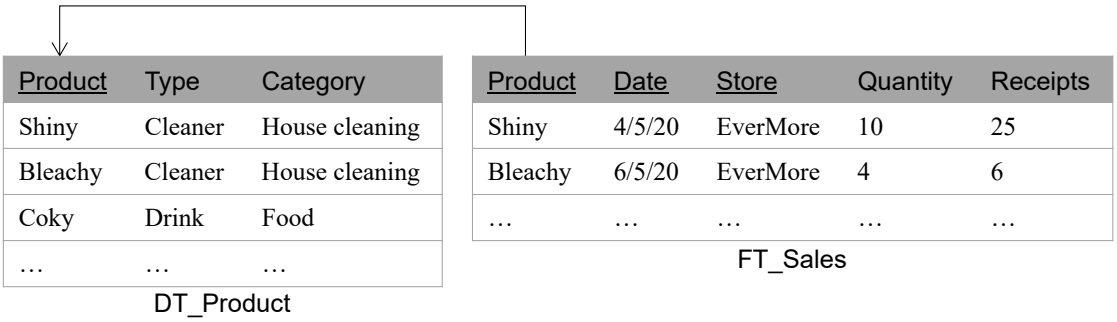
Figure 8.4: Relational representation of the Sales cube.

While MOLAP-based systems deliver the best performance through the natural representation of multidimensional data, data sparsity is a major drawback since multidimensional DBMSs store every cell in the fact space [55]. ROLAP-based systems rely on decades of evolution of relational DBMSs and received more research attention than MOLAP and HOLAP. ROLAP systems enable scalability, since there is no sparsity in relational systems when storing multidimensional data (Figure 8.4). In this thesis, we adopt the ROLAP model along with the *star schema* modeling technique. A star schema consists of (i) relations called dimension tables DT corresponding to cube hierarchies, and (ii) relations called fact tables FT corresponding to cubes (Figure 8.5). A FT references dimension tables (i.e., the dimensions of the cube), and includes an attribute for every measure. The primary key of the FT composes the keys imported from the referenced DTs. Every DT has a primary key and an attribute for each hierarchy level. Dimension tables denormalize all the hierarchy levels in the same relation. Whereas this causes redundancy, it significantly reduces the number of joins needed to retrieve information.

8.2 OLAP operators

Data in a data warehouse are analyzed at different levels of aggregation by applying a sequence of *OLAP operators*. Each operator is applied to the outcome of the previous one, creating a navigation path known as *OLAP session*. In this thesis, we refer to the following operators.

- *Roll-up* aggregates data at a coarser level by removing a detail level from a hierarchy (e.g., from Product to Type).



Product	Type	Category
Shiny	Cleaner	House cleaning
Bleachy	Cleaner	House cleaning
Coky	Drink	Food
…	…	…

DT_Product

Product	Date	Store	Quantity	Receipts
Shiny	4/5/20	EverMore	10	25
Bleachy	6/5/20	EverMore	4	6
…	…	…	…	…

FT_Sales

Figure 8.5: Example of dimension and fact table for the Sales cube.

- *Drill-down* aggregates data at a finer level by adding a detail level from a hierarchy (e.g., from Type to Product).
- *Slice-and-dice* combines *slice* and *dice* operators. *Slice* reduces the analysis dimensions by collapsing a dimension to a specific member (e.g., Product = "*Shiny*"). *Dice* reduces the retrieved data by a selection predicate (e.g., Year *between* 2018 *and* 2020). different viewpoint.
- *Drill-across* links concepts in interrelated cubes.

8.3 (Formal) Multidimensional model

We now introduce the formal setting necessary to manipulate multidimensional data (we summarize the notation symbols in Table 8.1 at the end of the section).

Definition 12 (Hierarchy) *A* hierarchy *is defined as a triple* $h = (L_h, \succeq_h, \geq_h)$ *where:*

1. L_h *is a set of categorical* levels*; each level* $l \in L_h$ *is coupled with a* domain $Dom(l)$ *including a set of* members *(all domains are disjoint);*
2. $\succeq_h$ *is a* roll-up *partial order of* L_h *(is restricted to a* total *order for* linear *hierarchies); and*
3. $\geq_h$ *is a* part-of *partial order of* $\bigcup_{l \in L_h} Dom(l)$.

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

Example 3 (Hierarchy) *For the hierarchy* Time*, it is* Date $\succeq$ Month $\succeq$ Year*. The hierarchy* Product *is linear. 02/12/2020 is a member of* Date *and 02/12/2020* $\geq$ *12/2020* $\geq$ 2020. □

Aggregation is the basic mechanism to query cubes, and it is captured by the following definition of group-by set. As normally done when working with the multidimensional model, if a hierarchy h does not appear in a group-by set it is implicitly assumed that a complete aggregation is done along h.

Definition 13 (Group-by Set) *Given a set of hierarchies H, a* group-by set *of H is a subset G of levels in the hierarchies of H including at most one level for each hierarchy. The roll-up order on the hierarchies of H induces a partial order on the group-by sets of H as follows:*

$$G \succeq G' \Leftrightarrow \forall l' \in G' \; \exists l \in G \text{ s.t. } l \succeq_h l'$$

The complexity of OLAP queries is also due to the collection/integration of data from different data sources. To facilitate OLAP analyses, the DW is typically partitioned 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.

Definition 14 (Cube, Cube Schema, and Data Mart) *Given a set of hierarchies H, a* cube *over H is defined as a triple $c = (G_c, M_c, \omega_c)$ where:*

1. *G_c is a* group-by set *of H;*
2. *M_c is a set of numerical measures; each measure $m \in M_c$ is coupled with one aggregation operator $op(m) \in \{\texttt{sum}, \texttt{avg}, \ldots\}$; and*
3. *ω_c is a partial function that maps the tuples of members for the levels of G_c to a numerical value for each measure $m \in M_c$.*

A cube schema *is a couple $\mathscr{C} = (H, M)$. A* data mart *DM is a couple of a set of hierarchies H and a set C of cubes over H, $DM = (H, C)$. Each coordinate that participates in ω_C, with its associated measure values, is called a* cell *of c.*

The levels, members, and measures of cube c are given the generic name of *md-elements* of c. Note that the ω_c is defined as partial since cubes are normally *sparse*; the *cardinality* of c is the number of tuples of members that are mapped through ω_c and is denoted with $|c|$. With a slight abuse of notation, we write $\chi \in c$ to state that χ is a cell of c.

Example 4 (Cube) *Given* $\mathsf{Sales} = (H, M)$ *(Figure 8.1), it is*

$$H = \{\mathsf{Time}, \mathsf{Product}, \mathsf{Store}\},$$
$$M = \{\mathsf{Quantity}, \mathsf{Receipts}\}$$

and $op(\mathsf{Quantity}) = op(\mathsf{Receipts}) = \mathtt{sum}$ *(i.e., all measures are addictive).* □

Cubes are queried through GPSJ (Generalized Projection / Selection / Join) queries, a well-known class of queries that was first studied in [124]. A GPSJ query is composed of joins, selection predicates, and aggregations. Remarkably, having all the cubes in *DM* defined over the same set of hierarchies H corresponds to assuming that the cubes share a set of *conformed dimensions*, which enables the formulation of *drill-across queries* (queries joining two or more cubes).

Definition 15 (Query) *A query q on data mart $DM = (H, C)$ is a triple $q = \langle G_q, P_q, M_q \rangle$ where*

- G_q *is a group-by set of* H*;*
- P_q *is a (possibly empty) set of selection predicates each expressed over one level of* H*;*
- $M_q \subseteq M$ *is the set of measures whose values are returned by* q*.*

Let $C_q \subseteq C$ be the subset of cubes such that at least one of their measures is part of M_q; query q is well-formed if all the measures it returns are available at the required granularity:

1. $\forall c \in C_q \; G_c \succeq G_q$ *if* $M_q \neq \varnothing$*;*
2. $\exists c \in C$ *s.t.* $G_c \succeq G_q$ *if* $M_q = \varnothing$ [1]

Example 5 (GPSJ query) *With reference to the* Sales *cube, the query asking for quantity of sold products for each month of 2019 and each product type is $q = \langle G_q, P_q, M_q \rangle$, with*

$$G_q = \{\mathsf{Month}, \mathsf{Type}\}$$
$$P_q = \{(\mathsf{Year} = 2019)\}$$
$$M_q = \{\mathsf{Quantity}\}$$

[1] We assume that, when $M_q = \varnothing$, query q asks for a count of the cardinality of G_q, so the existence of at least a cube at the required granularity ensures that q is well-formed.

Table 8.1: Notation summary

Notation	Meaning
c	Multidimensional cube
C	Set of cubes
$\mathscr{C}$	Cube schema
DM	Data mart
G	Group by set
H	Set of hierarchies
l	Hierarchy level
M	Set of measures
P	Selection predicate
q	Query
ω_c	Assign measure values to cube cell
χ	Cube cell

The SQL formulation of q on a star schema featuring fact table `FT_Sales` *and dimension tables* `DT_Time` *and* `DT_Product` *is*

```
SELECT DT_Time.Month, DT_Product.Type, sum(FT_Sales.Quantity)
FROM FT_Sales
    JOIN DT_Time ON (FT_Sales.Date = DT_Time.Date)
    JOIN DT_Product ON (FT_Sales.Product = DT_Product.Product)
WHERE DT_Time.Year = 2019
GROUP BY DT_Time.Month, DT_Product.Type
```

□

Definition 16 (Base Cube and Derived Cube) *A cube whose group-by set G_c includes all and only the dimensions of the hierarchies in H and such that $M_c = M$, is called a* base cube*, the others are called* derived cubes*. Let c_0 be a base cube over $\mathscr{C}$. The result of applying a query q to c_0 is a derived cube c such that (i) $G_c = G_q$, (ii) $M_c = M_q$, and (iii) ω_c assigns to each coordinate $\chi \in c$ satisfying the conjunction of the predicates in P_q and to each measure $m \in M_c$ the value computed by applying $op(m)$ to the values of m for all the coordinates of c_0 that roll-up to χ.*

In OLAP terms, a derived cube is the result of either a roll-up, a slice-and-dice, or a projection made over a base cube.

Chapter 9

Augmented OLAP

9.1 Introduction

With the disruptive advances in pervasive computing and industry 4.0, BI is shifting its focus on the integration of (internal) enterprise and (external) contextual data. In this direction, the synergy of analytical frameworks and augmented reality opens the door to *situated analytics*[1], in which users within a physical environment are provided with immersive analyses of local contextual data. Indeed, *augmented reality* (AR), a variation of virtual reality, allows users to superimpose digital information upon real-world objects [19], thus determining an augmented environment. Nowadays digital data returned to users are typically *operational*, meaning that they either describe the current state of the visualized objects (e.g., the temperature of a machine) or suggest the operations to be carried out (e.g., instructions to use the machinery). Conversely, limited attention has been devoted to providing *analytical* reports that can be used by decision makers to evaluate the behavior and performance of the visualized objects from a tactical and strategic point of view, for instance with reference to their past history or to other objects of the same type.

This new goal opens relevant research challenges and revamps many issues related to business intelligence and recommendation systems [56]. Indeed, when working with high-dimensional contextual data (the multidimensional nature of the context is well understood [6, 257]), identifying insightful queries and visualizations is not trivial [142] and requires several research issues to be solved [77]: How can data be sensed and accessed in real time? How is the sensed context mapped to enterprise data? Which data is salient to the user analysis? How do users interact with the retrieved data?

[1] Situated analytics has been defined as spatially-organized data representations attached to relevant entities for the purpose of understanding and decision-making [77]

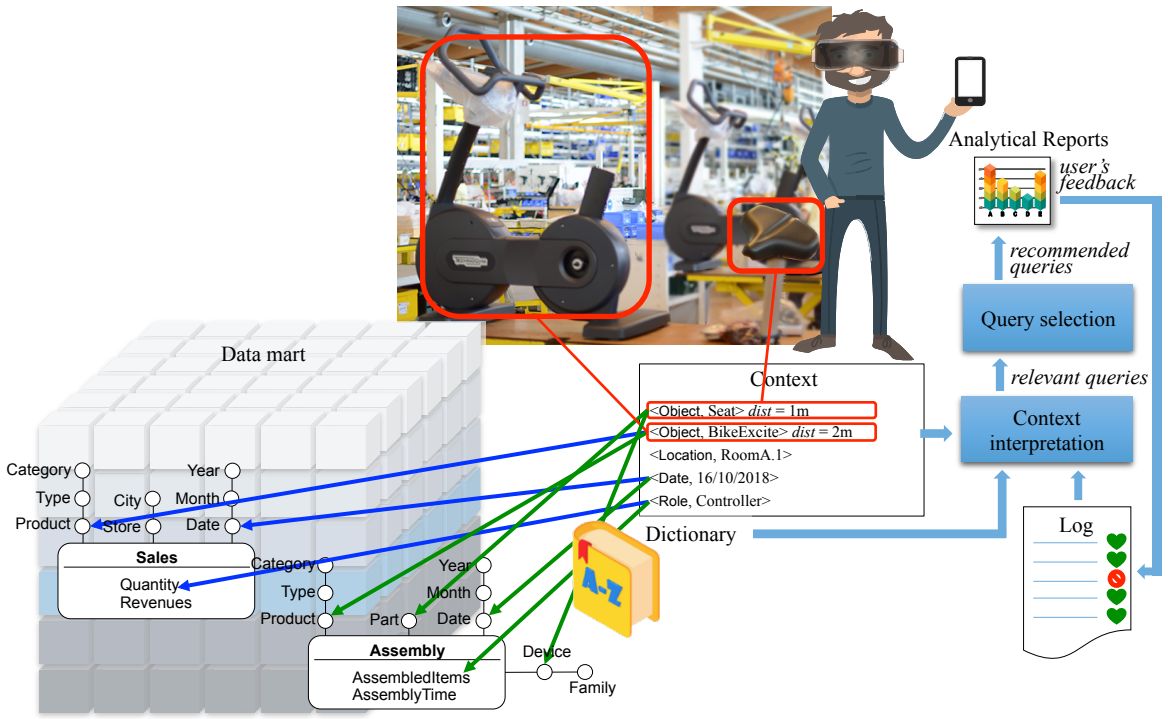


Figure 9.1: Overview of the AO framework

To the best of our knowledge, none of the context-aware recommender systems proposed in the literature addresses the above questions with reference to situated analytics in general, and to AR in particular. While the research on computer vision [35, 260] and situated visualization [80, 36, 140] is vivid, not much has been done to set up a business intelligence process bridging context sensing, data visualization, and decision making.

In this chapter, we envision and formalize a foundation for *Augmented OLAP* (AO), a framework empowering AR users with context-aware analytical information under visualization and time constraints. The context is modeled as a set of recognizable environment objects (e.g., a machine in a manufacturing environment) plus a set of additional user/environmental information (e.g., the user role and the room temperature). The analytical information returned is tailored to the context currently perceived by the user and comes in the form of reports obtained by running OLAP queries on the enterprise multidimensional cubes. The quantity of data returned must be limited in size and focused on the context to meet performance constraints and be easily interpretable by the user; furthermore, the intrinsic dynamics of AR applications ask for right-time (a few seconds) responsiveness of AO.

An overview of the AO framework is given in Figure 9.1 which shows Edgar, a controller working for a company producing fitness equipment and wearing AR smart devices featuring sensors of different types. We assume the pattern recognition capabilities necessary to

recognize the context are provided by these smart devices. Edgar's task is to optimize the production based on the assembling times of manufacturing devices, also considering the sale volumes of the different products. When he stares at an assembly machine, the AR glasses he is wearing recognize the machinery and some nearby objects (in the picture, a seat being assembled with an exercise bike); a context is generated accordingly, also including data about the current date and time and Edgar's position and role.

Our goal in this setting is to suggest to Edgar in real time the set of analytical queries over the enterprise multidimensional cubes that are more relevant according to both a-priori knowledge and feedback given by users in similar contexts. Relevant figures could be the number of produced items, the assembly times, and the revenues for the product being sensed. With reference to Figure 9.1, this task is carried out by the *Context interpretation* component using a collaborative filtering approach that relies on the query log. Although AO supports the possibility of learning meaningful queries from the log, its capability of returning the right information primarily comes from some a-priori knowledge provided by domain experts. This choice is not simply a solution to the well-known cold-start problem (i.e., the problem of providing significant recommendations when user feedbacks are still insufficient); it is rather a design choice aimed at enabling the system to give a useful answer in complex context scenarios, where learning from the log would require too many examples. The a-priori knowledge is modeled through a set of mappings between the potentially recognizable environment objects (stored in a *dictionary*) and the multidimensional elements of the enterprise cubes. Rather than proposing the most relevant query only, AO proposes a set of alternative queries to Edgar; all of them are related to the current context but they are different enough to offer to the user different flavors of the same information. This phase is implemented by the *Query selection* component. At this time, Edgar can either execute one of the proposed queries or express a new query to obtain a different report. Finally, Edgar gives his feedback on the proposed queries, which is stored in the log.

AO can be applied to different application domains ranging from healthcare [58] to factories [264, 39]; for this reason, the main modeling choices underlying our approach (e.g., how to define the relevance of an object in a context) have been formalized in a domain-independent fashion, while domain-dependent examples are provided in the context of AR in a factory where fitness equipment is produced.

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

1. We envision an AO framework, its functional architecture, and the user/system interaction process.

2. We explain how a-priori expert knowledge can be modeled by mapping context objects to relevant multidimensional elements.

Table 9.1: Comparison of recommender systems in terms of user input, context type, recommendation of multidimensional OLAP queries (MD), real-time constraint (RT), cardinality constraint (C), and query diversification (D)

	User input	**Context**	**MD**	**RT**	**C**	**D**
[145]	OLAP query	User profile	✓			
[13]	OLAP session	OLAP session log	✓	✓		
[154]	SQL query	Query logs		✓		✓
[268]	OLAP query	Dashboards, reports	✓			
[177]	SPARQL query	Web documents				
[274]	SQL query	Database statistics		✓		✓
[68]	SQL query	Result feedback			✓	
[121]	SQL query	Result statistics				
[57]	SQL query	Result statistics		✓		✓
[291]	Web query	Clicks, query log				
[146]	Web query	Clicks, query log				✓
[228]	Web query	Location		✓		✓
[46]	Web query	Query logs				
[105]	OLAP query	Query logs	✓			
AO	none	Physical env., log	✓	✓	✓	✓

3. We describe an efficient algorithm to generate relevant and diverse queries to be returned to the user.

4. We propose a collaborative filtering approach to let the system learn from user feedback.

The remainder of the chapter is organized as follows. Section 9.2 describes the related works in the field of context-aware recommendation systems. Section Section 9.3 formalizes the AO framework. Sections 9.4 and 9.5 describe the context interpretation and query selection components, respectively. Section 9.6 describes the results of experimental tests that measure the performance of AO. Finally, Section 9.7 sums up our contribution and gives future research directions.

9.2 Related works

The AO framework can be classified as a *recommender system* in the area of *business intelligence* based on a context made of *augmented entities*. Despite the huge amount of work in these areas, to the best of our knowledge, no approach lies at their intersection.

Over the years, scholars have highlighted the importance of exploiting contextual information to provide focused recommendations with the nature of contexts being quite heterogeneous (a summarized description in provided in Table 9.1), for instance space and

time [228], query logs [154, 268], statistics on results [121, 57] or databases [274], user interests [145], and social data [177]. Given such heterogeneity, other contributions (e.g., [286, 288, 177]) address the integration of contextual data to provide a common ground (e.g., a global schema [288] or an application programming interface [177]) enabling recommendation from multiple data sources. The previous context types have been widely adopted in several applications where the recommendation process is activated by an explicit user-defined input statement (e.g., query or keywords). Examples of applications are web query categorization [46, 13], recommendation [145, 291], and diversification [146]; query completion [154, 268]; localized web keywords suggestion [228]; and interactive exploration of databases [68]. The main differences between AO and these works are: (1) the multidimensional nature of the data handled and returned, (2) the nature of the context as well as the type of user/system interaction that triggers the recommendation, and the presence of (3) real-time, (4) cardinality and (5) diversification constraints. As to (1), multidimensional and hierarchical data support recommendations at different granularities, which intrinsically makes finding the best recommendation more complex; as to (2), physical contexts require ad-hoc solutions to choose the relevant context elements due to application specificities (e.g., object engagement) and to the possibility of having in the context elements that are perceived but that are not relevant for the user; as to (3), (4), (5), these constraints are required by immersive applications [77].

Recommender systems in business intelligence applications are well surveyed in [191]. Recommendations typically involve multidimensional queries [105] or sessions [13] (i.e., query sequences) using query logs as contexts. These approaches are based on collaborative filtering techniques that do not synthesize new queries from existing ones, but pick queries from the log depending on their similarity score. Conversely, AO allows the generation of queries not already present in the log by combining similar queries from the log and contextual information into a set of diverse queries. This assumption collocates AO as a hybrid approach to recommendation [191], differentiating AO from the above-mentioned contributions in multidimensional recommendation systems. Note that diversification and multiple recommendations are used to better meet user interests [309]. A further advantage of AO over pure collaborative filtering approaches is that AO does not suffer from the so-called cold start problem, since it is able to return an appropriate recommendation even when the log is empty [209].

In the area of AR and situated analytics, contexts play an even more central role. There, a context is the set of objects recognized in the environment that acts as situated stimulus (i.e., object properties) to be translated into inputs for a search query; it is augmented with virtual information and is returned to the user [36]. Physical environment becomes a source

of contextual information in [11], where user interaction with a physical environment is leveraged to retrieve operational data of interest. The usage of AR as an interactive medium opens to a natural data exploration and is especially helpful when the analysis goals are not specified [140]. Scholars focus on finding proper visualization to embed operational data in physical objects [80, 36, 140], with a particular effort on the implementation of toolkit allowing the rapid prototyping of such visualizations [253]. Although [80] considers multidimensional data, it is not specified how the process of information retrieval and analysis of data at multiple level of details is carried out. Besides, these approaches do not include collaborative filtering to discover potentially useful information. Interestingly, although [80] does not consider analytical data, it introduces a mantra for situated analytics: "details first, analysis, then context-on-demand" which contradicts the well-known mantra "overview first, zoom and filter, then details-on-demand" [252] of classical visualization systems. Indeed, when it comes to pure augmented visualizations, information is directly attached to single objects [80, 36], assigning higher priority to details than to generic information.

9.3 Preliminaries

Given the formal background introduced in Section 8.3, we introducing a formal setting to manipulate multidimensional data. For simplicity, we consider

- linear hierarchies only; and
- additionally to Definition 15, a query q is well-formed if (i) all predicates in P_q are *external*, i.e., they are expressed on levels that are less or equal to a level in G_q in the roll-up order [232]; and (ii) P_q is a set of Boolean *equality* clauses defined over members of levels of H.

Example 6 (Data Mart) *Our working example (Figures 9.1 and 9.2) includes two cubes,* Sales *and* Assembly*, which share hierarchies* Product *and* Date*; the two cubes are completed by hierarchies* Store *and* Device*, and* Part*. Sales are described by measures* Quantity *and* Revenues*, while Assembly is described by measures* AssembledItems *and* AssemblyTime*; all measures are additive, i.e., they are coupled with the* `sum` *operator. A member of the* Part *level is* Seat*, while a member of* Product *is* BikeExcite. □

In the following, we will often need to denote the md-elements to which query $q = \langle G_q, P_q, M_q \rangle$ *refers*; specifically,

- if $l \in G_q$, then q refers to level l;

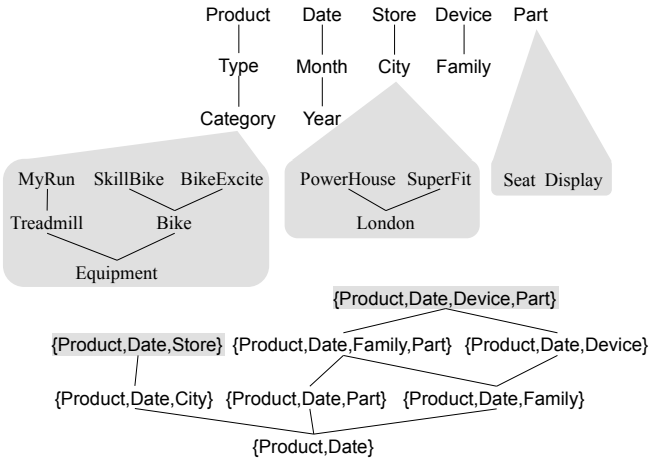


Figure 9.2: Roll-up total orders (top), part-of partial orders (middle), and an excerpt of the group-by set partial order (bottom) for our working example; the group-by sets of the Sales and Assembly cubes are in gray

- if $m \in M_q$, then q refers to measure m;
- if $(l = u) \in P_q$, then q refers to member u and to level l, being l the level whose domain includes u.

The set of md-elements to which q refers will be denoted with $ref(q)$.

Example 7 (Query) *With reference to the* Sales *cube, the query asking for quantity of sold products for each month of 2019 and each product type is* $q = \langle G_q, P_q, M_q \rangle$*, with*

$$\begin{aligned} G_q &= \{\mathsf{Month}, \mathsf{Type}\} \\ P_q &= \{(\mathsf{Year} = 2019)\} \\ M_q &= \{\mathsf{Quantity}\} \end{aligned}$$

The SQL formulation of q on a star schema featuring fact table `FT_Sales` *and dimension tables* `DT_Date` *and* `DT_Product` *would be*

```
SELECT DT_Date.Month, DT_Product.Type, sum(FT_Sales.Quantity)
FROM FT_Sales
JOIN DT_Date ON (FT_Sales.DateId = DT_Date.DateId)
JOIN DT_Product ON (FT_Sales.ProductId = DT_Product.ProductId)
WHERE DT_Date.Year = 2019
GROUP BY DT_Date.Month, DT_Product.Type
```

For this query it is

$$ref(q) = \{\mathsf{Month}, \mathsf{Type}, \mathsf{Year}, \mathsf{Quantity}, 2019\}$$

By adding measure AssembledItems*, from the* Assembly *cube, to M_q we get an example of a drill-across query whose SQL formulation is*

```
SELECT DT_Date.Month, DT_Product.Type,
      sum(FT_Sales.Quantity), sum(FT_Assembly.AssembledItems)
FROM FT_Sales
JOIN DT_Date ON (FT_Sales.DateId = DT_Date.DateId)
JOIN DT_Product ON (FT_Sales.ProductId = DT_Product.ProductId)
JOIN FT_Assembly ON (FT_Assembly.DateId = DT_Date.DateId AND
      FT_Assembly.ProductId = DT_Product.ProductId)
WHERE DT_Date.Year = 2019
GROUP BY DT_Date.Month, DT_Product.Type
```

□

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

Definition 17 (Dictionary) *A* dictionary *$\mathscr{D}$ is a set of classes, each coupled with a domain of values. Each pair $d = \langle class, value \rangle$ such that value belongs to the domain of class is called an* entry *of $\mathscr{D}$.*

Note that the dictionary can describe the sensed environment at different levels of precision. For example, if the smart device that perceives the environment successfully identifies a bike, but is not capable of labeling the specific bike model, it will return the $\langle \mathsf{Object}, \mathsf{Bike} \rangle$ entry. On the other hand, if the specific product BikeExcite is recognized, the smart device will return the entry $\langle \mathsf{Object}, \mathsf{BikeExcite} \rangle$.

The power of the AO framework comes from the ability to bind the perceived entries to the cube md-elements. This capability is rooted in a-priori knowledge that specifies which

md-elements can be interesting for the user when a given dictionary entry is perceived. This knowledge is defined through a dictionary-to-cube mapping established by a domain expert at setup time. To enhance the expressiveness of our framework, we consider that some md-elements may be interesting non *per se* but only when associated with other md-elements, so we map entries not on simple sets of md-elements but on sets of *fragments*, each fragment being a set of md-elements that should appear all together in queries.

Definition 18 (Mapping) *A* mapping *from dictionary $\mathscr{D}$ to data mart DM is a multivalued function μ that maps an entry d to a set of fragments, i.e., sets of md-elements of the cubes in DM. Each fragment $f \in \mu(d)$ has a mapping weight, $w_{map}(d,f) \in (0,1]$.*

The mapping function is multivalued since many fragments of md-elements may be of interest for each dictionary entry; this typically happens for hierarchy levels, which can be all interesting —even if with different values of w. For example, when some device is perceived, besides showing data for that specific device, also showing aggregated data for the device type could be interesting. Through mapping weights, domain experts give an a-priori quantification of the interest of each fragment for analyses when a given entry is part of the context.

Although a discussion about how the dictionary is created and the mappings are established is out of the scope of this chapter, we remark that this does not necessarily have to be done manually for all the cube members, which would be tedious for attributes with large domains, but it can be largely automated. For instance, to reduce the user's effort in populating the dictionary, an approach like the one proposed in [178] could be used. There, continual learning is applied to classify known objects and to learn objects of never-seen classes. Once sensed and marked as relevant, novel objects can be easily learned and added to the dictionary. Giving novel objects names equal or similar to names of md-elements ensures that a set of basic mappings from the dictionary to the data mart can be automatically created, to be possibly fine-tuned later by a domain expert. Alternatively, tentative mappings could be established by providing universally-quantified rules such as $\mu(\langle \mathsf{Object}, value \rangle) = \{\{value\}\}$ for each value that corresponds to a member of some level in a hierarchy.

Example 8 (Dictionary and Mappings) *The dictionary for our example includes, among the others, entries related to products (e.g., $\langle \mathsf{Object}, \mathsf{BikeExcite} \rangle$), product parts (e.g., $\langle \mathsf{Object}, \mathsf{Seat} \rangle$), and user roles (e.g., $\langle \mathsf{Role}, \mathsf{Controller} \rangle$). An excerpt of the mapping from this*

dictionary to the cubes of Example 6 may look like this (Figure 9.1):

$$\begin{aligned}\mu(\langle \mathsf{Object}, \mathsf{Seat}\rangle) &= \{\{\mathsf{Seat}\}, \{\mathsf{Device}\}\},\\ \mu(\langle \mathsf{Object}, \mathsf{BikeExcite}\rangle) &= \{\{\mathsf{BikeExcite}\}\},\\ \mu(\langle \mathsf{Role}, \mathsf{Controller}\rangle) &= \{\{\mathsf{AssembledItems}, \mathsf{Quantity}\}\},\\ \mu(\langle \mathsf{Date}, 16/10/2018\rangle) &= \{\{\mathsf{Date}\}\}\end{aligned}$$

The first line returns two fragments including a member and a level respectively; it states that, when the user senses a seat, she may be interested in analyzing either the part to be assembled or the data concerning the assembly device. The second line returns one fragment including a member; it states that, when the user senses a product, it is normally interested in analyzing the data for that product. The third line returns one fragment including two measures; it states that controllers are interested in comparing the number of assembled items with the quantity sold. The fourth line returns one fragment including a level; it states that users are normally interested in daily data. Finally, examples of mapping weights are

$$\begin{aligned}w_{map}(\langle \mathsf{Object}, \mathrm{BikeExcite}\rangle, \{\mathrm{BikeExcite}\}) &= 0.5\\ w_{map}(\langle \mathsf{Role}, \mathrm{Controller}\rangle, \{\mathsf{AssembledItems}, \mathsf{Quantity}\}) &= 1\end{aligned}$$

□

9.4 Context interpretation

In this section, we show how, given a set of perceived objects, AO produces a set of *relevant queries*, i.e., queries whose results may be of interest to the user.

9.4.1 Take the context...

The AO starting point is the *context*, i.e., a set of dictionary entries corresponding to the currently perceived environment objects. More formally:

Definition 19 (Context) *A* context *T over dictionary $\mathscr{D}$ is a set of entries of $\mathscr{D}$; each entry $d \in T$ is coupled with a* context weight $w_{cnt}(T,d) \in (0,1]$.

The value of the weight for each entry may depend on different factors, depending on the application domain. Non-perceived entries (i.e., for which it would be $w_{cnt}(T,d) = 0$) are not included in the context. In our case study we assume that a subset of entries are *engaged*,

meaning that they have explicitly been indicated by the user as being part of her current focus of interest [260]; for these entries, the weight is always 1. For the other entries, the weight is inversely proportional to the distance between the user and the specific object being observed.

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

Definition 20 (Image) *Given context T over dictionary $\mathscr{D}$ and mapping μ from $\mathscr{D}$ to data mart DM, the* image *of T through μ is the set of fragments that are mapped through μ from the entries in T:*

$$I_\mu(T) = \bigcup_{d \in T} \mu(d)$$

Example 9 (Context) *A possible context is the one depicted in Figure 9.1, where Edgar is inspecting the assembly of fitness equipment in Room A.1 on October 16th 2018:*

$$\begin{aligned} T = \{&\langle \mathsf{Object}, \mathrm{Seat} \rangle, \\ &\langle \mathsf{Object}, \mathrm{BikeExcite} \rangle, \\ &\langle \mathsf{Role}, \mathrm{Controller} \rangle, \\ &\langle \mathsf{Location}, \mathrm{RoomA.1} \rangle, \\ &\langle \mathsf{Date}, 16/10/2018 \rangle\} \end{aligned}$$

The BikeExcite product is engaged. Possible context weights are

$$\begin{aligned} w_{cnt}(T, \langle \mathsf{Object}, \mathrm{Seat} \rangle) &= 0.6, \\ w_{cnt}(T, \langle \mathsf{Object}, \mathrm{BikeExcite} \rangle) &= 1, \\ w_{cnt}(T, \langle \mathsf{Role}, \mathrm{Controller} \rangle) &= 1, \\ w_{cnt}(T, \langle \mathsf{Location}, \mathrm{RoomA.1} \rangle) &= 0.6, \\ w_{cnt}(T, \langle \mathsf{Date}, 16/10/2018 \rangle) &= 0.6 \end{aligned}$$

The image of T through the mapping μ described in Example 8 is

$$I_\mu(T) = \{\{\mathrm{Seat}\}, \{\mathsf{Device}\}, \{\mathrm{BikeExcite}\}, \{\mathsf{AssembledItems}, \mathsf{Quantity}\}, \{\mathsf{Date}\}\}$$

□

The image includes the set of fragments relevant to a context according to the mapping, but it does not specify how they will be used to generate the queries to be proposed to the

user when that context is sensed. Indeed, given an image, several queries can be generated, each including a subset of the fragments in the image.

9.4.2 ... add the log...

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

Definition 21 (Log) *A log L is a multiset of triples $\langle T, q, ok \rangle$ where T is a context, q is a query, and ok (feedback) is 1 if the user accepted the query, -1 if she rejected the query.*

Example 10 (Log) *A possible log for our working example is $L = (\langle T, q_1, -1 \rangle, \langle T, q_2, 1 \rangle)$, where*

$$\begin{aligned} q_1 = \langle & \{\mathsf{Date}, \mathsf{Part}, \mathsf{Product}\}, \\ & \{(\mathsf{Product} = \mathrm{BikeExcite})\}, \\ & \{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle \\ q_2 = \langle & \{\mathsf{Month}, \mathsf{Part}, \mathsf{Product}\}, \\ & \{(\mathsf{Product} = \mathrm{BikeExcite})\}, \\ & \{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle \end{aligned}$$

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

A log entry related to context T' should impact the recommendations related to the current context T only if the two contexts are similar, since it is reasonable to assume that the user will have similar behaviors in similar contexts.

Definition 22 (Context Similarity) *Given two contexts T, T' over dictionary $\mathscr{D}$, we define the* similarity *between T and T' as their Jaccard index:*

$$sim(T, T') = \frac{|T \cap T'|}{|T \cup T'|}$$

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

$$L_T = \{\langle T', q, ok\rangle \in L \text{ s.t. } sim(T, T') \geq \varepsilon\}$$

where ε is a similarity threshold. Then, $I_\mu(T)$ is extended by adding, for each query q in L_T, one fragment corresponding to the set of md-elements referred to by q:

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

In this way, $I^*_\mu(T)$ includes all the fragments that are relevant to context T *either according to the mapping or to the previous user experience*.

We now define the *log relevance* to T of fragment $f \in I^*_\mu(T)$ as the weighted number of times f has been accepted by the user (i.e., $ok = 1$) over the number of times it has been proposed (i.e., $ok = *$); weighting is based on the similarity between the current context T and the considered log context T':

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

where $L_T(f) = \{\langle T', q, ok\rangle \in L_T \text{ s.t. } f \sqsubseteq ref(q)\}$ and $\sqsubseteq$ is a hierarchy-aware containment relationship between sets of md-elements:

$$\begin{aligned} f \sqsubseteq ref(q) \Leftrightarrow\ & (\forall m \in f \text{ s.t. } m \text{ is a measure}, m \in ref(q)) \wedge \\ & (\forall u \in f \text{ s.t. } u \text{ is a member}, \exists u' \in ref(q) \text{ s.t. } u' \geq_h u) \wedge \\ & (\forall l \in f \text{ s.t. } l \text{ is a level}, \exists l' \in ref(q) \text{ s.t. } l' \succeq_h l) \end{aligned}$$

To avoid relevance to be 0 when f has never been accepted, a Laplace smoothing is applied in the formula above. Noticeably, the impact of Laplace smoothing decreases as the cardinality of $L_T(f)$ increases, that is, the weight tends to 0 if several queries referring f have been proposed but never accepted by the user. Conversely, it tends to $\frac{1}{2}$ if only a few queries referring f have been proposed.

It is now possible to define the *relevance* to T of each fragment $f \in I^*_\mu(T)$ by taking into account, for each context entry d that maps to f, not only the entry weight $w_{cnt}(T, d)$ and the mapping weight $w_{map}(d, f)$, but also the log relevance $\rho_T(L, f)$:

$$rel_T(f) = \rho_T(L, f) \cdot \left(\sum_{d \in T \text{ s.t. } f \in \mu(d)} w_{cnt}(T, d) \cdot w_{map}(d, f) \right)$$

where $w_{cnt}(T,d) = w_{map}(d,f) = \frac{1}{2}$ for all $f \in I^*_\mu(T) \setminus I_\mu(T)$. Clearly, the reason for providing a default value for all fragments present in the extended image but not deriving from the context is to avoid the corresponding contribution to the relevance to be null; choosing the default value of $\frac{1}{2}$ is in line with the Laplace smoothing applied to the log relevance.

Example 11 (Extended image) *With reference to the image $I_\mu(T)$ from Example 9 and to the log entries in Example 10, the extended image is*

$$\begin{aligned} I^*_\mu(T) = \{&\{\mathsf{Seat}\}, \{\mathsf{Device}\}, \{\mathsf{BikeExcite}\}, \{\mathsf{AssembledItems}, \mathsf{Quantity}\}, \{\mathsf{Date}\}, \\ &\{\mathsf{Month}, \mathsf{Part}, \mathsf{Product}, \mathsf{BikeExcite}, \mathsf{AssembledItems}, \mathsf{AssemblyTime}\}, \\ &\{\mathsf{Date}, \mathsf{Part}, \mathsf{Product}, \mathsf{BikeExcite}, \mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\} \end{aligned}$$

The last two fragments have been added to $I_\mu(T)$ as they corresponds to queries drawn from contexts similar to T. As to $\rho_T(L,f)$ and $rel_T(f)$ it is

$$\begin{aligned} \rho_T(L, \{\mathsf{Date}, \mathsf{Part}, \mathsf{Product}, \mathsf{BikeExcite}, \mathsf{AssembledItems}, \mathsf{AssemblyTime}\}) &= 0.33 \\ \rho_T(L, \{\mathsf{AssembledItems}, \mathsf{Quantity}\}) &= 0.5 \\ rel_T(\{\mathsf{Date}, \mathsf{Part}, \mathsf{Product}, \mathsf{BikeExcite}, \mathsf{AssembledItems}, \mathsf{AssemblyTime}\}) &= 0.16 \\ rel_T(\{\mathsf{AssembledItems}, \mathsf{Quantity}\}) &= 0.5 \end{aligned}$$

□

We close this section by observing that the log size will quickly increase with time and spending a few words about how the log can be curated. Indeed, in [15] it is highlighted that many recommended queries can become irrelevant (e.g., in case of sensible context variations such as room refurnishment) or non-computable (e.g., due to changes in the multidimensional schema). A basic way to deal with memory limitations when storing large logs would be to provide a log cache, with size limits, where only the latest entries are cached per user. A more sophisticated way would be to adopt an *indicator of obsolescence* as in [15] to decide whether to prune obsolescent log entries.

9.4.3 ... get the queries

As already stated, the context interpretation component is in charge of generating a set Q_T of queries relevant to context T. In principle, the query that includes *all* the md-elements in the fragments of the extended image of T might be directly proposed to the user. Unfortunately, when several objects are sensed in the environment and the context includes a large number

of entries, such queries would be *monster queries*, i.e., quite complex queries with very high cardinalities. Monster queries are particularly undesirable in AR applications since:

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

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

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

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

where $card(G_q)$ is the expected cardinality of a query with group-by set G_q and no selection predicates, and $sel(p)$ is the selectivity of each simple predicate p belonging to P_q. Note that we can safely use this formula to estimate $card(q)$ because, as a consequence of the way we create queries in our approach, all predicates in P_q are always *external*, i.e., they are expressed on levels that are less or equal to a level in G_q [232] in the roll-up partial order. As to $card(G_q)$, it must be computed taking the cube sparsity into account, considering that the sparsity differs from cube to cube. In the simple case in which all measures in M_q belong to a single cube c_i, it can be estimated for instance using the Cardenas formula as shown in [247, 115]:

$$card(G_q) = card_{c_i}(G_q) = \Phi(card_{max}(G_q), |c_i|)$$

where $|c_i|$ is the cardinality of c_i and $card_{max}(G_q)$ is the maximum cardinality (i.e., if there were no sparsity) of group-by set G_q:

$$card_{max}(G_q) = \prod_{l \in G_q} |Dom(l)|$$

If q is a drill-across query,[2] the measures in M_q are scattered across two or more cubes $c_1, \ldots, c_r$; in this case the sparsities of these cubes can be assumed to be mutually independent:

$$card(G_q) = card_{max}(G_q) \cdot \prod_{i=1}^{r} card_{c_i}(G_q)/card_{max}(G_q)$$

where $card_{c_i}(G_q)/card_{max}(G_q)$ expresses the probability that a given tuple of members is present in c_i, thus the product expresses the probability of that tuple to be present in *all* the involved cubes. Note that other approaches have been devised for a more precise cardinality estimation in presence of selection predicates on multiple attributes, for instance [224], which uses singular value decomposition, and [123], based on histograms.

Example 12 (Cardinality) *Given cubes* Assembly *and* Sales*, let* $|\mathsf{Assembly}| = 200000$*,* $|\mathsf{Sales}| = 400000$*,* $|Dom(\mathsf{Product})| = 100$*, and* $|Dom(\mathsf{Date})| = 1000$*. Consider query*

$$\begin{aligned} q = \langle & \{\mathsf{Date}, \mathsf{Product}\}, \\ & \{(\mathsf{Year} = 2019)\}, \\ & \{\mathsf{AssembledItems}, \mathsf{Quantity}\}\rangle \end{aligned}$$

By applying the formulas above we get

$$\begin{aligned} card_{max}(\{\mathsf{Date}, \mathsf{Product}\}) &= 100000 \\ card_{\mathsf{Assembly}}(\{\mathsf{Date}, \mathsf{Product}\}) &= 86467 \\ card_{\mathsf{Sales}}(\{\mathsf{Date}, \mathsf{Product}\}) &= 98169 \\ card(\{\mathsf{Date}, \mathsf{Product}\}) &= 84884 \end{aligned}$$

Assuming that $sel(\mathsf{Year} = 2019) = 0.25$*, we get* $card(q) = 21221$. □

As to (ii), i.e., considering only the most relevant fragments in the image, before we proceed we remove from the extended image $I^*_\mu(T)$ all the fragments f whose relevance $rel_T(f)$ is below a given threshold η, being the relevance defined as follows.

Definition 23 (Relevant Queries and Query Relevance) *Given context* T*, query* q *is said to be* relevant to T *if (i) it is* $f \subseteq ref(q)$ *for at least one fragment* f *in the extended image of* T*, (ii)* q *is well-formed, and (iii)* $card(q) \leq \gamma$*. The set of relevant queries to* T *is denoted*

[2] We recall from Section 9.3 that drill-across queries can be formulated because all cubes in the data mart share a set of conformed dimensions.

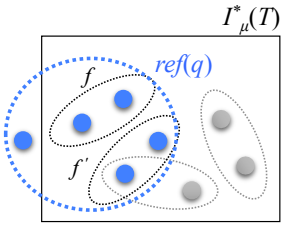


Figure 9.3: The relevance of q (in blu its md-elements) is the sum of the relevances of fragments f and f'

with Q_T. The relevance *to T of query $q \in Q_T$ in presence of log L is defined as*

$$rel_T(q) = \sum_{f \in I^*_\mu(T) \text{ s.t. } f \subseteq ref(q)} rel_T(f)$$

For instance, with reference to Figure 9.3, the relevance of q is estimated by summing up the relevances of fragments f and f' as these are the only fragments of $I_\mu(T)$ completely contained in $ref(q)$.

Example 13 (Query relevance) *Given the mapping weights in Example 8, the context weights in Example 9, the log relevance in Example 10, and the extended image in Example 11, the relevance of*

$$\begin{aligned} q = \langle &\{\mathsf{Month}, \mathsf{Part}, \mathsf{Product}\}, \\ &\{(\mathsf{Product} = \mathrm{BikeExcite})\}, \\ &\{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle \end{aligned}$$

is $rel_T(q) = 0.4$. The fragments contributing to the query relevance (i.e., those contained in $ref(q)$) are $\{\mathrm{BikeExcite}\}$ *and* $\{\mathsf{Month}, \mathsf{Part}, \mathsf{Product}, \mathrm{BikeExcite}, \mathsf{AssembledItems}, \mathsf{AssemblyTime}\}$. □

In the remainder of this section we explain how we create the set Q_T of relevant queries, as introduced in Definition 23, to be handed to the query selection component. Basically, in Algorithm 3 we follow a depth-first enumeration approach [219] to generate all possible combinations of the fragments in $I^*_\mu(T)$. This is done by calling the recursive procedure *Expand*, whose pseudocode is shown in Algorithm 4, for each fragment available. Within *Expand*, function $Gen(f)$ (Line 1) returns a query q using the md-elements in fragment f as follows:

- G_q includes all the levels in f plus the levels of all the members in f;

Algorithm 3 Generation of relevant queries

INPUT $I^*_\mu(T)$: extended image, γ: cardinality threshold
OUTPUT Q_T: set of relevant queries

1: $Q_T \leftarrow \varnothing$ ▷ Result set
2: $F \leftarrow I^*_\mu(T)$ ▷ Fragment set
3: **for each** $f \in F$ **do** ▷ For each fragment...
4: $F \leftarrow F \setminus \{f\}$
5: $Expand(F, f)$ ▷ ...generate relevant queries from f and add them to Q_T
6: **return** Q_T

Algorithm 4 Procedure $Expand(F, f)$

INPUT F: set of fragments, f: fragment to be used for generating queries

1: $q \leftarrow Gen(f)$ ▷ Generate a query from f
2: **if** $wellFormed(q) \wedge q \notin Q_T$ **then**
▷ If q is not well-formed and has already been generated, stop
3: **if** $card(q) \leq \gamma$ **then** ▷ If q has low cardinality...
4: $Q_T \leftarrow Q_T \cup \{q\}$ ▷ ...add it to Q_T
5: **for each** $f' \in F$ **do** ▷ For each other fragment in F...
6: $F \leftarrow F \setminus \{f'\}$
7: $Expand(F, f \cup f')$ ▷ ...add it to f and generate relevant queries

- P_q includes a selection predicate on each member in f;
- M_q includes all the measures in f.

To avoid redundancies in G_q and P_q, only levels and members at the finest granularity are kept for each hierarchy. If the query returned by $Gen(f)$ is not already present in Q_T, has low cardinality, and is well-formed, it is added to the result (Line 4). Then, recursion is triggered by calling *Expand* with the union of f and any other available fragment (Lines 5-7, parameter F is passed by value).

Remarkably, if q is not well-formed (Line 2), the current branch of recursion can safely be pruned. Indeed, a query is not well-formed when either (i) it has non-external predicates or (ii) it returns a measure that is not available at the required granularity (see Definition 15). As to (i), we note that $Gen(q)$ adds to the query group-by set the levels of all the members in f, so it cannot generate queries with non-external predicates. As to (ii), when proceeding with the recursion, new md-elements would be added to q; this may make the granularity of q finer but it cannot make it coarser. Thus, the queries obtained by adding further md-elements to a query q that is not well-formed can never be well-formed. The current branch can also be pruned if q has already been added to Q_T; in this case, since we are adopting a depth-first approach, the extension of q with further fragments has already been done as well. We finally

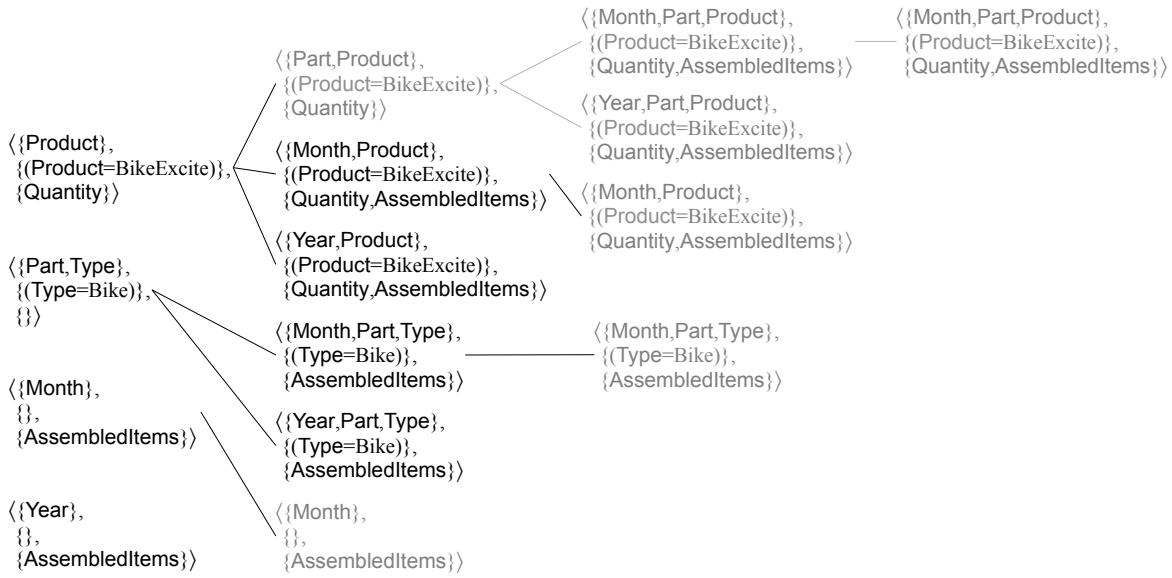


Figure 9.4: Depth-first query generation for Example 14 (gray arcs are safely pruned, so gray queries are not added to Q_T

note that cardinality cannot be used to prune the search space; indeed, by adding further md-elements to a high-cardinality query, we get a new query whose cardinality might be below the threshold γ.

Example 14 (Query generation) *Let*

$$I^*_\mu(T) = \{\{\mathsf{BikeExcite}, \mathsf{Quantity}\}, \{\mathsf{Part}, \mathsf{Bike}\}, \\ \{\mathsf{Month}, \mathsf{AssembledItems}\}, \{\mathsf{Year}, \mathsf{AssembledItems}\}\}$$

As sketched in Figure 9.4, Algorithm 3 works as follows. The first fragment picked by the for cycle of Line 3 is $\{\mathsf{BikeExcite}, \mathsf{Quantity}\}$*, which corresponds to query* $q = \langle\{\mathsf{Product}\}, \{(\mathsf{Product} = \mathsf{BikeExcite})\}, \{\mathsf{Quantity}\}\rangle$*. This query is well-formed and it has cardinality equal to 1, so it is added to* Q_T*. The depth-first exploration continues by picking fragment* $\{\mathsf{Part}, \mathsf{Bike}\}$ *(Algorithm 4, Line 5) and calling Expand on the union fragment* $\{\mathsf{BikeExcite}, \mathsf{Quantity}, \mathsf{Part}, \mathsf{Bike}\}$*. Since measure Quantity is not defined at the part granularity, the query obtained is not well-formed and this branch of recursion is pruned (Algorithm 4, Line 2). The next fragments picked by Line 5 of Algorithm 4 are* $\{\mathsf{Month}, \mathsf{AssembledItems}\}$ *and* $\{\mathsf{Year}, \mathsf{AssembledItems}\}$*, which are expanded as shown in Figure 9.4.*

Once the recursion started from $\{\mathsf{BikeExcite}, \mathsf{Quantity}\}$ *is terminated, the other three fragments are picked by Line 3 of Algorithm 3. In particular, when* $\{\mathsf{Month}, \mathsf{AssembledItems}\}$

is picked, the corresponding query is added to the result (assuming it has low cardinality). Now, the fragment is expanded with {Year, AssembledItems} *(Algorithm 4, Line 5), producing the union fragment* {Month, AssembledItems, Year}*; remarkably, since the corresponding query is already present in the result (*Year *is coarser than* Month*, so it is removed by Gen*(f)*), this branch is pruned.* □

9.5 Query selection

Context interpretation returns the set Q_T of relevant queries, whose results may be of interest to the user. This set is potentially exponential in the number of fragments; as we will show in Figure 9.12, discarding ill-formed and high-cardinality queries does not drastically reduce the cardinality of Q_T, which may easily turn out to be several thousands. Clearly, some selection has to be done to avoid flooding the user with tons of (probably very similar to each other) queries. Thus, the goal of the step discussed in this section is to select from Q_T a fixed number *rq* of queries to be recommended to the user. The guiding criterion is to select the subset of queries that are both maximally relevant to the context and diverse; this is done by defining an ad-hoc measure of *query set relevance* that takes both relevance and diversity into account.

To this end, we start by generalizing to sets of queries the definition of similarity given for query pairs in [14]. This definition combines three components: one related to group-by sets, one to selection predicates, and one to measure sets. Let $levBelow(G_q)$ denote the set of levels that are below a level of G_q in the roll-up order:

$$levBelow(G_q) = \{l' : l \succeq_h l', \text{ for } l \in G_q\}$$

and $memBelow(P_q)$ denote the set of members that are below a member of P_q in the part-of order:

$$memBelow(P_q) = \{u' : u \geq_h u', \text{ for } u \in P_q\}$$

Definition 24 (Query Similarity) *Given context T, let $Q \subseteq Q_T$. The* similarity *of the queries in Q is defined as*

$$sim(Q) = \alpha \cdot \sigma_{gbs}(Q) + \beta \cdot \sigma_{sel}(Q) + \gamma \cdot \sigma_{meas}(Q)$$

Table 9.2: Md-elements for computing quest similarity in Example 15; intersecting md-elements are in bold

	md-element	q'	q''	q'''
$\bigcup_{q\in Q} levBelow(G_q)$	Product	✓	✓	
	Type	✓	✓	✓
	Category	✓	✓	✓
	Month	✓	✓	
	Year	✓	✓	
$\bigcup_{q\in Q} memBelow(P_q)$	Bike	✓		
	Equipment	✓	✓	✓
$\bigcup_{q\in Q} M_q$	**AssembledItems**	✓	✓	✓
	AssemblyTime		✓	
	Quantity	✓		

where α, β, and γ are normalized to 1 and

$$\sigma_{gbs}(Q) = \frac{|\bigcap_{q\in Q} levBelow(G_q)|}{|\bigcup_{q\in Q} levBelow(G_q)|}$$
$$\sigma_{sel}(Q) = \frac{|\bigcap_{q\in Q} memBelow(P_q)|}{|\bigcup_{q\in Q} memBelow(P_q)|}$$
$$\sigma_{meas}(Q) = \frac{|\bigcap_{q\in Q} M_q|}{|\bigcup_{q\in Q} M_q|}$$

Like in [14], we choose $\alpha = 0.35$, $\beta = 0.5$, and $\gamma = 0.15$.

Example 15 (Query similarity) *Let $Q = \{q', q'', q'''\}$, with*

$$q' = \langle\{\mathsf{Product}, \mathsf{Month}\}, \{(\mathsf{Type} = \mathrm{Bike})\}, \{\mathsf{AssembledItems}, \mathsf{Quantity}\}\rangle$$
$$q'' = \langle\{\mathsf{Product}, \mathsf{Month}\}, \{(\mathsf{Category} = \mathrm{Equipment})\}, \{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle$$
$$q''' = \langle\{\mathsf{Type}\}, \{(\mathsf{Category} = \mathrm{Equipment})\}, \{\mathsf{AssembledItems}\}\rangle$$

Considering the roll-up and part-of orders in Figure 9.2 and the involved md-elements (see Table 9.2), it is $sim(Q) = 0.35 \cdot \frac{2}{5} + 0.5 \cdot \frac{1}{2} + 0.15 \cdot \frac{1}{3} = 0.44$. □

Based on the definition of query similarity, we can now define the relevance to the context of any set of queries. The global relevance of a set of queries cannot be properly computed as the sum of their relevances due to the intersections between their md-elements, thus we have to apply the well-known inclusion-exclusion principle [233]. The inclusion-exclusion

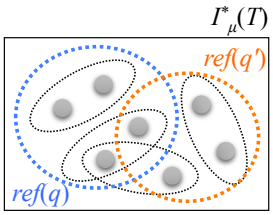


Figure 9.5: Estimation of query set relevance in Example 16

principle counts the number of distinct elements in the union of finite sets by summing up the cardinalities of the individual sets, subtracting the number of elements that appear in at least two sets, adding back the number of elements that appear in at least three sets, and so on. Similarly, to estimate the global relevance of a set of queries we sum the relevances of individual queries, subtract the average query relevance weighted by their similarity for any pair of queries, add back the average query relevance weighted by their similarity for any triple of queries, and so on.

Definition 25 (Query Set Relevance) *Given context T, let $Q \subseteq Q_T$. The* relevance *to T of Q is defined as*

$$rel_T(Q) = \sum_{\varnothing \neq Q' \subseteq Q} sim(Q') \cdot \frac{\sum_{q \in Q'} rel_T(q)}{|Q'|} \cdot (-1)^{|Q'|+1}$$

Example 16 (Query set relevance) *With reference to Figure 9.5, given $Q = \{q, q'\}$ such that $rel(q) = 0.8$, $rel(q') = 0.7$, and $sim(Q) = 0.2$, it is $rel_T(Q) = rel_T(q) + rel_T(q') - sim(q,q') \cdot (rel_T(q) + rel_T(q'))/2 = 1.35$.* □

Applying the inclusion-exclusion principle in Definition 25 ensures that, at a parity of relevances of the single queries, the more diverse these queries are, the higher the query set relevance. Thus, selecting from Q_T the subset R of rq queries with the maximum query set relevance implicitly allows AO to recommend queries that are both relevant to the context and diverse. More formally:

Problem 1 (Query Selection) *Given context T, select the subset R, $\varnothing \neq R \subseteq Q_T$, such that $rel_T(R) \geq rel_T(R')$ for each $\varnothing \neq R' \subseteq Q_T$.*

The query selection problem can be mapped to a *Weighted Maximum Coverage Problem* (WMCP) where each $q \in Q_T$ corresponds to a set of elements (i.e., fragments), each with its own weight (i.e., relevance). We want to find out the subset $R \subseteq Q_T$ with maximal weight and such that $|R| < rq$. The reduction in relevance due to query similarity is taken into account in

Algorithm 5 Query selection

INPUT Q_T: set of relevant queries, rq: number of queries to be recommended
OUTPUT R: recommended query set

1: $R \leftarrow \varnothing$ ▷ Result set
2: $Q \leftarrow Q_T$ ▷ Search space
3: **while** $(|R| < rq) \wedge (Q \neq \varnothing)$ **do** ▷ Still room in R and search space not empty
4: $q \leftarrow argmax_Q(rel_T(R \cup \{q\}))$ ▷ Pick the most promising query...
5: $Q \leftarrow Q \setminus \{q\}$ ▷ ... remove it from the search space
6: $R \leftarrow R \cup \{q\}$ ▷ ... and add it to the result
7: **return** R

the WMCP, which adds only once the weight of elements appearing in more than one $q \in R$. It is easy to verify that also every WMCP can be mapped to a query selection problem where (i) for each element a new fragment f is created, (ii) for each set of elements a new query $q \in Q_T$ is created, and (iii) $rel_T(f)$ is set to the weight of the element corresponding to f.

Since the two problems can be mapped to each other, they must have the same complexity. In [136] it is shown that the WMCP is NP-hard and that it can be faced with polynomial complexity by adopting a greedy algorithm that, at each iteration, picks the most promising element; so we adopt the greedy approach whose pseudocode is shown in Algorithm 5. Basically, at each iteration we pick from Q_T the query that, if added to the result R, maximizes the query set relevance rel_T of R (at the first iteration, this equals to picking the most relevant query). The algorithm is incremental, so queries can be recommended as soon as they are picked —without having to wait for the algorithm to terminate.

Example 17 (Query selection) *Let $Q_T = \{q', q'', q'''\}$ such that $rel(q') = 0.7$, $rel(q'') = 0.6$, $rel(q''') = 0.5$, $sim(q', q'') = 0.6$, $sim(q', q''') = 0.1$, and $sim(q'', q''') = 0.2$. After initialization, Algorithm 5 picks q' from Q_T (Line 4) as it has top query relevance so it also maximizes query set relevance. At the second iteration, q''' is picked at Line 4: although $rel(q'') > rel(q''')$, q'' is more similar to q' than q''', thus the query set relevance if q'' is added to R is lower ($rel_T(q', q'') = 0.91$, $rel_T(q', q''') = 1.14$). Assuming $rq = 2$, query selection stops here.* □

9.6 Experimental tests

In this section, we evaluate the AO framework in terms of (1) effectiveness, (2) efficiency, and (3) user satisfaction (i.e., to what extend the recommended queries meet the users' desiderata).

As to (1) and (2), we compare AO to our previous implementation A-BI [95]. AO extends A-BI mainly by (i) generalizing queries to operate on multiple cubes (*drill-across queries* in the OLAP terminology) to better fit real decision-making contexts; (ii) generalizing the object-to-cube mappings to map onto sets of multidimensional elements, so as to enhance their expressiveness; (iii) giving a new definition of query relevance and the corresponding formalization of the query selection problem as an optimization problem; and (iv) providing an extensive experimental evaluation based on a real manufacturing environment. Tests are carried out against a synthetic benchmark since we assume the problem of context generation to be addressed by the smart device and, to the best of our knowledge, no AR open dataset exists.

The user-system interaction works as follows: a session simulates a user walking through a factory of 10 rooms. While moving, she collects one view of each room (in a session each room is visited once). From each view, the smart device recognizes a set of objects belonging to the dictionary and lists them into a context. For each context, AO recommends a set R of queries to the user; she either chooses one of these queries (i.e., she gives a positive feedback for one of the recommended queries) or formulates an additional query that is slightly different from the ones proposed (i.e., she gives a negative feedback for all the queries and adds a new one). After some time, the user ends her exploration of the factory (i.e., her session). When a new user enters the factory (i.e., a new session begins), AO relies on the query log to recommend a new set of queries that better suit her interests. Since each session covers 10 rooms, after each session 10 contexts are added to the log together with the corresponding user feedback. The contexts related to each room may be slightly different, since the user could perceive the room from a different point of view, or the smart device could fail to recognize some of the objects.

This interaction is simulated by randomly generating 10 seed contexts, each corresponding to a different room. Seed contexts differ significantly from each other. Then, to simulate multiple visits of each room, small context variations are generated starting from each seed (i.e., different room perspectives). The number of objects recognized in each room (i.e., the context cardinality) ranges between 10 and 16. The test is repeated 10 times and the average behavior is considered.

We denote with s the number of sessions, i.e., the number of times each room has already been sensed. Besides, for each context, we call:

- q_{max} the query with maximal relevance in R. We recall that Algorithm 5 always recommends the set of rq well-formed queries with the highest relevance to the context.
- q_u the query formulated by the user. We denote with δ the similarity between the user query and the maximal query ($\delta = sim(\{q_u, q_{max}\})$, as of Definition 24). The lower the

Table 9.3: Notation summary

Notation	**Meaning**
T	Context (corresponds to a view of a factory room)
R	Set of recommended queries
$rq = \|R\| \in [1,4]$	Number of recommended queries
q_{max}	Query with maximal relevance to the context
q_u	User query
q_{div}	Query most similar to q_u
$s \in [0,8]$	Number of times the user has already sensed a context
$\delta \in [0.5,1]$	Similarity between q_u and q_{max}
$\gamma = 1000$	Query cardinality threshold
$\varepsilon = 0.8$	Context similarity threshold
$\eta = 0.2$	Fragment relevance threshold
$\theta \in [0.05, 0.25]$	Diversification threshold for A-BI

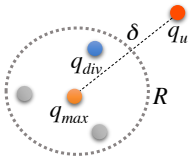


Figure 9.6: The user query q_u, the maximal query q_{max}, and the set R of recommended queries; among them, q_{div} is the one most similar to q_u

value of δ, the higher the difference between the user and maximal queries; if $\delta = 1$, the user exactly chooses the maximal query.

- q_{div} the query most similar to q_u among those in R.

A notation summary is provided in Table 9.3.

Queries q_u and q_{max} can be different since the relevance initially estimated by AO might not be aligned with the user's perceived one. This gap, evaluated in Section 9.6.3, decreases as the user returns in the same rooms since AO can exploit the log to align its estimation of relevance to the user's one. An intuitive representation of q_u, q_{max}, and q_{div} is shown in Figure 9.6, where the query space is represented as a Cartesian plane with Euclidean distances.

We executed our tests against a cube including 5 linear hierarchies with 5 levels each. Each dimension has 64 members, determining a maximum cube cardinality of about 10^9. The dictionary includes one entry for each md-element (i.e., we assume the smart device can recognize every single element of the cube); each dictionary entry d is mapped to a fragment f containing at most 3 md-elements. Mapping weights $w_{map}(d,e)$ randomly range in $[0.2,1]$. Note that AO entails mappings with higher expressiveness than A-BI, where

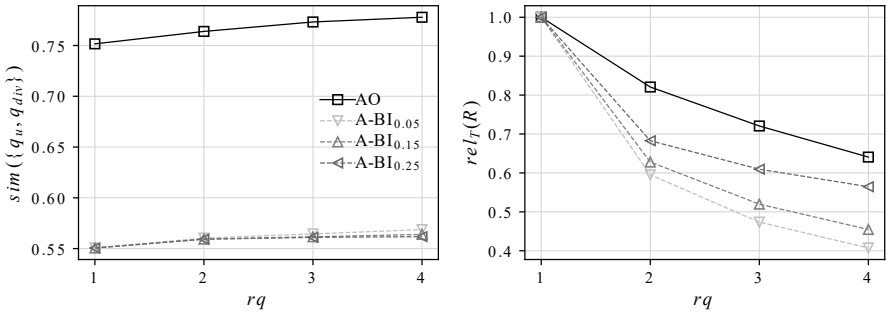


Figure 9.7: Average similarity between q_{div} and q_u (left) and average relevance of the recommended query set (right) for increasing values of rq ($s = 0$, $\delta = 0.7$, $|T| = 12$)

dictionary entries were mapped to single md-elements. Also, while in AO diversification is inherently tied to the maximization of the relevance of the returned queries, in A-BI it is ruled by a specific parameter, θ. We will compare the two approaches using different thresholds of diversification $\theta \in [0.05, 0.25]$; in the figures, with A-BI$_{0.15}$ we denote a run of A-BI with a diversification threshold set to 0.15. Values of θ higher than 0.25 deviate too much from the queries related to the context and are not considered [95].

9.6.1 Effectiveness

AO can recommend a variable number of queries, rq. The higher rq, the larger the user effort in choosing the best query out of the recommended ones. In an AR context, due to real-time and visualization constraints, this aspect becomes even more critical; thus, we limit the maximum number of retrieved queries to $rq = 4$.

Figure 9.7 characterizes R when different numbers of relevant queries are recommended to the user. The left part of the figure shows that the recommendation effectiveness, measured as the similarity between the user's query q_u and q_{div}, improves as rq increases. Remarkably, the similarity between q_{div} and q_u is always higher for AO than for A-BI, independently of the diversification strength θ used by A-BI. AO overcomes A-BI due to (1) its enriched mapping expressiveness; (2) the improved algorithm for generating relevant queries (Algorithm 4); and (3) the implicit diversification process. As to (1), given two md-elements that are only relevant if picked together, in AO they can only appear together in a query, while in A-BI they may be added to the query individually. As to (3), the diversification effectiveness in AO is better understood from Figure 9.7 (right), which shows that the relevance of the recommended query set is always superior to A-BI, independently of the diversification strength θ.

AO

Figure 9.8: Average similarity between q_{div} and q_u as q_u diverges from the context; solid and dashed lines refer to $rq = 1$ (no diversification) and $rq = 4$ (diversification), respectively ($s = 0$, $|T| = 12$)

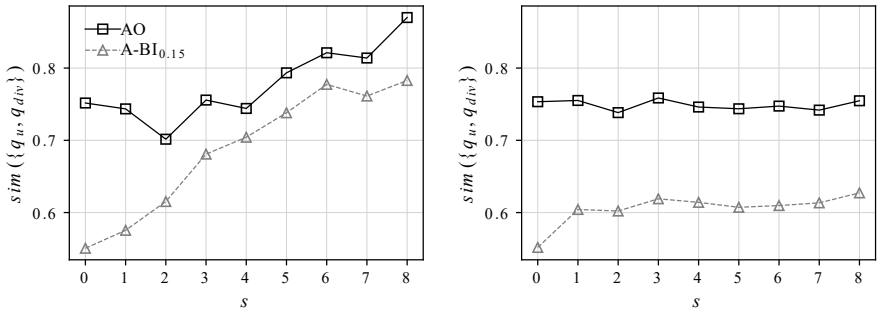


Figure 9.9: Average similarity between q_{div} and q_u for repeated visits when the context slightly changes (left) and q_u slightly changes (right) ($\delta = 0.7$, $rq = 1$, $|T| = 12$)

Although Figure 9.7 (left) shows that the similarity between q_{div} and q_u slightly increases with rq, the actual impact of diversification will be better appreciated in Section 9.6.3, where we will see that the users preferred a recommended query different from the most relevant query q_{max} in 15% of the times. This confirms the benefit of diversification in offering users different query flavors among which to choose [276]). Clearly, when there is a low correlation between the user's interest and the context, it becomes hard for a recommender to return useful answers. In our tests this divergence between the context and the user's query is simulated by increasing δ; Figure 9.8 shows that even in this case AO improves over A-BI, and that diversification helps in mitigating the correlation gap.

As rooms are repeatedly visited, collaborative filtering comes into play and the effectiveness of AO improves. Figure 9.9 depicts to what extent the query log helps in making q_{div} closer to q_u. In a real scenario, both the context and the user query q_u could slightly change in different visits. Figure 9.9 compares the recommendation effectiveness when the user query is fixed (left) and when the context is fixed (right). It is apparent that, when q_u is fixed, q_{div} quickly converges to q_u. Convergence is not complete due to context variations: like hybrid recommendation approaches [191], AO merges the user's interests stored in the log with the a-priori knowledge stored in the enriched image. If, for the very same context, the user requires slightly different queries across different visits, convergence is limited to the query fragments that are permanently required. In all cases, AO performs better than A-BI.

To better emphasize how the hybrid nature of AO impacts effectiveness, in Figure 9.10 we compare it to the two baselines given by its pure collaborative filtering behavior on the

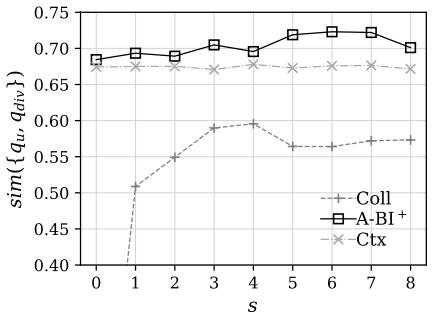


Figure 9.10: Average similarity between q_{div} and q_u for repeated visits when both the context and q_u slightly changes ($rq = 2$, $|T| = 12$)

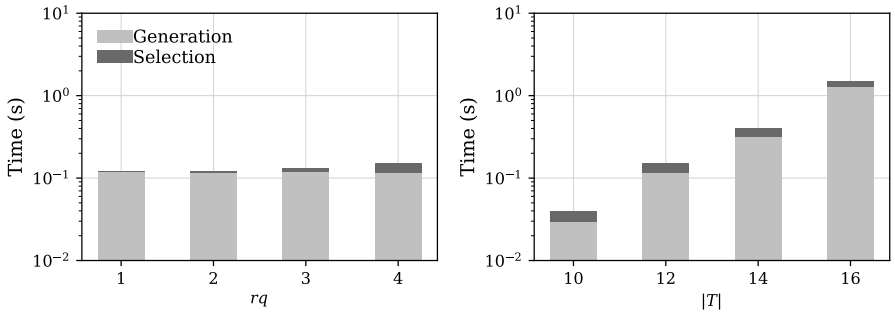


Figure 9.11: Efficiency of AO for increasing values of rq and $|T|$ ($s = 0$, $\delta = 0.7$; where not specified, $rq = 4$ and $|T| = 12$)

one hand, by its pure context-based behavior on the other. The first baseline, named *Coll*, returns the query that was chosen in the past from the most similar context; the second one, *Ctx*, returns the maximal query. Overall, *Coll* achieves worse performances than AO since (i) no query is returned for $s = 0$ (i.e., $sim() = 0$), (ii) it completely ignores the currently sensed entities as it only contains entities sensed in the past, and (iii) if the user picks different queries in similar contexts, collaborative filtering initially oscillates between different queries. Conversely, AO outperforms *Ctx* since the latter cannot keep into account the fragments that are actually chosen by the user even if they are not coded by mappings. Finally, *Coll* is worse than *Ctx* since its recommendations sum up two errors: the exclusion of currently sensed entities, and the inclusion of entities sensed in the past that are included in the target query.

9.6.2 Efficiency

We ran the tests on a machine equipped with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU and 8GB RAM, with the AO framework implemented in Scala; the log is stored in main

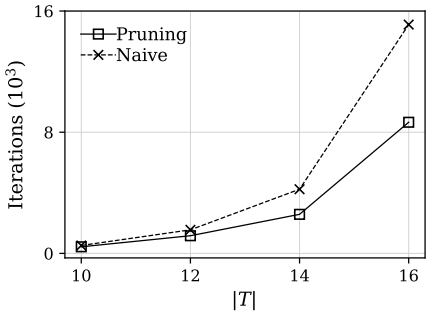


Figure 9.12: Effects of pruning in Algorithm 4 (Line 2) as the number of algorithm iterations

memory. Figure 9.11 (left) shows the *total* time required to recommend increasing numbers of queries. Remarkably, the order of magnitude is 10^{-1} seconds. Besides, although the execution time slightly increases with *rq*, the time needed to return the first recommendation is fixed as Algorithm 5 never drops a query once it has been added to the result set R. Figure 9.11 (right) shows the increase in execution time for larger contexts (the higher the number of context entries, the higher the number of mappings). Query generation (which encompasses Algorithms 3 and 4) accounts for most of the time; indeed, the generative approach is computationally heavy, requiring to find, inside the fragment space, a potentially exponential number of relevant queries. Pruning in Algorithm 4 (Line 2) helps in constraining the generation search space. Figure 9.12 shows how pruning decreases the (exponential) number of generated queries for different context cardinalities. Remarkably, when large contexts are considered it is possible to cut down generation times by constraining the number of generated queries.

We finally emphasize that the execution time corresponds to the time necessary to generate the recommended queries, and not to the time to actually execute them. Queries are executed against the enterprise data mart and their performance clearly depends on the underlying multidimensional engine.

9.6.3 User evaluation

There is no point in recommending a set of queries if the relevance estimated by the recommender significantly differs from the user's one. To assess how close the user's relevance is to the one of AO or, in other words, to assess the recommendation quality, we conducted a set of tests with 30 users, mainly master students in data science with basic or advanced knowledge of business intelligence and data warehousing. The evaluation is based on a real-world factory environment provided by Technogym, a large Italian company producing gym equipment. After a 10 minute introduction to AO, we simulated two user sessions in

which the user enters two rooms for the first time (i.e., the log is not considered). For each room, the user was asked to impersonate a production controller and to suggest a GPSJ query q_u that could help her in carrying out an assigned task based on a given context. To avoid biases, the assigned tasks were generic, meaning that there is not a single query that obviously fulfills the task, so the suggested queries depend on the personal interpretation of the task. In each room, once the user has provided her query, three queries recommended by AO were presented to her. Finally, the user was asked to provide (1) the *perceived* similarity of each recommended query to the query she suggested, and (2) a score (on a scale from 1 to 10) indicating how the recommended queries are deemed to be relevant to the context and to the proposed task. The first question enables the evaluation of how the similarity adopted in AO is perceived by the users independently of the relevance of the recommended queries to the context. Conversely, the second question is aimed at understanding the perceived relevance of the recommended queries to the context/task.

Example 18 (Room visit) *With reference to the* Assembly *cube and to the context represented in Figure 9.1*

$$T = \{\langle\{\mathsf{Object}, \mathrm{BikeExcite}\}\rangle, \langle\{\mathsf{Object}, \mathrm{Seat}\}\rangle, \\ \langle\{\mathsf{Date}, 20/05/2019\}\rangle, \langle\{\mathsf{Role}, \mathrm{Controller}\}\rangle\}$$

the assigned task is: "Analyze the assembly speed with reference to the context". Examples of queries recommended by AO are

$$\begin{aligned} q &= \langle\{\mathsf{Year}, \mathsf{Part}, \mathsf{Product}\}, \\ &\quad \{(\mathsf{Year} = 2019)\ \mathit{AND}\ (\mathsf{Part} = \mathrm{Seat})\ \mathit{AND}\ (\mathsf{Product} = \mathrm{BikeExcite})\}, \\ &\quad \{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle \\ q' &= \langle\{\mathsf{Year}, \mathsf{Part}, \mathsf{Category}\} \\ &\quad \{(\mathsf{Year} = 2019)\ \mathit{AND}\ (\mathsf{Part} = \mathrm{Seat})\ \mathit{AND}\ (\mathsf{Category} = \mathrm{Sport})\}, \\ &\quad \{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle \\ q'' &= \langle\{\mathsf{Date}, \mathsf{Part}, \mathsf{Category}\}, \\ &\quad \{(\mathsf{Date} = 20/05/2019)\ \mathit{AND}\ (\mathsf{Part} = \mathrm{Seat})\ \mathit{AND}\ (\mathsf{Category} = \mathrm{Sport})\}, \\ &\quad \{\mathsf{AssembledItems}, \mathsf{AssemblyTime}\}\rangle \end{aligned}$$

□

The results are summarized in Table 9.4. The average perceived similarity between the user query q_u and the three queries recommended by AO is 0.58 ± 0.20 for Room 1 and

Table 9.4: Results of user evaluation

	Room 1	**Room 2**
Avg. $sim()$	0.59 ± 0.15	0.61 ± 0.15
Avg. perceived similarity	0.58 ± 0.2	0.57 ± 0.2
Pearson correlation	0.41	0.41
q_{max} matches	62%	69%
q_{div} matches	77%	88%

0.57 ± 0.20 for Room 2, which is very close to the one computed through $sim()$ (0.59 ± 0.15 and 0.61 ± 0.15, respectively). Having near values is not enough to certify the coherence between the two similarities, thus we also computed their Pearson correlation coefficient obtaining an overall correlation of 0.41, which further supports this coherence.

As to the perceived relevance of recommended queries to the assigned tasks, the users evaluated the relevance of the recommended query set as 7.85 for Room 1 and 7.62 for Room 2, proving their satisfaction towards the recommendation. We finally emphasize that, without diversification ($rq = 1$), AO would return only the most relevant query, q_{max}, which turned out to be the most similar one to q_u in 62% of cases for Room 1 and 69% of cases for Room 2. When diversification is taken into account ($rq = 3$), these percentages increase to 77% for Room 1 and 88% for Room 2.

9.7 Conclusion

The AO framework is a first result in the direction of establishing a tight connection between analytical reporting and AR applications. Besides proposing a reference functional architecture and an interaction process, in this chapter we have shown that query recommendations can be given in real-time, highlighting the role of diversification and collaborative filtering in improving their effectiveness. Noticeably, our framework could be easily generalized to operate in other contexts, e.g., to recommend analytical queries concerning nearby objects based on the recognition of RFID tags.

AO can be improved along different directions. First of all, it would be interesting to investigate how AO could be turned into a purely statistical framework where all weights are expressed in terms of probabilities and reasoning is probabilistic as well; in this case, the log could be used to directly update mapping weights. Another possibility is to extend our model of context to a graph, so as to base recommendations on separate groups of entries (e.g., to distinguish foreground from background objects and to make mappings role-aware); this could be particularly relevant to take egocentric computer vision and engagement into

account [260]. Also the execution performances of recommended queries deserve further attention. Some possible enhancements here would be (i) to add a criterion for query selection that also considers an estimate of the query performance and (ii) to give users, for each recommended query, an estimate of its execution time plus a quick preview of its its results; note that the latter point would raise some interesting possibilities for multiquery optimization and caching. Finally, recommendation could also be extended from plain OLAP queries to complex analytics, e.g., anomaly detection: some event that is not in line with historical trends is going on, so it should be singled out.

Chapter 10

Conversational OLAP

10.1 Introduction

Nowadays, one of the most popular research trends in computer science is the democratization of data access, analysis, and visualization, which means opening them to end-users lacking the required vertical skills on the services themselves. Smart personal assistants [120] (Alexa, Siri, etc.) and auto-machine-learning services [250] are examples of such research efforts that are now on corporate agendas [34].

In particular, interfacing natural language processing (either written or spoken) to database systems opens to new opportunities for data exploration and querying [168]. Actually, in the area of data warehouse, OLAP itself is an *"ante litteram"* smart interface, since it supports the users with a "point-and-click" metaphor to avoid writing well-formed SQL queries. Nonetheless, the possibility of having a conversation with a smart assistant to run an OLAP session (i.e., a sequence of related OLAP queries) opens to new scenarios and applications. It is not just a matter of further reducing the complexity of posing a query: a conversational OLAP system must also provide feedback to refine and correct wrong queries, and it must have memory to relate subsequent requests. A reference application scenario for this kind of framework is *augmented OLAP* (Chapter 9), where hand-free interfaces are mandatory.

In this chapter, we propose COOL, a COnversational OLap framework able to convert a natural language text into a GPSJ query and to support query disambiguation and OLAP navigation. GPSJ [124] is the main class of queries used in OLAP since it enables Generalized Projection, Selection and Join operations over a set of tables. Although some natural language interfaces to databases have already been proposed, to the best of our knowledge this is the first proposal addressing full-fledged OLAP analytical sessions through a natural language interface.

In our vision, the desiderata for an OLAP smart interface are the following.

#1 It must be automated and portable: it must exploit cubes metadata (e.g., hierarchy structures, role of measures, attributes, and aggregation operators) to increase its understanding capabilities and to simplify the user-machine interaction process.

#2 It must handle OLAP sessions rather than single queries: in an OLAP session the first query is fully described by the text, while the following ones are implicitly/partially described by an OLAP operator (e.g., drill down, roll up, slice and dice) and require to handle the context and to have memory of the previous queries.

#3 It must be robust with respect to user inaccuracies in using syntax, OLAP terms, and attribute values; also, it must be able to exploit implicit information.

#4 It must be easy to configure on a data warehouse (DW) without a heavy manual definition of the lexicon.

More technically, our text-to-SQL approach is based on a grammar driving the parsing of natural language descriptions of GPSJ queries. The recognized entities include a set of typical query terms (e.g., group by, select) and the domain-specific terms and values automatically extracted from the DW (see desiderata #1 and #4). Robustness (desiderata #3) is one of the main goals for COOL and is pursued in all the interpretation steps: lexicon identification is based on a string similarity function, multi-word terms are handled through n-grams, and alternative query interpretations are scored and ranked. To sum up, the main contributions of this chapter are:

1. a list of features and desiderata for an effective conversational OLAP system;
2. an architectural view of COOL;
3. an original approach to translate a natural language analytical session into an OLAP session that starts with a well-formed GPSJ query (`Full query` step in Figure 10.1) and that refines the GPSJ query with known OLAP operators (`OLAP operator` step in Figure 10.1);
4. an analysis of the specificities of natural language interfaces in the OLAP context;
5. a set of tests to verify the efficiency and effectiveness of our approach. In particular, we carried out tests with real users to assess how well (i.e., quick, simple and accurate) COOL supports interactions.

The remainder of the chapter is organized as follows. Section 10.2 provides an overview of COOL by sketching the functional architecture and the interpretation steps.

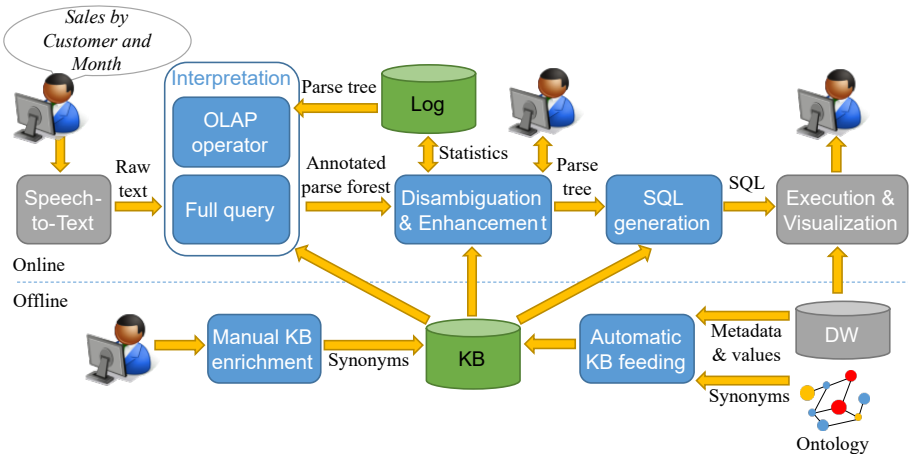


Figure 10.1: A functional architecture of COOL. Grayed-out elements are out of the chapter scope.

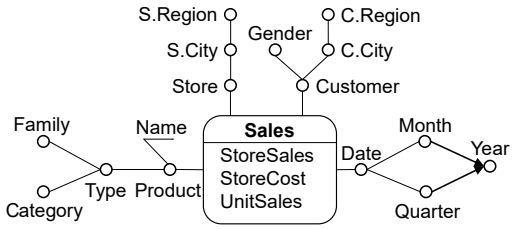


Figure 10.2: Simplified DFM representation of the Foodmart Sales cube.

Section 10.3 presents the contents of the `KB`, while the following sections introduce the core steps within the `Intepretation` step, i.e., `Tokenization & Mapping` (Section 10.4), `Parsing` (Section 10.5), and `Checking & Annotation` (Section 10.6). The remaining steps `Disambiguation & Enhancement` and `SQL generation` are respectively discussed in Section 10.7 and Section 10.8. In Section 10.9, a large set of tests assesses the effectiveness, efficiency, and user experience of COOL. Section 10.10 discusses related works on natural language interface to database systems. Finally, Section 10.11 draws the conclusions and discusses the evolution of COOL.

10.2 Overview of COOL

Figure 10.1 sketches a functional view of the architecture. Given a `DW`, that is a set of multidimensional cubes together with their metadata, we distinguish between an offline phase (to initialize and configure the system) and an online phase (to enable the user interaction).

10.2.1 The offline phase

The *offline* phase extracts the DW-specific terms used by users to express the queries. Such information are stored in the knowledge base of COOL (KB), which relies on the Dimensional Fact Model (DFM) expressiveness [112]. Noticeably, this phase runs only when the DW undergoes modification either in the cube schemas or in their instances. More in detail, the Automatic KB feeding process extracts the categorical attribute values and metadata from the cubes (e.g., attribute and measure names, table names, hierarchy structures) and possibly extends them with synonyms, automatically extracted from open data ontologies (Wordnet [197] in our implementation) to widen the language understood by COOL. Besides the domain-specific terminology, the KB also includes the set of standard OLAP terms that are domain independent and that do not require any feeding (e.g., group by, where, select). Further enrichment can be optionally carried out manually (i.e., by the Manual KB enrichment step) when the application domain involves a non-standard vocabulary (i.e., when the physical names of tables and columns do not match the words of a standard vocabulary). A closer look to the contents of the KB is given in Section 10.3.

10.2.2 The online phase

The *online* phase runs every time a natural language query is issued to COOL. The spoken query is initially translated to text by the Speech-to-text software module. This task is out of scope in our research and we exploited the public Web Speech API in our implementation (https://wicg.github.io/speech-api/).

The uninterpreted text is then analyzed by the Interpretation step that actually consists of two alternative steps: Full query is in charge of interpreting the texts describing full queries (which typically happens when an OLAP session starts), while OLAP operator modifies the latest query when the user states an OLAP operator along an OLAP session. The switch between the two steps to manage the conversation (i.e., a user/COOL dialog) is modeled by two states: *engage* and *navigate*.

- *Engage*: this is the initial state, in which the system expects a full query to be issued and whose interpretation is demanded to Full query. When COOL achieves a successful interpretation (i.e., it is able to run the query) it switches to the *navigate* state.
- *Navigate*: the dialogue evolves by iteratively applying OLAP operators that refine the query (i.e., which define an OLAP session). The management of these steps is demanded to OLAP operator until a *reset* command is applied, making COOL return to the *engage* state.

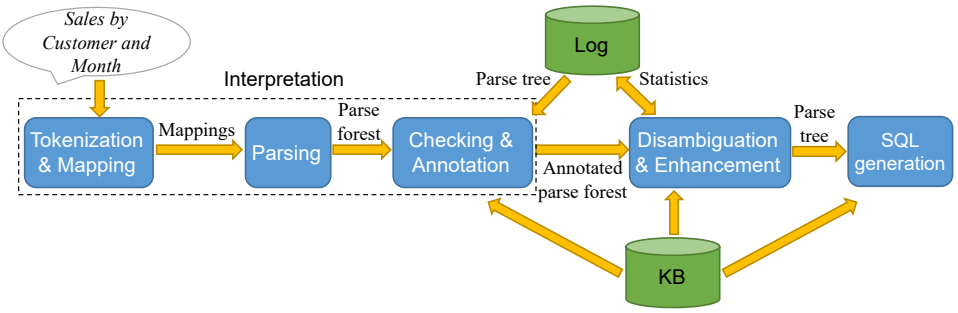


Figure 10.3: Interpretation of natural language; the KB is involved in all steps. The Interpretation steps are replicated for both Full query and OLAP operator.

On the one hand, understanding a single OLAP operator is simpler since it involves less elements than a complete GPSJ query. On the other hand, it requires to have memory of previous queries (stored in the Log) and to understand which part of the previous query must be modified. Both Full query and OLAP operator follow the computational steps represented in Figure 10.3: (i) Tokenization & Mapping (see Section 10.4), (ii) Parsing (see Section 10.5), and (iii) Checking & Annotation (see Section 10.6), but provide different implementations of (ii) and (iii).

Due to natural language ambiguities, speech-to-text inaccuracies and wrong query formulations (e.g., applying count operator on a measure or grouping by a descriptive attribute), part of the text can be misunderstood. The Disambiguation & Enhancement step solves ambiguities (if any) by asking appropriate questions to the user. The reasons behind the misunderstandings are manifold, including (but not limited to): ambiguities in the aggregation operator to be used; inconsistency between attribute and value in a selection predicate; identification of relevant elements in the text without understanding their role in the query.

The output of the previous steps is a data structure (i.e., a parse tree) that models the query and that can be automatically translated into an SQL query by exploiting the DW structure stored in the KB. Finally, the obtained query is run on the DW and the results are reported to the user by the Execution & Visualization software module. Such module could exploit a standard OLAP visualization tool or it could implement voice-based approaches [269] to create an end-to-end conversational solution. The visual interaction could rely on the DFM, which natively provides a graphical representation for multidimensional cubes and queries: such representation is conceptual and user-oriented, and its effectiveness is confirmed by its adoptions in commercial tools [143] for both modeling and descriptive purposes. Although we have built a prototype to enable the evaluation of COOL (see Section 10.9), the discussion of this module is out of the scope of the chapter.

10.3 The knowledge base

The Knowledge Base (KB in Figure 10.1) relies on the basic expressiveness of the DFM [112], which includes the multidimensional concepts introduced in Section 8.3.

The content of KB can be divided into *entities* and *structural information*. Entities compose the translated lexicon (i.e., the Interpretation step directly looks for their occurrence in the user's text), while structural information supports the interpretation (e.g., patterns necessary to recognize dates and numbers) and enables consistency checks on the interpreted query and the SQL generation(e.g., DW schema). More in detail, an entity $E = \langle t_1, ..., t_r \rangle$ is a sequence of textual words (i.e., a single distinct meaningful element of speech or writing). We refer to the set of *all* entities in the KB as $\mathscr{E} = \{E_1, ..., E_m\}$. Additionally, several synonyms can be stored for each entity (Table 10.1), enabling COOL to cope with slang and different shades of the text.

Orthogonally, entities and structural information are either *domain-agnostic* or *domain-dependent*. The domain-agnostic content includes those keywords and patterns that are typically used to express a query.

- **Intention keywords**: entities expressing the role of the subsequent part of text. Examples of intention keywords are group by, select and where.
- **Operators**: entities including logic (not, and, or), comparison ($=, <>, >, <, \geq, \leq$) and aggregation operators (e.g., sum, avg, min, max).
- **Patterns of dates and numbers**: structures used to automatically recognize dates and numbers in raw text.

The KB domain-dependent content is automatically collected by querying the DW and its data dictionary and is stored in a QB4OLAP [82] compatible repository.

- **DW element names**: entities corresponding to measures, dimensional attributes and fact names.
- **DW element values**: entities corresponding to values from categorical attributes (together with their frequency), and to statistical values (e.g., minimum and maximum) for numerical attributes.
- **Hierarchy structure**: information about the roll up relationships between attributes.
- **Aggregation operators**: information about the operators applicable to each measure and the default one.

Table 10.1: Sample of domain-agnostic and domain-specific entities with their synonyms. Domain-specific entities refer to the Sales cube from Figure 10.2

Domain	Type	Entity	Synonym samples
Agnostic	Intention keyword	where	in, on, such that, filter
		group by	by, for each, per
	Operator	$=; \geq$	equal to; greater than
		sum	total, amount
		avg	average, medium
Specific	DW element name	Sales	transactions
		UnitSales	quantities
		Gender	sex
	DW element value	*Drink*	beverage

- **DB tables**: information about the structure of the database implementing the DW, including table and attribute names, primary and foreign key relationships.

Example 19 (Cube, Entities and Synonyms) *In the* Sales *fact schema in Figure 10.2,* Product *and* Store *are dimensions,* Month *is a dimensional attribute and* Name *is a descriptive attribute.* StoreSales *and* UnitSales *are measures. It is possible to aggregate* StoreSales *using* sum *and* avg *aggregation operators. An example of GPSJ query would ask to "return the total quantity sold by month and type only for Italian stores". Drilling down from* Type *to* Product *means grouping on a finer attribute. Conversely rolling up from* Month *to* Year *means grouping on a coarser attribute. Finally, we can further slice and dice by adding a filter on a specific* Category *of products. With reference to the* Sales *cube,* Month *and* UnitSales *are domain-specific entities, while* avg *is a domain-agnostic entity. Examples of synonyms for* avg *are "average" and "medium".* □

10.4 Tokenization & mapping

A raw text T can be modeled as a sequence of tokens (i.e., single words) $T = \langle t_1, ..., t_z \rangle$. The goal of this step is to identify in T the entities, i.e., the only elements that will be involved in the Parsing step. Turning a text into a sequence of entities means finding a *mapping* between tokens in T and $\mathscr{E}$.

Definition 26 (Mapping & Mapping function) *A mapping function $M(T)$ is a partial function that associates sub-sequences (or n-*grams*) [1] from T to entities in $\mathscr{E}$ such that:*

[1] The term *n*-gram is used as a synonym of sub-sequence in the area of text mining.

- *sub-sequences of T have length n at most;*
- *the mapping function determines a partitioning of T;*
- *a sub-sequence $T' = \langle t_i, ..., t_l \rangle \in T$ (with $|T'| \leq n$) is associated to an entity E if and only if $Sim(T', E) > \alpha$ (where $Sim()$ is a similarity function, later defined) and $E \in TopN(\mathscr{E}, T')$ (where $TopN(\mathscr{E}, T')$ is the set of N entities in $\mathscr{E}$ that are the most similar to T' according to $Sim(T', E)$).*

The output of a mapping function is a sequence $M = \langle E_1, ..., E_l \rangle$ on $\mathscr{E}$ that we call a mapping. *Given the mapping M, the similarities $Sim(T', E_i)$ between each entity E_i and each corresponding token T' are also retained and denoted with $Sim_M(E_i)$. A mapping is said to be* valid *if the fraction of mapped tokens in T is higher than a given threshold β. We call $\mathscr{M}$ the set of valid mappings.*

Several mappings might exist between T and $\mathscr{E}$ since Definition 26 admits sub-sequences of variable lengths (corresponding to different partitionings of T) and associates the top similar entities to each sub-sequence. This increases interpretation robustness (since it allows to choose, in the next steps, the *best text interpretation* out of a higher number of candidates), but it can lead to an increase in computation time. The generated mappings differ both in the number of entities involved and in the specific entities mapped to a token. In the simple case where multi-token mappings are not possible (i.e., $n = 1$ in Definition 26) the number of generated mappings for a raw text T, such that $|T| = z$, is:

$$\sum_{i=\lceil z \cdot \beta \rceil}^{z} \binom{z}{i} \cdot N^i \tag{10.1}$$

The formula counts the possible configurations of sufficient length (i.e., higher or equal to $\lceil z \cdot \beta \rceil$) and, for each length, count the number of mappings determined by the top similar entities. Since the number of candidate mappings is exponential, we consider only the most significant ones through α, β, and N: α imposes sub-sequence of tokens to be very similar to an entity; N further imposes to consider only the N entities with the highest similarity; finally, β imposes a sufficient portion of the text to be mapped.

The similarity function $Sim()$ is based on the Levenshtein distance and keeps token permutation into account to make similarity robust to token permutations (e.g., sub-sequences $\langle P., Edgar \rangle$ and $\langle Edgar, Allan, Poe \rangle$ must result similar). Given two token sequences T and

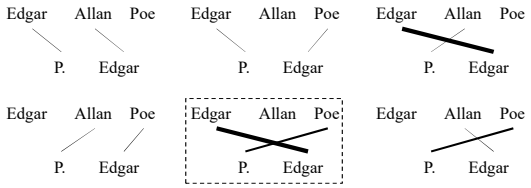


Figure 10.4: Token dispositions: arcs denote the correspondence of tokens for a specific disposition (the bolder the line, the higher the similarity). The disposition determining the maximum similarity is shown in a dashed rectangle.

W with $|T| = l, |W| = m$ such that $l \leq m$ it is:

$$Sim(\langle t_1, ..., t_l \rangle, \langle w_1, ..., w_m \rangle) = \max_{D \in Disp(l,m)} \frac{\sum_{i=1}^{l} sim(t_i, w_{D(i)}) \cdot \max(|t_i|, |w_{D(i)}|)}{\sum_{i=1}^{l} \max(|t_i|, |w_{D(i)}|) + \sum_{i \in \hat{D}} |w_i|}$$

where $D \in Disp(l,m)$ is an l-disposition of $\{1, ..., m\}$ and $\hat{D}$ is the subset of values in $\{1, ..., m\}$ that are not present in D. Function $Sim()$ weights token similarity based on their lengths (i.e., $\max(|t_i|, |w_{D(i)}|)$ and penalizes similarities between sequences of different lengths that imply unmatched tokens (i.e., $\sum_{i \in \hat{D}} |w_i|$).

Example 20 (Token similarity) *Figure 10.4 shows some of the possible token dispositions for the two token sequences* $T = \langle P., Edgar \rangle$ *and* $W = \langle Edgar, Allan, Poe \rangle$*. The disposition determining the highest similarity is surrounded by a dashed rectangle; the similarity is* 0.46 *and it is calculated as*

$$Sim(T,W) = \frac{sim(P., Poe)|Poe| + sim(Edgar, Edgar)|Edgar|}{|Poe| + |Edgar| + |Allan|}$$

□

We assume the best interpretation of the input text to be the one where (1) all the entities discovered in the text are included in the query (i.e., all the entities are parsed through the grammar) and, (2) each entity discovered in the text is perfectly mapped to one subsequence of tokens (i.e., $Sim_M(E_i) = 1$). The two previous statements are modeled through the following score function. Given a mapping $M = \langle E_1, ..., E_m \rangle$, we define its score as

$$Score(M) = \sum_{i=1}^{m} Sim_M(E_i) \tag{10.2}$$

The score is higher when M includes several entities with high values of Sim_M. Although at this stage it is not possible to predict if a mapping will be fully parsed, it is apparent that the higher the mapping score, the higher its probability to determine an optimal interpretation. As it will be explained in the Section 10.6, sorting the mappings by descending score also enables pruning strategies to be applied.

Example 21 (Tokenization & mapping) *With reference to Table 10.1, given the set of entities $\mathscr{E}$ and a tokenized text $T = \langle medium, sales, in, 2019, by, the, region \rangle$, examples of mappings M_1 and M_2 are:*

$$M_1 = \langle \mathsf{avg}, \mathsf{UnitSales}, \mathsf{where}, 2019, \mathsf{group\ by}, \mathsf{region} \rangle$$
$$M_2 = \langle \mathsf{avg}, \mathsf{UnitSales}, \mathsf{where}, 2019, \mathsf{group\ by}, \mathit{Regin} \rangle$$

where "medium" is mapped to avg *since "medium" is a known synonym of the aggregation operator* avg*, "in" is mapped to* where *since "in" can express time-related predicates, "the" is discarded being a stop word, and "region" is mapped to the attribute* Region *in M_1 and to the value "Regin" in M_2 (where "Regin" is a value of attribute* Customer *that holds a sufficient similarity).* □

10.5 Parsing

Parsing is the process of analyzing a sequence of entities (i.e., a mapping) to determine its syntactical structure with respect to a formal grammar. Parsing outputs a data structure called parse tree that is used by COOL to translate a mapping into SQL.

10.5.1 Full query parsing

In `Full query`, `Parsing` is responsible for the interpretation of a complete GPSJ query stated in natural language. Parsing a full query means searching in a mapping the complex syntax structures (i.e., *clauses*) that build-up the query. Given a mapping M, the output of a parser is a *parse tree PT_M*, i.e. an ordered tree that represents the syntactic structure of a mapping according to the grammar described in Figure 10.5. To the aim of parsing, entities are terminal elements in the grammar.

As a GPSJ query consists of 3 clauses (measure, group by and selection), in our grammar we identify four types of derivations[2]:

[2] A derivation in the form $\langle \mathsf{X} \rangle ::= e$ represent a substitution for the non-terminal symbol $\langle \mathsf{X} \rangle$ with the given expression e. Symbols that never appear on the left side of $::=$ are named terminals. Non-terminal symbols are enclosed between $\langle\rangle$, while terminal symbols are enclosed between " ".

⟨GPSJ⟩ ::= ⟨MC⟩⟨GC⟩⟨SC⟩ | ⟨MC⟩⟨SC⟩⟨GC⟩ | ⟨SC⟩⟨GC⟩⟨MC⟩ | ⟨SC⟩⟨MC⟩⟨GC⟩
| ⟨GC⟩⟨SC⟩⟨MC⟩ | ⟨GC⟩⟨MC⟩⟨SC⟩ | ⟨MC⟩⟨SC⟩ | ⟨MC⟩⟨GC⟩
| ⟨SC⟩⟨MC⟩ | ⟨GC⟩⟨MC⟩ | ⟨MC⟩

⟨MC⟩ ::= (⟨Agg⟩⟨Mea⟩ | ⟨Mea⟩⟨Agg⟩ | ⟨Mea⟩ | ⟨Cnt⟩⟨Fct⟩ | ⟨Fct⟩⟨Cnt⟩
| ⟨Cnt⟩⟨Attr⟩ | ⟨Attr⟩⟨Cnt⟩)+

⟨GC⟩ ::= "*group by*" ⟨Attr⟩+

⟨SC⟩ ::= "*where*" ⟨SCA⟩

⟨SCA⟩ ::= ⟨SCN⟩ "*and*" ⟨SCA⟩ | ⟨SCN⟩

⟨SCN⟩ ::= "*not*" ⟨SSC⟩ | ⟨SSC⟩

⟨SSC⟩ ::= ⟨Attr⟩⟨Cop⟩⟨Val⟩ | ⟨Attr⟩⟨Val⟩ | ⟨Val⟩⟨Cop⟩⟨Attr⟩ | ⟨Val⟩⟨Attr⟩ | ⟨Val⟩

⟨Cop⟩ ::= " = " | " <> " | " > " | " < " | " ≥ " | " ≤ "

⟨Agg⟩ ::= "*sum*" | "*avg*" | "*min*" | "*max*" | "*stdev*"

⟨Cnt⟩ ::= "*count*" | "*count distinct*"

⟨Fct⟩ ::= *Domain-specific facts*

⟨Mea⟩ ::= *Domain-specific measures*

⟨Attr⟩ ::= *Domain-specific attributes*

⟨Val⟩ ::= *Domain-specific values*

Figure 10.5: Backus-Naur representation of the `Full query` grammar. Entities from the `KB` are terminal symbols. "+" identifies a list of at least 1 symbol.

- **Measure clause** ⟨MC⟩: this derivation consists of a list of measure/aggregation operator pairs.
- **Group by clause** ⟨GC⟩: this derivation consists of a sequence of attribute names preceded by the entity "*group by*".
- **Selection clause** ⟨SC⟩: this derivation consists of a Boolean expression of simple selection predicates ⟨SSC⟩ preceded by the entity "*where*".
- **GPSJ query** ⟨GPSJ⟩: this derivation assembles the final query. Only the measure clause is mandatory since a GPSJ could aggregate a single measure with no selections. The order of clauses is irrelevant; this implies the proliferation of derivations due to clause permutations.

As shown in Figure 10.5, some derivations admit also partial forms (e.g., a measure clause ⟨MC⟩ missing its aggregation operator ⟨Agg⟩); in these cases, the `Checking & Annotation` step will try to infer the omitted derivations (see Section 10.6).

(a) The entire mapping is parsed (i.e., $PF_M = PT_M$). (b) "*group by*" $\langle \mathsf{Val} \rangle$ cannot be parsed.

Figure 10.6: Parse forests from Example 21.

The GPSJ grammar is LL(1)[3] [23], is not ambiguous (i.e., each mapping admits, at most, a single parse tree PT_M) and can be parsed by a LL(1) parser with linear complexity [23]. If the input mapping M is fully parsed, PT_M includes all the entities as leaves. Conversely, if only a portion of the input belongs to the grammar, an LL(1) parser produces a partial parsing, meaning that it returns a parse tree including the portion of the input mapping that belongs to the grammar (i.e., the PT rooted in $\langle \mathsf{GPSJ} \rangle$). The remaining entities can be either singletons or complex clauses that were not possible to connect to the main parse tree. We will call *parse forest* PF_M the union of the parse tree with residual clauses. Obviously, if all the entities are parsed, it is $PF_M = PT_M$. Considering the whole forest rather than the simple parse tree enables disambiguation and errors to be recovered in the `Disambiguation & Enhancement` step (Section 10.7). To keep the terminology simple, we will refer to the parser's output as a parse forest independently of the presence of residual clauses.

Example 22 (Parsing) *Figure 10.6 reports the parsing outcome for the two mappings in Example 21. M_1 is fully parsed, thus its parse forest corresponds to the parse tree (i.e., $PT_{M_1} = PF_{M_1}$). Conversely, in M_2 the last token is wrongly mapped to the attribute value* Regin *rather than to the attribute name* Region*. This prevents the full parsing and the parse tree PT_{M_2} does not include all the entities in M_2 (i.e., $PT_{M_2} \neq PF_{M_2}$).* □

10.5.2 OLAP operator parsing

In `OLAP operator`, `Parsing` is responsible for searching in a mapping the syntactic structures of the OLAP operators that build-up the conversation. The whole grammar is described in Figure 10.7 (we do not report grammar derivations $\langle \mathsf{Attr} \rangle$, $\langle \mathsf{MC} \rangle$ and $\langle \mathsf{SSC} \rangle$ in common

[3]The rules presented in Figure 10.5 do not satisfy LL(1) constraints for readability reasons. It is easy to turn such rules in a LL(1) compliant version, but the resulting rules are much more complex to be read and understood.

$$
\begin{aligned}
\langle \text{OPERATOR} \rangle ::= {} & \langle \text{DRILL} \rangle \mid \langle \text{ROLLUP} \rangle \mid \langle \text{SAD} \rangle \mid \langle \text{ADD} \rangle \mid \langle \text{DROP} \rangle \mid \langle \text{REPLACE} \rangle \\
\langle \text{DRILL} \rangle ::= {} & \text{“}\textit{drill}\text{”} \ \langle \text{Attr} \rangle_{from} \ \text{“}\textit{to}\text{”} \ \langle \text{Attr} \rangle_{to} \mid \text{“}\textit{drill}\text{”} \ \langle \text{Attr} \rangle \\
\langle \text{ROLLUP} \rangle ::= {} & \text{“}\textit{rollup}\text{”} \ \langle \text{Attr} \rangle_{from} \ \text{“}\textit{to}\text{”} \ \langle \text{Attr} \rangle_{to} \mid \text{“}\textit{rollup}\text{”} \ \langle \text{Attr} \rangle \\
\langle \text{SAD} \rangle ::= {} & \text{“}\textit{slice}\text{”} \ \langle \text{SSC} \rangle \\
\langle \text{ADD} \rangle ::= {} & \text{“}\textit{add}\text{”} \ (\langle \text{MC} \rangle \mid \langle \text{Attr} \rangle \mid \langle \text{SSC} \rangle) \\
\langle \text{DROP} \rangle ::= {} & \text{“}\textit{drop}\text{”} \ (\langle \text{MC} \rangle \mid \langle \text{Attr} \rangle \mid \langle \text{SSC} \rangle) \\
\langle \text{REPLACE} \rangle ::= {} & \text{“}\textit{replace}\text{”} \ (\langle \text{MC} \rangle_{old} \ \text{“}\textit{with}\text{”} \ \langle \text{MC} \rangle_{new} \mid \langle \text{Attr} \rangle_{old} \text{“}\textit{with}\text{”} \ \langle \text{Attr} \rangle_{new} \\
& \mid \langle \text{SSC} \rangle_{old} \ \text{“}\textit{with}\text{”} \ \langle \text{SSC} \rangle_{new})
\end{aligned}
$$

Figure 10.7: Backus-Naur representation of the `OLAP operator` grammar. Entities from the `KB` are terminal symbols. We omit the derivations $\langle \text{MC} \rangle, \langle \text{Attr} \rangle, \langle \text{SSC} \rangle$ that are in common with Figure 10.5. Decoration of non-terminals with subscript is used for the sake of description.

with Figure 10.5). Our conversation steps are inspired by well-known OLAP visual interfaces (e.g., Tableau[4]). To apply an OLAP operator, COOL must be in state *navigate* (i.e., a full GPSJ query has been already successfully interpreted). By definition, the interpreted query correspond to a parse tree PT_C that acts as a context for the operator. For the sake of explanation, we assume that PT_C takes the form $\langle \text{GPSJ} \rangle ::= \langle \text{MC} \rangle \langle \text{GC} \rangle \langle \text{SC} \rangle$ (even if $\langle \text{GC} \rangle$ and/or $\langle \text{SC} \rangle$ can be missing), and with a slight abuse of notation we adopt the set notation to denote that a clause is contained in another clause (e.g., $\langle \text{Attr} \rangle \in \langle \text{GC} \rangle$ and $\langle \text{GC} \rangle \in \langle \text{GPSJ} \rangle$).

- **Drill down** $\langle \text{DRILL} \rangle$: this derivation substitutes the coarser attribute $\langle \text{Attr} \rangle_{from} \in \langle \text{GC} \rangle$ with a finer attribute $\langle \text{Attr} \rangle_{to}$.
- **Roll up** $\langle \text{ROLLUP} \rangle$: this derivation substitutes the finer attribute $\langle \text{Attr} \rangle_{from} \in \langle \text{GC} \rangle$ with a coarser attribute $\langle \text{Attr} \rangle_{to}$.
- **Slice and dice** $\langle \text{SAD} \rangle$: this derivation specifies a new selection predicate $\langle \text{SSC} \rangle \in \langle \text{SC} \rangle$.
- **Add** $\langle \text{ADD} \rangle$: this derivation adds a measure/attribute/selection clause to the corresponding clause $\langle \text{MC} \rangle / \langle \text{GC} \rangle / \langle \text{SC} \rangle \in \langle \text{GPSJ} \rangle$.
- **Drop** $\langle \text{DROP} \rangle$: this derivation drops a measure/attribute/selection clause from the corresponding clause $\langle \text{MC} \rangle / \langle \text{GC} \rangle / \langle \text{SC} \rangle \in \langle \text{GPSJ} \rangle$.
- **Replace** $\langle \text{REPLACE} \rangle$: this derivation combines **Add** and **Drop** to substitute an existing *old* clause, either measure, attribute or selection, for a *new* one.

[4]https://www.tableau.com/

Similarly to the GPSJ grammar (Figure 10.5), the OLAP operator grammar admits partial forms for **Drill down** and **Roll up** (e.g., only a single attribute ⟨Attr⟩ is provided in a ⟨DRILL⟩ clause) that the `Checking & Annotation` step will try to complete (see Section 10.7). Also note that the **Add** and **Drop** operators respectively behave as **Drill down** and **Roll up** when applied to attributes; otherwise, they behave as **Slice and dice** when applied to selection clauses.

Example 23 (Conversational OLAP Session) *Given the parse tree from Figure 10.9, an example of conversation is the following. At first the user issues the natural language sentence "Drill down region to city", COOL parses such sentence (Figure 10.8a), recognizes a drill down operation and modifies the previous parse tree (Figure 10.8b). Then, the user asks to "replace the unit sales with the sum of store sales". Despite the aggregation operation is not specified, COOL recognizes the measure clause (Figure 10.8c) to be substituted and replaces it with the new one (Figure 10.8d). Finally, the user asks to "slice on milk" (Figure 10.8e): COOL automatically infers the attribute containing the value "milk" and combines the new condition with the previous existing clause (Figure 10.8f).* □

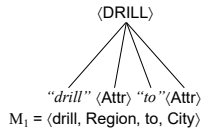
10.6 Parse forest checking and annotation

The `Parsing` step does not always output a parse forest that can be directly translated into executable SQL code. Indeed, syntactic adherence to the grammar does not guarantee the conformance of the full query (or OLAP operator) with multidimensional structure and constraints. As it happens for compilers, parsing and type checking (i.e., verifying and enforcing the constraints of data types) are kept separated to reduce the overall complexity.

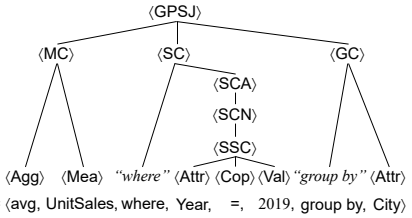
In this step, COOL seeks for these problems, and annotates the parse forest accordingly. Two types of annotations are possible.

- ***Ambiguity***: an inconsistency solvable through disambiguation. Ambiguities are solvable either automatically by COOL or by interacting with the user (e.g., *"sum unit sales for Salem"*, but *Salem* is member of both City and StoreCity);
- ***Error***: an inconsistency that does not admit solution and that can be only notified to the user (e.g., *"remove unit sales"*, but the measure UnitSales is not included in ⟨GPSJ⟩).

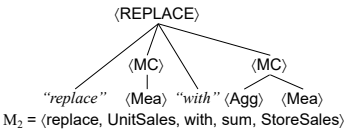
`Checking & Annotation` is responsible for searching problematic clauses (i.e., subtrees) in the parse forest and annotating them. Depending on the type of the clause, `Checking & Annotation` evaluates the conformance of the clause to the multidimensional structure and constraints and marks the subtrees failing such constraints.



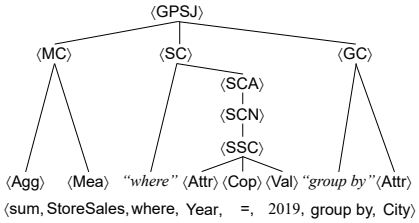
(a) Parsing "Drill down region to city".



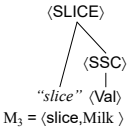
(b) Drill down Region to City w.r.t. Figure 10.6a.



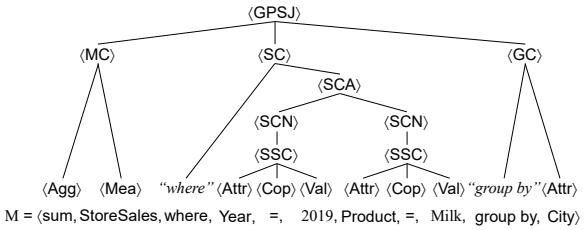
(c) Parsing "Replace unit sales with the sum of store sales".



(d) Replace Avg UnitSales with Sum StoreSales.



(e) Parsing "Slice on milk".



(f) Add a new slice on Product = *Milk*.

Figure 10.8: A conversational session modifies the parse tree from Figure 10.6a.

We now describe the allowed annotations for the parse forests produced by `Full query` and `OLAP operator`.

10.6.1 Full query annotation

In `Full query`, `Checking & Annotation` searches *PT* for problematic clauses (i.e., subtrees) in a depth-first fashion [265]. Table 10.2 reports the annotations resulting from failing checks.

- **Ambiguous attribute (AA)**: the ⟨SSC⟩ clause has an implicit attribute but the parsed value belongs to multiple attribute domains.
- **Ambiguous aggregation operator (AAO)**: the ⟨MC⟩ clause has an implicit aggregation operator but the measure is associated with multiple aggregation operators.
- **Attribute-value mismatch (AVM)**: the ⟨SSC⟩ clause includes a value that does not belong to the domain of the specified attribut.

Table 10.2: `Full query` annotations

Type	Name	Gen. derivation sample	Example
Ambiguity	AA	⟨SSC⟩ ::= ⟨Val⟩	*"sum unit sales for Salem"*, but *Salem* is member of City and StoreCity
Ambiguity	AAO	⟨MC⟩ ::= ⟨Mea⟩	*"unit sales by product"*, but sum and avg are valid aggregations
Ambiguity	AVM	⟨SSC⟩ ::= ⟨Attr⟩⟨Cop⟩⟨Val⟩	*"sum unit sales for product New York"*, but *New York* is not a Product
Ambiguity	MV	⟨MC⟩ ::= ⟨Agg⟩⟨Mea⟩	*"sum prices per store"*, but Price is not additive
Ambiguity	AV	⟨GC⟩ ::= "*group by*" ⟨Attr⟩+	*"average prices by name"*, but Product is not in ⟨GC⟩
Ambiguity	UC	–	*"average unit sales by Regin"*, but *Regin* is not an attribute

- **Violation of a multidimensional constraint on a measure (MV)**: the ⟨MC⟩ clause contains an aggregation operator that is not allowed for the specified measure.
- **Violation of a multidimensional constraint on an attribute (AV)**: the ⟨GC⟩ clause contains a descriptive attribute without the corresponding dimensional attribute[5].
- **Unparsed clause (UC)**: A clause has been properly parsed, but the parser was not able to to connect it to the ⟨GPSJ⟩ derivation. Thus, a parse forest including a ⟨GPSJ⟩ derivation and one or more dangling clauses has been returned.

10.6.2 OLAP operator annotation

Although the `OLAP operator` grammar returns simpler parse trees than `Full query`, annotating an OLAP operator is more complex since the parse tree must be internally coherent and *also* compliant to the previous query context, i.e. the latest full query to which the OLAP operator should be applied. Let PT_C be the parse tree for the previous query context, and let PT be the parse tree of the OLAP operator. In `OLAP operator`, `Checking & Annotation` extends the checks in Section 10.6.1 by searching also for possible inconsistencies between each clause in PT_C and PT. Checking is achieved in a depth-first fashion [265] and exploits the PT_C structure to reduce the search space (e.g., it is meaningful to search for a ⟨SSC⟩ only within the ⟨SC⟩ ∈ ⟨GPSJ⟩). As a consequence, checking an OLAP operator requires additional constraints and annotations with respect to the ones defined in Section 10.6.1 (Table 10.3):

- **Not Existing Clause (NEC)**: PT references a clause that should be present in PT_C, but that is missing. For example, $\langle \text{Attr} \rangle_{from}$ in the ⟨DRILL⟩ and ⟨ROLLUP⟩ derivations must exist in ⟨GC⟩ of PT_C;

[5] According to DFM a *descriptive attribute* is an attribute that further describes a dimensional level (i.e., it is related one-to-one with the level), but that can be used for aggregation only in combination with the corresponding level.

Table 10.3: `OLAP operator` annotations

Type	Name	Gen. derivation sample	Example
Ambiguity	TO	⟨DRILL⟩ ::= "*drill*" ⟨Attr⟩	*"drill down from year"*
Ambiguity	FR	⟨DRILL⟩ ::= "*drill*" ⟨Attr⟩	*"drill down to year"* but ⟨GC⟩ contains Month and Quarter
Error	NEC	⟨DROP⟩ ::= "*drop*" ⟨MC⟩	*"remove unit sales"* but UnitSales is not in ⟨GPSJ⟩
Error	AEC	⟨ADD⟩ ::= "*add*" ⟨MC⟩	*"add max unit sales"* but max UnitSales is already in ⟨GPSJ⟩
Error	ID	⟨DRILL⟩ ::= "*drill*" ⟨Attr⟩	*"drill down from product"*
Error	FA	⟨DRILL⟩ ::= "*drill*" ⟨Attr⟩	*"drill down from name"*
Error	HM	⟨DRILL⟩ ::= "*drill*" $\langle Attr \rangle_{from}$ "*to*" $\langle Attr \rangle_{to}$	*"drill down from product to city"*
Error	HSM	⟨DRILL⟩ ::= "*drill*" $\langle Attr \rangle_{from}$ "*to*" $\langle Attr \rangle_{to}$	*"drill down from product to type"*

- **Already Existing Clause (AEC)**: PT cannot be applied as it references an element that is already present in PT_C, while it should not be there. For example, $\langle Attr \rangle_{to}$ in the ⟨DRILL⟩ and ⟨ROLLUP⟩ derivations cannot exist in ⟨GC⟩ of PT_C;
- **Invalid drill (ID)**: it is impossible to drill down on $\langle Attr \rangle_{from}$ in a ⟨DRILL⟩ derivation, since $\langle Attr \rangle_{from}$ is already the finest attribute in the hierarchy.
- **Forbidden Attribute (FA)**: it is impossible to drill down/roll up on a descriptive attribute.
- **Hierarchy mismatch (HM)**: $\langle Attr \rangle_{from}$ and $\langle Attr \rangle_{to}$ in the ⟨DRILL⟩ (or ⟨ROLLUP⟩) derivation belong to two different hierarchies.
- **Hierarchy structure mismatch (HSM)**: $\langle Attr \rangle_{from}$ in the ⟨DRILL⟩ (or ⟨ROLLUP⟩) derivation is finer (or coarser) than $\langle Attr \rangle_{to}$.
- **Branching ambiguity (TO)**: it is impossible to infer $\langle Attr \rangle_{to}$ in a ⟨DRILL⟩ (or ⟨ROLLUP⟩) derivation due to the presence of a branch in the hierarchy.
- **Rolling ambiguity (FR)**: it is impossible to infer $\langle Attr \rangle_{from}$ in a ⟨DRILL⟩ (or ⟨ROLLUP⟩) derivation due to the presence of multiple finer (or coarser) attributes in ⟨GC⟩.

10.6.3 Parse forest scoring

A textual query generates several parse forests, one for each mapping. In our approach, only the most promising one is proposed to the user in the `Disambiguation & Enhancement` step. This choice comes from two main motivations:

- Proposing more than one alternative queries to the user can be confusing and makes very difficult to contextualize the disambiguation questions.

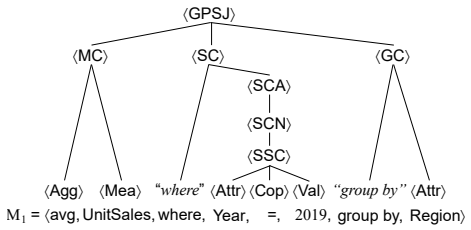


Figure 10.9: Parse forest enhancement: the implicit attribute Year has been added to M_1 in Figure 10.6a.

- Proposing only the most promising choice makes it easier to create a baseline query, even though the optimal derivation could be missed. The baseline query can still be improved by adding or removing clauses through further interactions enabled by the `OLAP operator` step.

Definition 27 (Parse Forest Score) *Given a mapping M and the corresponding parse forest PF_M, we define its score as $Score(PF_M) = Score(M')$ where M' is the sub-sequence of M belonging to the parse tree PT_M.*

The parse forest holding the highest score is the one proposed to the user. This ranking criterion is based on an *optimistic-pessimistic* forecast of the outcome of the `Disambiguation & Enhancement` step. On the one hand, we optimistically assume that the ambiguities belonging to PT_M will be positively solved in the `Disambiguation & Enhancement` step and the corresponding clauses and entities will be kept. On the other hand, we pessimistically assume that non-parsed clauses belonging to PF_M will be dropped. The rationale of our choice is that an annotated clause included in the parse tree is more likely to be a proper interpretation of the text. As shown in Figure 10.10, a totally pessimistic criterion (i.e., exclude from the score all the annotated clauses and entities) would carry forward a too simple, but non-ambiguous, forest; conversely, a totally optimistic criterion (i.e., consider the score of all the entities in PF_M) would make preferable a large but largely non-parsed forest. Please note that the bare score of the mapping (i.e., the one available before parsing) corresponds to a totally optimistic choice since it sums up the scores of all the entities in the mapping.

The ranking criterion defined above enables the pruning of the mappings to be parsed as shown by Algorithm 6. Reminding that mappings are parsed in descending score order, let us assume that, at some step, the best parse forest is $PF_{M'}$ with score $Score(PF_{M'})$. If the next mapping to be parsed, M'', has score $Score(M'') < Score(PF_{M'})$, we can stop the algorithm and return $PF_{M'}$ since the score of M'' is an upper bound to the (optimistic) score of the corresponding parse forest.

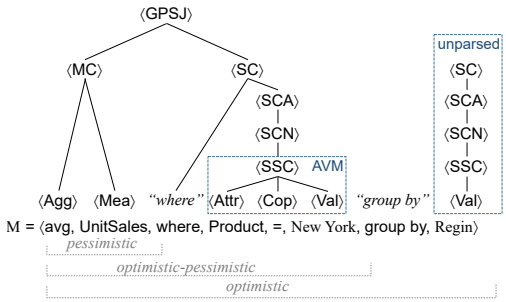


Figure 10.10: Portion of the parse forest contributing to its score depending on the adopted scoring criterion.

Algorithm 6 Selection of the parse forest

INPUT $\mathscr{M}$: set of valid mappings
OUTPUT PF^*: best parse forest

1: $\mathscr{M} \leftarrow sort(\mathscr{M})$ ▷ sort mappings by score
2: $PF^* \leftarrow \varnothing$
3: **while** $\mathscr{M} \neq \varnothing$ **do** ▷ while mapping space is not exhausted
4: $M \leftarrow head(\mathscr{M})$ ▷ get the mapping with highest score
5: $\mathscr{M} \leftarrow \mathscr{M} \setminus \{M\}$ ▷ remove it from $\mathscr{M}$
6: $PF_M \leftarrow parse(M)$ ▷ parse the mapping
7: **if** $score(PF_M) > score(PF^*)$ **then**
▷ if current score is higher than previous score
8: $PF^* \leftarrow PF_M$ ▷ store the new parse forest
9: $\mathscr{M} \leftarrow \mathscr{M} \setminus \{M' \in \mathscr{M}, score(M') \leq score(PF^*)\}$
▷ remove mappings with lower scores from $\mathscr{M}$
return PF^*

Algorithm 6 works as follows. At first, mappings are sorted by their score (Line 1), the best parse forest is initialized, and the iteration begins. While the set of existing mappings is not exhausted (Line 3), the best mapping is picked, removed from the set of candidates, and its parse forest is generated (Lines 4–6). If the score of the current forest is higher than the score of the stored one (Line 7), then the current forest is stored (Line 8) and all the mappings with a lower score are removed from the search space (Line 9) as the pruned mappings cannot produce parse forests with a score greater than what has been already parsed.

Note that, as *Score*() requires a parse forest and the `Parsing` step always produces a parse forest (which might coincide with the parse tree), *Score*(), ranking and pruning work for both `Full query` and OLAP `operator` steps.

10.7 Parse forest disambiguation and enhancement

If no ambiguities or errors were found in the previous steps, the parse forest coincides with a parse tree[6] that can be directly translated into an executable SQL query. Conversely, if COOL detects an error the only possible solution is to return an informative warning to the user in order to help her resubmit a correct command. If none of the above apply, an annotated parse forest is returned. An annotation does not necessarily imply an interaction with the user to be solved. Indeed, COOL tries to minimize the number of such interactions by exploiting the `KB` and previous activities of the user stored in the `Log`. At the end of the `Disambiguation & Enhancement` step, the ambiguous parse forest is reduced to a non-ambiguous parse tree as all the ambiguities are solved in this step (existing unparsed clauses are either added to ⟨GPSJ⟩ or dropped).

We identify three ways to solve ambiguities: implicit, default-based, and user-based.

Implicit Refers to the cases where the parse forest does not include all the information necessary to produce the SQL code, but the missing parts can be automatically inferred from the `KB` since only one solution is possible. In this case the parse forest is automatically completed and no interaction with the user is required. Examples of automatic inference are the following.

- **Implicit aggregation operator in ⟨MC⟩**: the aggregation operator in a ⟨MC⟩ clause is implicit and the measure is associated with a single aggregation operator.
- **Implicit attribute in ⟨SSC⟩**: the ⟨SSC⟩ clause has an implicit attribute and the parsed member belongs to a single attribute domain.
- **Implicit attribute in ⟨DRILL⟩ (or ⟨ROLL⟩)**: the ⟨DRILL⟩ (or ⟨ROLL⟩) clause has an implicit attribute, and, based on the previous query context, it is necessary to infer its role (*from* or *to*) to complete the OLAP operator. Given the attribute ⟨Attr⟩ in the ⟨DRILL⟩ (or ⟨ROLL⟩) operator, if $Attr \in \langle GC \rangle$, we assume *from* as the role of ⟨Attr⟩, and $\langle Attr \rangle_{to}$ is inferred as the finer (or coarser) attribute of ⟨Attr⟩. Otherwise, if $Attr \notin \langle GC \rangle$, we assume *to* as the role of ⟨Attr⟩, and $\langle Attr \rangle_{from}$ is inferred as the coarser (or finer) attribute of ⟨Attr⟩.

Default-based Refers to the cases where more than one solution can be applied to complete the parse tree but either a default solution has been defined in the `KB` or it can be inferred from

[6] As unparsed clauses are annotated as ambiguities, if no ambiguities are found then all clauses are included in the parse tree.

Table 10.4: User interaction templates and actions for *user-based* ambiguity solution. The line splits annotations related to full-query interpretation from those related to OLAP-operator interpretation

Name	Template	Action
AA	⟨Val⟩ is member of these attributes [...]	Pick attribute/drop
AAO	⟨Mea⟩ allows these operators [...]	Pick operator/drop
AVM	⟨Attr⟩ and ⟨Val⟩ domains mismatch, possible values are [...]	Pick value/drop
MV	⟨Mea⟩ does not allow ⟨Agg⟩, possible operators are [...]	Pick operator/drop
AV	Impossible to group by on ⟨Attr⟩ without ⟨Attr⟩	Add attribute/drop
UC ⟨GC⟩	There is a dangling grouping clause ⟨GC⟩	Add to ⟨GPSJ⟩/drop
UC ⟨MC⟩	There is a dangling measure clause ⟨MC⟩	Add to ⟨GPSJ⟩/drop
UC ⟨SC⟩	There is a dangling predicate clause ⟨SC⟩	Add to ⟨GPSJ⟩/drop
TO	$\langle \mathsf{Attr} \rangle_{from}$ can be generalized/specialized to [...]	Pick attribute/drop
FR	$\langle \mathsf{Attr} \rangle_{to}$ can be specialized/generalized from [...]	Pick attribute/drop
NEC	The specified clause does not exist in ⟨GPSJ⟩	Alert & drop
AEC	The specified clause already exists in ⟨GPSJ⟩	Alert & drop
ID	Impossible to drill down from the finest attribute ⟨Attr⟩	Alert & drop
FA	Impossible to roll up/drill down on the descriptive attribute ⟨Attr⟩	Alert & drop
HM	$\langle \mathsf{Attr} \rangle_{from}$ and $\langle \mathsf{Attr} \rangle_{to}$ belong to different hierarchies	Alert & drop
HSM	$\langle \mathsf{Attr} \rangle_{from}$ is coarser/finer than $\langle \mathsf{Attr} \rangle_{to}$	Alert & drop

the Log according to the previous user disambiguations. In this case, the preferred solution is applied and the user interaction is limited to evidencing it as *hint*. The user can *optionally* manually refine this solution. More details on this case are provided later in this section.

User-based Refers to the remaining cases where no automatic solutions apply. In this case a user interaction is required to disambiguate. Table 10.4 reports the user interaction templates and actions for *user-based* ambiguity resolutions. Each template allows to either provide the missing information or to drop the annotated clause. Templates are standardized and user choices are limited to keep the interaction easy. This allows also unskilled users to obtain a baseline query.

As for default-based ambiguities, if the Log highlights a recurring choice in the way of solving ambiguities, COOL infers a preference and this knowledge can be leveraged to reduce the number of user interactions, turning a user-based solution in a smoother default-based one. To this end, similarly to [96], at each disambiguation step d we store in the Log L a triple $d = (A, T, T^*)$, where $d.A$ is the annotation name, $d.T$ is the annotated subtree and $d.T^*$ is the disambiguated subtree (eventually empty if the user dropped T). Given L, the log-based disambiguation frequency is defined as the ratio between the number of times in which a certain replacement is chosen to resolve an ambiguity and the number of times in which the ambiguity occurred.

$$f(d) = \frac{|\{d' \in L \; s.t. \; d'.A = d.A, d'.T = d.T, d'.T^* = d.T^*\}|}{1 + |\{d' \in L \; s.t. \; d'.A = d.A, d'.T = d.T\}|}$$

Algorithm 7 Log-based Disambiguation

INPUT PF_M: annotated parse forest, τ: frequency threshold
OUTPUT PF_M^*: partially solved annotated parse forest

1: $PF_M^* \leftarrow PF_M$ ▷ Initialize the parse forest
2: **for each** $d \in annotations(PF_M)$ **do** ▷ Iterate over its annotated subtrees
3: $T^* = argmax_{T'} f((d.A, d.T, T'))$ ▷ Get the most frequent disambiguation
4: **if** $f((d.A, d.T, T^*)) \geq \tau$ **then** ▷ If frequency is higher than threshold
5: Replace $d.T$ with T^* in PF_M^* ▷ ... replace it
6: Annotate T^* as hint ▷ ... annotate the subtree as hint
7: **return** PF_M^*

Given the frequency threshold τ, Algorithm 7 shows the automatic disambiguation process. Given an annotated parse forest (Line 1), the algorithm iterates over the annotated subtrees (Line 2). For each annotation, the algorithm picks the disambiguation with the highest frequency (Line 3). If the frequency is higher than a given threshold (Line 4), then the algorithm replaces the annotated subtree with the frequent one within the parse forest (Line 5) and marks it as a hint (Line 6).

Example 24 (Log-based disambiguation) *Given the mapping* $M = \langle \mathsf{sum}, \mathsf{UnitSales}, \mathsf{where}, \mathit{New\ York} \rangle$, `Parsing` *outputs a parse forest where the subtree* $T' = SSC(New\ York)$[7] *corresponding to the SSC clause New York (where New York is a member of both* StoreCity *and* CustomerCity*) is annotated with an ambiguous attribute AA annotation. Given a threshold* $\tau = 0.5$ *and the* `Log`

$$L = \{(AA,\ SSC(New\ York),\ SSC(\mathsf{StoreCity}, =, New\ York)),$$
$$(AA,\ SSC(New\ York),\ SSC(\mathsf{StoreCity}, =, New\ York)),$$
$$(AA,\ SSC(New\ York),\ SSC(\mathsf{CustomerCity}, =, New\ York)),\ ...\}$$

the subtree T' *is automatically replaced with* $T^* = SSC(\mathsf{StoreCity}, =, New\ York)$ *as* $f((AA,\ SSC(New\ York),\ SSC(\mathsf{StoreCity}, =, New\ York))) = \frac{2}{4} = 0.5 \geq \tau$. □

10.8 SQL generation

SQL generation translates a full-query parse tree into an executable SQL query. If an OLAP operator has been submitted, the context parse tree PT_C must be updated according to the OLAP operator parse tree PT. All the OLAP operators can be implemented atop the

[7] Elements between brackets represent children of the parent node. The node *SCC* is the parent of the node *New York*.

addition/removal of new/existing nodes in PT_C. As in Section 10.6.2, we apply a depth-first search algorithm to retrieve the clauses interested by the OLAP operator. We recall that adding a new clause to ⟨GPSJ⟩ (e.g., “add city" requires to add the attribute City) requires to append the new clause to the existing ⟨MC⟩/⟨GC⟩/⟨SC⟩ (and to create it if it does not exist in ⟨GPSJ⟩). Note that an ⟨SSC⟩ is appended with the Boolean and operator as in Figure 10.8f. Conversely, when dropping a clause produces an invalid parent clause (e.g., an empty ⟨GC⟩ or an unbalanced ⟨SSA⟩) in the parse tree, it is sufficient to remove the parent clause from ⟨GPSJ⟩.

Given a full query parse tree PT, the generation of its corresponding SQL requires to fill in the `SELECT`, `WHERE`, `GROUP BY` and `FROM` statements. The SQL generation applies to both star and snowflake schemas [112] and is done as follows:

- `SELECT`: measures and aggregation operators from ⟨MC⟩ are added to the query selection clause together with the attributes in the group by clause ⟨GC⟩;
- `WHERE`: predicates from the selection clause ⟨SC⟩ (i.e., values and their respective attributes) are added to the query predicate;
- `GROUP BY`: attributes from the group by clause ⟨GC⟩ are added to the query group by set;
- `FROM`: measures and attributes/values identify, respectively, the fact and the dimension tables involved in the query. Given these tables, the join path is identified by following the referential integrity constraints (i.e., by following foreign keys from dimension tables imported in the fact table).

Example 25 (SQL generation) *Given the GPSJ query “sum the unit sales by type in the month of July", its corresponding SQL is:*

```
SELECT Type, sum(UnitSales)
FROM Sales s JOIN Product p ON (s.pid = p.id)
             JOIN Date d ON (s.did = d.id)
WHERE Month = "July"
GROUP BY Type
```

□

Table 10.5: Parameter values for testing

Symbol	Range	Meaning
N	$\{2,4,6\}$	Num. of top similar entities
α	$\{0.4,0.5,0.6\}$	Token/entity minimum similarity
β	70%	Sentence coverage threshold
n	4	Maximum sub-sequence length

10.9 Experimental tests

In this section, we evaluate COOL in terms of effectiveness, efficiency, and user experience. Effectiveness and efficiency evaluate COOL's capabilities in correctly interpreting a text in a time compatible with real-time interaction. User experience evaluates how well COOL helps the user in formulating OLAP sessions. Indeed, there is no point in creating a natural language interface that is not easy to use by inexperienced users.

Tests are based on the Foodmart schema[8]. The `Automatic KB feeding` module populated the `KB` with 1 fact, 39 attributes, 12500 entities. 50 additional synonyms were manually added in the `KB enrichment` step (e.g., *"for each"*, *"for every"*, *"per"* are synonyms of the group by statement).

To the best of our knowledge, no standard benchmark exists for natural language GPSJ queries. Nonetheless, [75] describes a real-word benchmark for generic analytics queries. 110 queries out of 147 (i.e. 75% of the benchmark) turned out to meet the GPSJ expressiveness, confirming how general and standard GPSJ queries are. Since queries in [75] refer to private datasets, we mapped them to the Foodmart schema. While mapping natural language to the sales domain, we preserved the structure of the original queries (e.g., word order, typos, etc.). For each query, we manually defined the ground truth, i.e. the parse tree resulting from the correct text interpretation.

10.9.1 Effectiveness

Tests in this section quantitatively evaluate how well COOL interprets natural language queries. Effectiveness is evaluated as the parse tree similarity $TSim(PT, PT^*)$ between the parse tree PT produced by COOL and the correct one PT^* (i.e., the manually-defined ground truth). Parse tree similarity is based on the tree distance [305]: it ranges in $[0,1]$ and it keeps into account both the number of correctly parsed entities (i.e., the parse tree leaves) and the

[8] A public dataset about food sales between 1997 and 1998 (https://github.com/julianhyde/foodmart-data-mysql).

tree (i.e., query) structure (e.g., the selection clauses *"(A and B) or C"* and *"A and (B or C)"* refers to the same parsed entities but underlie different tree structures).

Table 10.5 summarizes the parameters considered in our approach: token sub-sequences has maximum length $n = 4$ (i.e., the [1..4]-grams) as no entity in the `KB` is longer than 4 words; each sub-sequence is associated to the top N similar entities with similarity at least higher than α such that at least a percentage β of the tokens in T is covered. The value of β is fixed to 70% based on an empirical evaluation of the benchmark queries.

We first consider the behavior of COOL without disambiguation. Figures 10.11 and 10.12 depict the performance of our approach with respect to variations in either the number of retrieved top similar entities (i.e., $N \in \{2,4,6\}$) or the similarity threshold (i.e., $\alpha \in \{0.4,0.5,0.6\}$). Values are reported for the top-k trees (i.e., the k trees with the highest score). We remind that only one parse forest is involved in the `Disambiguation & Enhancement` step; nonetheless, for testing purposes, it is interesting to see if the best parse tree belongs to the top-k ranked ones. Effectiveness slightly changes by varying N and it ranges in $[0.88,0.91]$. Effectiveness is more affected by the entity/token similarity threshold α and ranges in $[0.83,0.91]$. In both cases, the best results are obtained when more similar entities are admitted, and more candidate mappings are generated. Independently of the chosen thresholds, the results of COOL are very stable (i.e., the effectiveness variations are limited) and, even by considering only the top-1 query, its effectiveness is at the state of the art [168, 289, 240]. This confirms that (i) the choice of proposing only one query to the user does not negatively impact on performances (while it positively impacts on interaction complexity and efficiency) and (ii) our scoring function properly ranks parse tree similarity to the correct interpretation for the query since the best ranked is in most cases the most similar to the correct solution.

Only 58 queries out of 110 are not ambiguous and produce parse trees that can be fed *as-is* to `SQL generation` and `Execution & Visualization`. This means that 52 queries — despite being very similar to the correct tree, as shown by the aforementioned results — are not directly executable without disambiguation (we recall the ambiguities from Table 10.2). Indeed, of these 52 queries, 38 contain one ambiguity annotation, 12 contain two ambiguity annotations, and 2 contain three or more ambiguity annotations. Figure 10.13 depicts the performance when the best parse tree undergoes iterative disambiguation (i.e., an increasing number of correcting actions is applied). Starting from the best configuration from the previous tests (for $N = 6$ and $\alpha = 0.4$), by applying an increasing number of correcting actions, the effectiveness increases from 0.89 up to 0.94. Unsolved differences between PT and PT^* are mainly due to missed entities in the mappings.

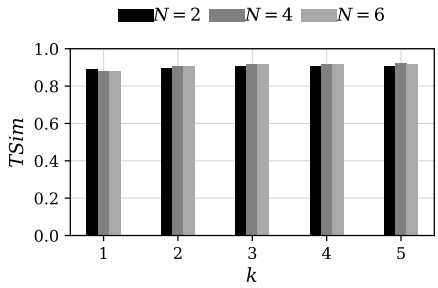


Figure 10.11: Effectiveness varying the number of retrieved top similar entities N and the number of $Top\text{-}k$ queries returned (with $\alpha = 0.4$).

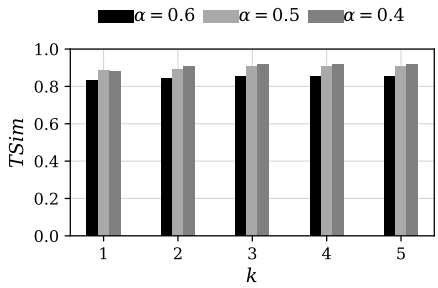


Figure 10.12: Effectiveness varying the similarity threshold α and the number of $Top\text{-}k$ queries returned (with $N = 6$).

Although our approach shows an effectiveness comparable to the state-of-the-art proposals [168, 289, 240], it was impossible to run a comparison against them as (i) the implementations are private and the provided descriptions are far from making them reproducible, and (ii) despite the availability of natural language datasets (e.g., the ones used in [168]), these datasets are hardly compatible with multidimensional constraints and GPSJ queries.

10.9.2 Efficiency

In Section 10.4, we have shown that the mapping search space increases exponentially in the text length. Figure 10.14 confirms this result showing the number of generated mappings as a function of the number of entities $|M|$ included in the optimal parse tree PT^*, $|M|$ is strictly related to the number of tokens in the text, $|T|$. Note that, with reference to the parameter values reported in Table 10.5, the configuration analyzed in Figure 10.14 is the worst case since it determines the largest search space due to the high number of admitted similar

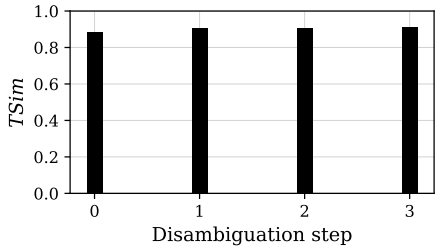


Figure 10.13: Effectiveness as a function of the number of disambiguation steps (i.e., application of correcting actions; with $k = 1$, $N = 6$ and $\alpha = 0.4$).

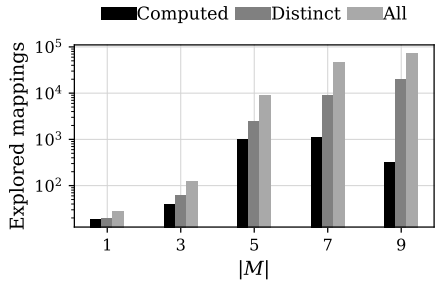


Figure 10.14: Number of generated, distinct and parsed mappings varying the number $|M|$ of entities in the optimal tree (with $N = 6$ and $\alpha = 0.4$). Given *all* the mappings generated in the `Tokenization & Mapping` step, some mappings are identical and only the *distinct* ones are kept. The ranking and scoring functions further prune the actually *computed* mapping.

entities. Noticeably, pruning rules strongly limit the number of mappings to be actually parsed.

We ran the tests on a machine equipped with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU and 8GB RAM, with COOL implemented in Java. Figure 10.15 shows the average execution time by varying $|M|$ and the number of allowed top similar entities N. The execution time increases with the number of entities included in the optimal parse tree, as such also the number of top similar entities impacts the overall execution time. Effectiveness remains high also for $N = 6$, corresponding to an execution time of 1 second. We emphasize that the execution time corresponds to the time necessary for the interpretation, and not to the time to actually execute them. Queries are executed against the enterprise data mart and their performance clearly depends on the underlying multidimensional engine and on the complexity of the query itself.

It is worth reasoning on what happens by increasing the number of attributes/members. As for efficiency, the performance of COOL are based on two aspects: mapping and parsing. Increasing the number of attributes/members affects mapping due to the wider search space

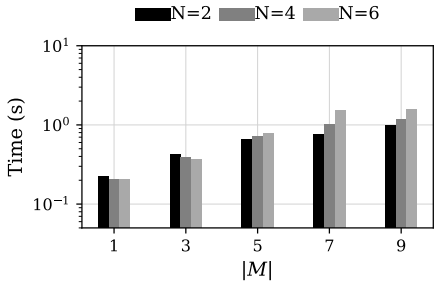


Figure 10.15: Execution time varying the number $|M|$ of entities in the optimal tree and the number of top similar entities N (with $\alpha = 0.4$).

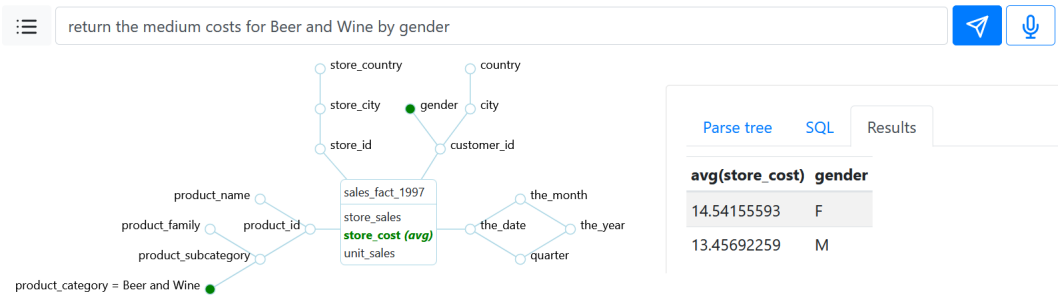
for entity similarity (see Equation (10.1)). However, as we pick a fixed number of synonyms, this does not affect parsing since the number of generated mappings does not change. As for effectiveness, a larger number of attributes/members could reduce accuracy depending on the similarity between them. Indeed, a textual token will probably match more entities. However, this really depends on the dataset at hand. Nonetheless, when the quality of the spoken/written English is high (e.g., few typos or text matches DW entities with high similarity) it is intuitive to verify that COOL matches the *right* entities.

10.9.3 User experience evaluation

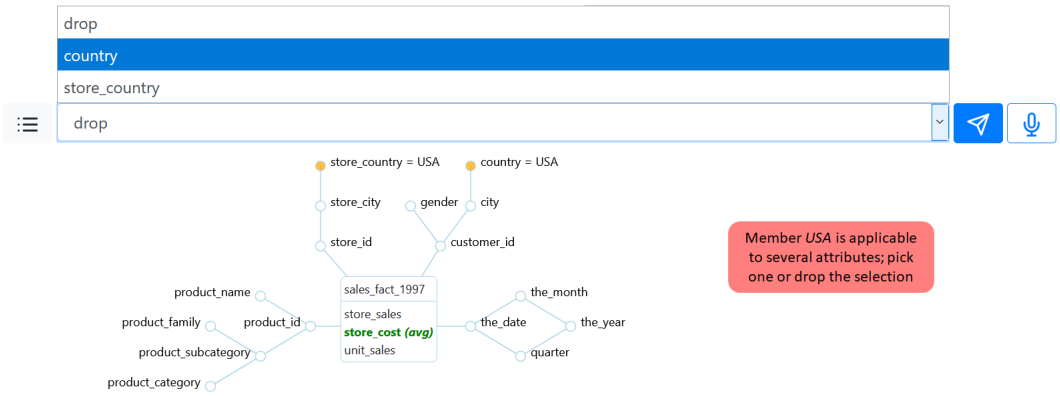
The main goal of a natural language interface is to enable a user to easily formulate a command. To this end, we tested COOL with 40 users, mainly master students in data science, with basic or advanced knowledge of business intelligence and data warehousing. On a scale from 1 (very poor) to 5 (very high), on average, users scored 3.60 ± 0.7 their familiarity with the English language, and 3.28 ± 1.1 their familiarity with the OLAP paradigm.

For the sake of testing, we implemented a web application atop the DFM model (Figure 10.16) in which users submit a written or spoken descriptions of full queries/OLAP operators. The following testing protocol has been adopted:

- A 10-minute tutorial was presented to show the users how COOL works and how the test is organized.
- Users undergo 3 OLAP sessions $\{s_a, s_b, s_c\}$ with increasing complexity. Each OLAP session is composed by three steps (q, op', op''), where q is a full query and op' and op'' are the OLAP operators that are iteratively applied to the full query. In each OLAP session, we provided three formal queries (q, q', q''), with q' (or q'') slightly changing from q (or q'). At first, users were asked to produce the natural language description



(a) The non-ambiguous query *"return the medium costs for Beer and Wine by gender"* is issued. Elements highlighted in green have been understood by COOL.



(b) The ambiguous query *"return the medium costs in USA"* is issued. COOL asks to the user if *USA* is member of Country or StoreCountry or if *USA* should be dropped (ambiguity AA in Table 10.4). Elements highlighted in yellow require user-based disambiguation.

Figure 10.16: User interface of COOL implemented for the testing purpose.

of the full query q. Then, by difference with the following query, users were asked to understand which OLAP operator op changed q (or q') to q' (or q''), and to issue the natural language description of op to COOL.

- At each step, users are asked to issue the natural language description in English to COOL. If fully parsed, COOL visualizes the query result as a pivot table (Figure 10.16a). Otherwise, the disambiguation process starts, and users are asked to disambiguate what COOL was not capable to automatically infer (Figure 10.16b).
- When all ambiguities (if any) are solved, the next step in the session is proposed.
- Users explicitly end the OLAP session after running q''.

Example 26 (OLAP session) *Given the formal full query*

$$q = \{\{\mathsf{avg\ StoreCost}\}, \{\mathsf{gender}\}, \mathsf{category} = \text{“}\textit{Beer and Wine}\text{”}\}$$

a user is asked to produce a natural language description of the query, such as “return the medium costs for Beer and Wine by gender". As COOL is capable of fully interpret this query (Figure 10.16a), the next formal query

$$q' = \{\{\mathsf{avg\ StoreCost}\}, \{\mathsf{gender}, \mathsf{month}\}, \mathsf{category} = \text{“}\textit{Beer and Wine}\text{”}\}$$

is presented to the user. User is asked to understand which OLAP operator can be applied to q to produce q' and to issue its natural language description to COOL. Note that the applicable OLAP operator is not unique. For instance, depending on the familiarity with the OLAP paradigm, the user may choose either to “drill down to month" or to “add the month". □

We call PT_s^u the parse tree of the first full query issued by the user (i.e., the user description of q), and PT_e^u the parse tree of the last full query modification (i.e., when the user ends the session). We call PT_s^* and PT_e^* our ground truth, i.e. the parse trees corresponding to q and q''. Note that there is not a single correct/predefined path driving the OLAP session. For instance: (i) the OLAP session can diverge/converge from/to the ground truth (e.g., a full query that does not comprehend all the necessary entities can be completed with consequent OLAP operators); users can concatenate ⟨ADD⟩ and ⟨REMOVE⟩ instead of the ⟨REPLACE⟩ operator; or some disambiguation steps might be necessary to complete the session due to ambiguities or inaccuracies.

For each OLAP session, we evaluate both the starting full query and the consequent OLAP operators. We adopt the following key performance indicators (KPIs): accuracy $TSim()$, interpretation time T, extra user interactions I (in the optimal case $I = 0$; i.e., the interpretation discloses no ambiguities that require user interactions). In detail:

- `Full query`. $TSim(PT_s^u, PT_s^*)$ measures how good is the interpretation of the full query; T_s adds up the time it takes for the user to formulate the query and COOL to interpret it; and I_s counts the number of user interactions to produce a fully-parsed full query.

- `OLAP operator`. $TSim(PT_e^u, PT_e^*)$ measures how close the user query is to the correct query after applying the OLAP operators. For each OLAP operator op, T_{op} sums the time it took for the user to formulate the operator, and for COOL to interpret them

Table 10.6: Results of user evaluation in terms of average accuracy $TSim()$, average elapsed time T (in seconds), average number of user interactions I, and average number of ambiguities A

	Full query			OLAP operator		
Id	$TSim(PT_s^u, PT_s^*)$	$T_s(s)$	I_s	$TSim(PT_e^u, PT_e^*)$	$T_{op}(s)$	I_{op}
s_a	0.98	203	0.26	0.96	159	0.47
s_b	0.92	210	0.65	0.80	131	0.29
s_c	0.91	183	0.61	0.94	77	0.11
All	0.94	198	0.51	0.90	122	0.29

and to apply it to the user full query; and I_{op} counts the number of user interactions to produce a fully-parsed OLAP operator.

All KPIs are averaged on all users (Table 10.6). Following the previous effectiveness evaluation, we set $\alpha = 0.4$, $\beta = 70\%$, $N = 6$ and $n = 4$. As for effectiveness, the average accuracy for full query interpretation is 0.94, consolidating the results depicted in Section 10.9.1. The decrease in accuracy $TSim(PT_s^u, PT_s^*)$ from s_a to s_c is justified by the increasing complexity of the initial full query. The considerations on accuracy also hold for the average similarity of the OLAP operator (besides some vocabulary issues on s_b solvable by enriching the knowledge base). As for efficiency, as expected, the time it takes for a user to formulate a full query is higher than the time needed for the interpretation and application of an OLAP operator (3 minutes vs 2 minutes). We argue that the time could be sensibly lower if users were already familiar with the multidimensional cube. Indeed, the time required for the OLAP operator sensibly decreases along with an increasing familiarity with COOL (the same happens for the full query, but the effect is compensated by the increasing complexity). As for interactions, the small number of extra interactions—together with the high accuracy—proves COOL's robustness in natural language interpretation (we recall that misinterpreted clauses count as unparsed ambiguities). In turn, this enable users to issue complete queries and OLAP operators in either 1 or 2 interactions (i.e., 0 or 1 extra interactions).

10.10 Related works

Conversational business intelligence can be classified as a *natural language interface* (NLI) to *business intelligence* systems to drive analytic sessions. Despite the plethora of contributions in each area, to the best of our knowledge, no approach lies at their intersection.

NLIs to operational databases enable users to specify complex queries without previous training on formal programming languages (such as SQL) and software; a recent and

comprehensive survey is provided in [7]. Overall, NLIs are divided into two categories: question answering and dialog. While the former are designed to operate on single queries, only the latter are capable of supporting sequences of related queries as needed in OLAP analytic sessions. However, to the best of our knowledge, no dialog-based system for OLAP sessions has been provided so far. The only contribution in the dialog-based direction is [182], where the authors provide an architecture for querying relational databases; with respect to this contribution we rely on the formal foundations of the multidimensional model to drive analytic sessions (e.g., according to the multidimensional model it is impossible to group by a measure, compute aggregations of categorical attributes, aggregate by descriptive attributes, ensure drill-across validity). Also differently from [182], the results we provide are supported by extensive effectiveness and efficiency performance evaluation that completely lack in [182]. Finally, existing dialog systems, such as [239], address the exploration of linked data. Hence, they are not suitable for analytics on the multidimensional model.

As for question answering, existing systems are well understood and differentiate for the knowledge required to formulate the query and for the generative approach. Domain agnostic approaches solely rely on the database schema. NaLIR [168] translates natural language queries into dependency trees [188] and brute-forcefully transforms promising trees until a valid query can be generated. In our approach we rely on *n*-grams instead of dependency trees [188] since the latter cannot be directly mapped to entities in the knowledge base (i.e., they require tree manipulation) and are sensible to the query syntax (e.g., *"sum unit sales"* and *"sum the unit sales"* produce two different trees with the same meaning). SQLizer [289] generates templates over the issued query and applies a "repair" loop until it generates queries that can be obtained using at most a given number of changes from the initial template. Domain-specific approaches add semantics to the translation process by means of domain-specific ontologies and ontology-to-database mappings. SODA [31] uses a simple but limited keyword-based approach that generates a reasonable and executable SQL query based on the matches between the input query and the database metadata, enriched with domain-specific ontologies. ATHENA [240] and its recent extension [248] map natural language into an ontology representation and exploit mappings crafted by the relational schema designer to resolve SQL queries. Analyza [67] integrates the domain-specific ontology into a "semantic grammar" (i.e., a grammar with placeholders for the typed concepts such as measures, dimensions, etc.) to annotate and finally parse the user query. Additionally, Analyza provides an intuitive interface facilitating user-system interaction in spreadsheets. Unfortunately, by relying on the definition of domain-specific knowledge and mappings, the adoption of these approaches is not plug-and-play as an ad-hoc ontology is rarely available and is burdensome to create.

In the area of business intelligence, the road to conversation-driven OLAP is not paved yet. The recommendation of OLAP sessions to improve data exploration has been well-understood [13] also in domains of unconventional contexts [95] where hand-free interfaces are mandatory. Recommendation systems focus on integrating (previous) user experience with external knowledge to suggest queries or sessions, rather than providing smart interfaces to BI tools. To this end, personal assistants and conversational interfaces can help users unfamiliar with such tools and SQL language to perform data exploration. However, end-to-end frameworks are not provided in the domain of analytic sessions over multidimensional data. QUASL [161] introduces a QA approach over the multidimensional model that supports analytical queries but lacks both the formalization of the disambiguation process (i.e., how ambiguous results are addressed) and the support to OLAP sessions (with respect to QA, handling OLAP sessions requires to manage previous knowledge from the `Log` and to understand whether the issued sentence refines previous query or is a new one). Complementary to COOL, [269] recently formalized the vocalization of OLAP results.

To summarize, the main differences of our approach to the previous works are the following.

1. The implementation of an end-to-end general-purpose *dialog-driven* framework named COOL, supporting full-fledged OLAP sessions.

2. The definition of the framework's functional architecture and the formalization of its steps.

3. The enforcement of multidimensional constraints on GPSJ queries by the means of (i) a formal grammar ensuring syntactic validity, and (ii) a type checker ensuring consistency (e.g., it is impossible to assign the "sum" aggregation operator to a non-additive measure).

4. A plug-and-play implementation that allows COOL to run on top of existing data warehouses with no impact on it; furthermore, the integration with external knowledge is supported but not mandatory.

Being an NLI, this work builds on fundamental notions of Natural Language Processing (NLP), such as tokenization, parsing, and disambiguation. We refer the reader to a referential contribution in the areas of Natural Language Processing [187] to further delve into this subject. With respect to the single steps of the approach, we remark the following differences.

- Our parsing technique Section 10.5 is based on a novel formal grammar, used to interpret OLAP queries and OLAP operations expressed in natural language and to

obtain parse trees. Among related approaches: [31] relies on a limited keyword-based technique; [240] uses a similar technique, even though its grammar is not specific to OLAP and is aimed at mapping text tokens to ontology concepts; [168, 289, 67] rely on standard natural language parsers where the produced dependency trees are sensitive to small linguistic variations in the user query.

- By analyzing the specificities of natural language interfaces in the OLAP context, we formally define and retrieve the ambiguities that may arise in OLAP queries and operations (Section 10.6). With respect to related approaches, this allows us to discover and manage ambiguities that are related more to the OLAP metaphor rather than to natural language per se.

- Related work deals differently with the disambiguation activity. [289] relies on a completely automatic and probabilistic method to resolve ambiguities, whereas [67] always prompts the user. [168] adopts a hybrid approach by distinguishing easy ambiguities (automatically solvable) from hard ambiguities (requiring user interaction); [31, 240] also adopt an automatic solving technique, but by relying on a domain-specific ontology. COOL integrates these methods by i) implicitly solving some ambiguities (also thanks to the OLAP specific parsing and annotation techniques), ii) inferring the solution from the log whenever possible, and iii) by asking the user when no solution has been found (Section 10.7).

- Every related approach defines its way to generate SQL queries from the data structure parsed from the user query. The technique we discuss in Section 10.8 is specifically tied to our novel grammar.

- The techniques we adopt for tokenization and mapping (Section 10.4) are not specific to OLAP; nonetheless, a detailed explanation of tokenization and mapping is important to introduce basic concepts that are later used and to ensure the reproducibility of the approach.

We sketched the idea of Conversational OLAP in [88]; this chapter largely extends the previous contribution by (i) proposing and implementing a solution for the interpretation and disambiguation of OLAP operators, (ii) providing a log-based ambiguity resolution mechanism that automatically resolves ambiguities by learning the most frequent disambiguations, (iii) providing a visual interface to handle the interaction, designed on top of the DFM model (see Section 10.9), and (iv) carrying out extensive tests with real users to assess the usability of COOL.

10.11 Conclusion

In this chapter, we proposed COOL, a conversational OLAP framework supporting the translation of a natural language conversation into an OLAP session. COOL supports both the interpretation of GPSJ queries and OLAP operators. Besides proposing a technical solution and a reference architecture, the chapter contribution lies in the discussion of specific issues related to conversational OLAP systems, paving the way to the application of conversational OLAP in the context of smart assistants and hand-free scenarios. Tests carried out on a real-word benchmark, as well as with 40 real users, show that COOL achieves the state-of-the-art performance. More in detail, we have shown that coupling a grammar-based text interpretation with automatic/user-based disambiguation determines a 94% accuracy. We are completing COOL with new functionalities essential for improving user experience, such as the *vocalization* of *query summaries* (i.e., applying mining techniques to reduce the cardinality of the returned results as in [97]) and the support for a full-fledged visual metaphor based on the DFM model.

Chapter 11

Intentional OLAP

11.1 Introduction

Data warehousing and OLAP have been progressively gaining a leading role in enabling business analyses over enterprise data since the early 90's. During these thirty years, the underlying technologies have evolved from the early relational implementations (still widely adopted in corporate environments), to the new architectures solicited by Business Intelligence 2.0 scenarios, and up to the challenges posed by the integration with big data settings. However, recently, it has become more and more evident that the OLAP paradigm, alone, is no more sufficient to keep the pace with the increasing needs of new-generation decision makers. Indeed, the enormous success of machine learning techniques has consistently shifted the interest of corporate users towards sophisticated analytical applications.

In this direction, the *Intentional Analytics Model* (IAM) has been envisioned as a way to tightly couple OLAP and analytics [275]. The IAM approach relies on two major cornerstones: (i) the user explores the data space by expressing her analysis *intentions* rather than by explicitly stating what data she needs, and (ii) in return she receives both multidimensional data and knowledge insights in the form of annotations of interesting subsets of data. As to (i), five intention operators are proposed, namely, describe (describes one or more cube measures, possibly focused on one or more level members), assess (judges one or more cube measures with reference to some baseline), explain (reveals some hidden information in the data the user is observing, for instance in the form of a correlation between two measures), predict (shows data not in the original cubes, derived for instance with regression), and suggest (shows data similar to those the current user, or similar users, have been interested in). As to (ii), first-class citizens of the IAM are *enhanced cubes*, defined as multidimensional cubes coupled with *highlights*, i.e., sets of cube cells associated with interesting components of *models* automatically extracted from cubes [275]. Each operator is applied to an enhanced

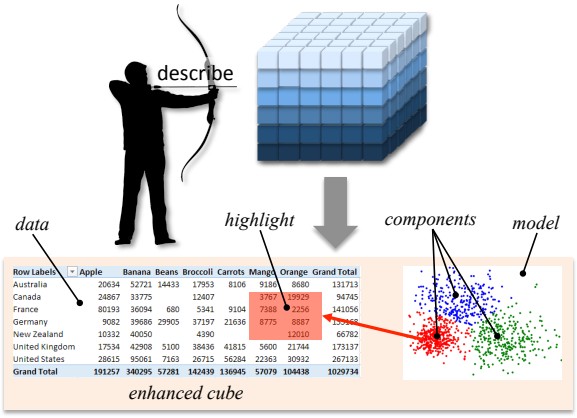


Figure 11.1: The IAM approach

cube and returns a new enhanced cube. To assess interestingness of model components, a measure based on their (objective) significance and (subjective) surprise is proposed.

An overview of the approach is given in Figure 11.1. Noticeably, having different models automatically computed and evaluated in terms of their interestingness relieves the user from the time-wasting effort of trying different possibilities.

Example 27 *Let a* SALES *cube be given, and let the user's intention be*

with SALES describe quantity for month = '1997-04' by type using clustering size 3

First, the subset of cells for April 1997 are selected from the SALES *cube, aggregated by product type, and projected on measure* quantity *(in OLAP terms, a slice-and-dice and a roll-up operator are applied). Then, these cells are clustered into 3 clusters based on the values of* quantity. *Finally, a measure of interestingness is computed for each cluster, and the cells belonging to the cluster with maximum interestingness are highlighted in the results shown to the user.* □

Clearly, the IAM vision raises a number of research challenges, e.g., (i) investigate if there are any other intention operators that should be considered besides the basic ones proposed, and how different operators can be combined; (ii) find techniques for automatically tuning the algorithms that create enhanced cubes by computing models; (iii) enrich the IAM framework with an approach to select the most effective chart or graph for visualizing each cube depending on its features such as number of dimensions, size, etc.; (iv) devise a visual metaphor for displaying enhanced cubes and interacting with them.

The goal of this chapter is to provide a proof-of-concept for the IAM vision by delivering an end-to-end implementation of the describe operator. Specifically, we address challenges

(ii) by experimenting two techniques to automatically set the number of model components. We also address challenges (iii) and (iv) by proposing a visualization that couples text-based representations and selected graphical representations with a component-driven interaction paradigm. This will save the time it would take to the user to try different visualizations; besides, by automatically selecting the most suitable charts based on the features of each cube, we discourage the user from adopting inappropriate visualizations which might lead her to wrong interpretations of data.

The chapter outline is as follows. After introducing a formalism to manipulate cubes and queries in Section 11.2, in Section 11.3 we introduce models, components, and enhanced cubes, and define an interestingness measure. Then, in Section 11.4 we show how an intention is transformed into an execution plan, in Section 11.5 we discuss how to automatically set the model size, i.e., its number of components, and in Section 11.6 we explain how enhanced cubes are visualized. Finally, in Section 11.7 we discuss the related literature, while in Section 11.8 we draw the conclusion.

11.2 Formalities

With respect to the formal background introduced in Section 8.3, to simplify the formalization, we restrict to consider linear hierarchies (the presence of branches in hierarchies makes Definition 30 overly complex).

Example 28 *Given the definition of cube schema (Definition 14), for our working example it is* $\mathsf{SALES} = (H, M)$ *where*

$$
\begin{aligned}
&H = \{h_{\mathsf{Date}}, h_{\mathsf{Customer}}, h_{\mathsf{Product}}, h_{\mathsf{Store}}\},\\
&M = \{\mathsf{quantity}, \mathsf{storeSales}, \mathsf{storeCost}\},\\
&\mathsf{date} \succeq \mathsf{month} \succeq \mathsf{year},\\
&\mathsf{customer} \succeq \mathsf{gender},\\
&\mathsf{product} \succeq \mathsf{type} \succeq \mathsf{category},\\
&\mathsf{store} \succeq \mathsf{city} \succeq \mathsf{country}
\end{aligned}
$$

and $op(\mathsf{quantity}) = op(\mathsf{storeSales}) = op(\mathsf{storeCost}) = \mathtt{sum}$. □

Definition 28 (Coordinate) *Given cube schema* $\mathscr{C} = (H, M)$ *(Definition 14) and a* group-by set *G of $\mathscr{C}$ (Definition 13), a* coordinate *of group-by set G is a tuple of members, one for each level of G. Given coordinate χ of group-by set G and another group-by set G' such*

that $G \succeq_H G'$, we will denote with $rup_{G'}(\chi)$ the coordinate of G' whose members are related to the corresponding members of χ in the part-of orders, and we will say that χ roll-up*s to $rup_{G'}(\chi)$. By definition, $rup_G(\chi) = \chi$.*

Example 29 *Two group-by sets of* SALES *are $G_1 = \{\mathsf{date}, \mathsf{type}, \mathsf{country}\}$ and $G_2 = \{\mathsf{month}, \mathsf{category}\}$, where $G_1 \succeq_H G_2$. G_1 aggregates sales by date, product type, and store country (for all customers), G_2 by month and category. Example of coordinates of the two group-by sets are, respectively, $\chi_1 = \langle$1997-04-15, Fresh Fruit, Italy$\rangle$ and $\chi_2 = \langle$1997-04, Fruit$\rangle$, where $rup_{G_2}(\chi_1) = \chi_2$.* □

We recall that a cube whose group-by set G_c includes all and only the dimensions of the hierarchies in H and such that $M_c = M$, is called a *base cube*, the others are called *derived cubes*. In OLAP terms, a derived cube is the result of either a roll-up, a slice-and-dice, or a projection made over a base cube.

Example 30 *The cube query over* SALES *used in Example 27 is $q = (G_q, P_q, M_q)$ where $G_q = \{\mathsf{type}\}$, $P_q = \{\mathsf{month} = \text{'1997-04'}\}$, and $M_q = \{\mathsf{quantity}\}$. A cell of the resulting cube is $\langle$*Canned Fruit$\rangle$ *with associated value 138 for* quantity. □

11.3 Enhancing cubes with models

Models are concise, information-rich knowledge artifacts [267] that represent relationships hiding in the cube cells. The possible models range from simple functions and measure correlations to more elaborate techniques such as decision trees, clusterings, etc. A model is bound to (i.e., is computed over the levels/measures of) one cube, and is made of a set of components (e.g., a clustering model is made of a set of clusters). In the IAM, a relevant role is taken by data-to-model mappings. Indeed, a model partitions the cube on which it is computed into two or more subsets of cells, one for each component (e.g., the subsets of cells belonging to each cluster).

Definition 29 (Model and Component) *A* model *is a tuple $\mathscr{M} = (t, alg, c, In, Out, \mu)$ where:*

1. *t is the model type;*
2. *alg is the algorithm used to compute $\mathscr{M}$*
3. *c is the cube to which $\mathscr{M}$ is bound;*

4. *In is the tuple of levels/measures of $\mathscr{C}$ and parameter values supplied to alg to compute $\mathscr{M}$;*

5. *Out is the set of components that make up $\mathscr{M}$;*

6. *μ is a function mapping each cell of c to one component of Out.*

Each model component *is a tuple of a component identifier plus a variable number of properties that describe that component.*

In the scope of this work, it is $t \in \{\text{top-k}, \text{bottom-k}, \text{skyline}, \text{outliers}, \text{clustering}\}$. The components for these model types are as follows:

1. For $t =$ top-k, there are two components: one for top-k cells, one for the others (similarly for bottom-k). Each component is described by the average z-score of its cells.

2. For $t =$ skyline, there are two components: one for the cells in the skyline, one for the others.

3. For $t =$ outliers, there are two components: one for outlier cells, one for the others. Each component is described by its outlierness.

4. For $t =$ clustering, there is one component for each cluster. Each component is described by the centroid of the corresponding cluster.

Example 31 *A possible model over the* SALES *cube is characterized by*

$$t = \textit{clustering},\ \textit{alg} = \textit{K-Means},\ c = \mathsf{SALES},$$
$$\textit{In} = \langle \mathsf{quantity}, n = 5, \textit{rndSeed} = 0 \rangle,\ \textit{Out} = \{c1, \ldots, c5\},$$
$$\mu(\langle \textit{Bagels} \rangle) = \mu(\langle \textit{Batteries} \rangle) = \mu(\langle \textit{Wine} \rangle) = \mu(\langle \textit{Beer} \rangle) = c1;$$
$$\mu(\langle \textit{Canned Fruit} \rangle) = \mu(\langle \textit{Muffins} \rangle) = \mu(\langle \textit{Fresh Fruit} \rangle) = c2;\ \ldots$$

where n is the desired number of clusters and rndSeed is the seed to be used by the k-means algorithm to randomly generate the 5 seed clusters. Component $c1$ is characterized by property centroid with value 151.3. □

As the last step in the IAM approach, the cube is enhanced by associating it with a highlight, i.e., with the subset of cells corresponding to the most interesting component of the model; these cells are determined via function μ. The measure proposed in [275] to assess

the interestingness of each component is based on the idea of *prior belief* [27]: specifically, it defines the *surprise* of a component as the difference of belief for corresponding cells in the cube before (c) and after (c') the application of the intention. This requires first of all to define the concept of "corresponding cell(s)" in c of each cell χ' of c', which is done through a *proxy* function. To simplify this function, we will assume that the intention does not change both the cube group-by set and its selection predicate. Intuitively, if the intention changes the group-by set, the corresponding cell(s) of χ' are determined via the part-of order; if the intention changes the selection predicate, the corresponding cell of χ' is χ' itself if it is part of c, the whole cube c otherwise.

Definition 30 (Proxy Cells) *Let c and c' be two cubes over cube schema $\mathscr{C}$, and let q and q' the queries producing c and c', respectively. Let χ' be a coordinate of c'. The* proxy cell*s of χ' over c are defined as*

$$proxy(\chi') = \begin{cases} \{\chi : rup_{G_{q'}}(\chi) = \chi'\}, \text{ if } G_q \succ_H G_{q'} \\ \{rup_{G_q}(\chi')\}, \text{ if } G_{q'} \succ_H G_q \end{cases} \text{, if } G_{q'} \neq G_q \ (P_{q'} = P_q);$$

$$proxy(\chi') = \begin{cases} \{\chi'\}, \text{ if } \chi' \in c \\ \{\chi : \chi \in c\}, \text{ if } \chi' \notin c \end{cases} \text{, if } P_{q'} \neq P_q \ (G_{q'} = G_q);$$

$$proxy(\chi') = \{\chi'\}, \text{ otherwise.}$$

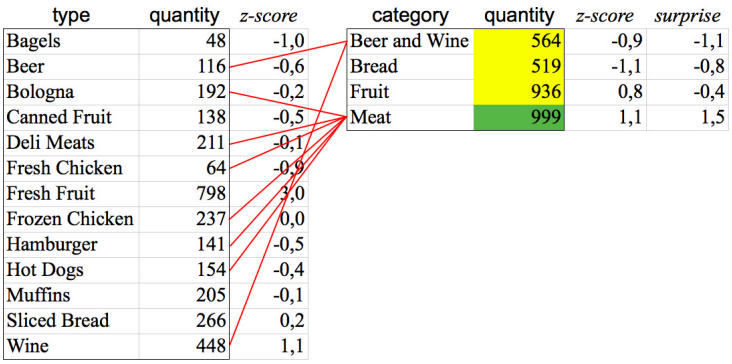
For the first intention, the starting cube c is the base cube; since in this case the user has no prior belief, we put $proxy(\chi') = \varnothing$ for all $\chi' \in c'$.

Definition 31 (Interestingness) *Let $\mathscr{M} = (t, alg, c', In, Out, \mu)$ be a model and $c \in Out$ be one of its components. The* interestingness *of c is defined as*

$$int(c) = avg_{\chi' \in \mu^{-1}(c)}\{surprise(\chi')\}, \text{ where}$$
$$surprise(\chi') = max_{m \in c'}\{z_m(\chi') - avg_{\chi \in proxy(\chi')}\{z_m(\chi)\}\}$$

and function $z_m()$ returns the z-score of a cell for measure m over the whole cube that cell belongs to.

Definition 32 *An* enhanced cube *E is a triple of a cube c, a set of models $\{\mathscr{M}_1, \ldots, \mathscr{M}_r\}$ bound to c, and a highlight $c_{high} = argmax_{\{c \in \bigcup_{i=1}^{r} Out_i\}}(int(c))$.*



type	quantity	*z-score*
Bagels	48	-1,0
Beer	116	-0,6
Bologna	192	-0,2
Canned Fruit	138	-0,5
Deli Meats	211	-0,1
Fresh Chicken	64	-0,9
Fresh Fruit	798	3,0
Frozen Chicken	237	0,0
Hamburger	141	-0,5
Hot Dogs	154	-0,4
Muffins	205	-0,1
Sliced Bread	266	0,2
Wine	448	1,1

category	quantity	*z-score*	*surprise*
Beer and Wine	564	-0,9	-1,1
Bread	519	-1,1	-0,8
Fruit	936	0,8	-0,4
Meat	999	1,1	1,5

Figure 11.2: Cubes c (left) and c' (right) in Example 32; in green and yellow two components for the top-k model, in red the proxy relationship for two cells in c'

Example 32 *Let*

with SALES describe quantity for month = '1997-04' by type

with SALES describe quantity for month = '1997-04' by category

be a sequence of two intentions formulated by the user. While the plan generated for the first intention relies on query q defined in Example 30, the one for the second intention relies on $q' = (G_{q'}, P_{q'}, M_{q'})$ where $G_{q'} = \{\text{category}\}$, $P_{q'} = \{\text{month} = \text{'1997-04'}\}$, and $M_{q'} = \{\text{quantity}\}$. Let c and c' be the cubes resulting from q and q', respectively. In practice, q' corresponds to a roll-up of q, so the proxy cells of each cell in c' are determined using the rup operator as shown in Figure 11.2 (red lines). The figure also shows the z-score for each cell of c and c', and the surprise of each cell of c'. Then, let $\mathscr{M}$ be the model of type top-k, with $k = 1$, computed on c'; this model has two components: c_1, including only the top-1 cell, and c_2, including all the others (shown in green and yellow in the figure). The interestingness values for these two components are $int(c_1) = 1.5$ and $int(c_2) = 0.8$, respectively. So, the enhanced cube resulting from the second intention includes c', $\mathscr{M}$, and the highlight c_1. □

11.4 Execution plans for describe intentions

The describe operator provides an answer to the user asking "show me my business" by describing one or more cube measures, possibly focused on one or more level members, at some given granularity [275]. The cube is enhanced by showing either the top/bottom-k cells, the skyline, the outliers, or clusters of cells.

The basic idea we followed to define the syntax for describe is that the user will work in sessions, similarly to the OLAP paradigm. As a consequence, the group-by set (by clause)

and the selection predicate (for clause) of each intention are formulated as increments over those of the previous intention.

Let c be a cube returned by executing query $q = (G_q, P_q, M_q)$ over base cube c_0 having cube schema $\mathscr{C} = (H, M)$. The general syntax for describe is

$$\text{with } c \text{ describe } m_1, \ldots, m_z \text{ [for } P' \text{] [by } l' \text{] [using } t_1, \ldots, t_r \text{] [size } k \text{]}$$

where $m_1, \ldots, m_z \in M$ are measures of $\mathscr{C}$, P' is a set of selection predicates each over one level of H, l' is a level of H, $t_1, \ldots, t_r$ are model types, and k is the desired size to be applied to all the models returned as explained in point 2 below (optional parts of the syntax are in brackets).

Now let l be the level in G_q belonging to the same hierarchy of l', and let $P \subseteq P_q$ be the set of predicates in P_q expressed over levels belonging to the same hierarchies than those over which the predicate in P' are expressed. The plan corresponding to a fully-specified intention with c, i.e., one where all optional clauses have been specified, is:

1. Execute query $q' = (G_{q'}, P_{q'}, \{m_1, \ldots, m_z\})$, where $G_{q'} = G_q \setminus \{l\} \cup \{l'\}$ and $P_{q'} = P_q \setminus P \cup P'$. Let c' be the cube resulting from the execution of q' over c_0.
2. For $1 \leq i \leq r$, compute model $\mathscr{M}_i = (t_i, alg_i, c', In_i, Out_i, \mu_i)$ and for each $c \in Out_i$, compute $int(c)$. Size k is used for clustering to determine the number of clusters to be computed, for top-k and bottom-k to determine the number of cells to be returned, for outliers to determine the number of outliers; it is neglected for the skyline.
3. Find the highlight $c_{high} = argmax_{\{c \in \bigcup_i Out_i\}}(int(c))$.
4. Return the enhanced cube consisting of c', $\{\mathscr{M}_1, \ldots \mathscr{M}_r\}$, and highlight c_{high}.

Partially-specified intentions are interpreted as follows:

- If the for P' clause has not been specified, we consider $P_{q'} = P_q$.
- If the by l' clause has not been specified, we consider $G_{q'} = G_q$.
- If the using $t_1, \ldots, t_r$ clause has not been specified, all model types listed in Section 11.3 are computed over c' (the skyline is computed only if $z > 1$, i.e., at least two measures have been specified).
- If the size k clause has not been specified, the value of k is determined automatically as discussed in Section 11.5.

When the first intention is issued on a cube, to constrain the size of the data to be returned it is required that either the by or the for clauses are specified at least.

Example 33 *Consider the following session on the* SALES *cube:*

with SALES describe quantity for month = '1997-04' by type
with SALES describe quantity by category using clustering size 3
with SALES describe quantity, storeSales for country = 'Italy' using skyline

The queries corresponding to the first two intentions are q in Example 32 and q' in Example 32, respectively. The query corresponding to the third intention is q'' characterized by $G_{q''} = \{\mathsf{category}\}$, $P_{q''} = \{\mathsf{month} = \text{'1997-04'}, \mathsf{country} = \text{'Italy'}\}$, and $M_{q''} = \{\mathsf{quantity}, \mathsf{storeSales}\}$. The models computed for the first intention are top-k, bottom-k, clustering, and outliers (computing the skyline for a single measure makes no sense). For the second and the third intentions, a clustering producing 3 clusters and the skyline are computed, respectively. □

11.5 Setting the model size

Our approach to find the best value for the size parameter k when it is not specified in the intention is based on good practices in hierarchical clustering, especially when single-linkage is used, meaning that inter-cluster distance is measured by the closest two points of the clusters. The best separation of clusters can then be found by finding the knee of the evaluation graph of the clustering algorithm, We tested two solutions from the literature, L-method [241] and Kneedle [246]. Since Kneedle is quicker and provides more consistent results, we have adopted it to determine k, both for clustering (k being the number of clusters), top/bottom-k (where k is the number of points in the first cluster, i.e., the one with higher values) and outliers (where k is the number of points in the first and last cluster).

11.6 Visualizing enhanced cubes

In this section we discuss how to provide an effective description of an enhanced cube by coupling text-based representations (a pivot table and a ranked component list) and graphical representations (one or more charts) with an ad-hoc interaction paradigm. The guidelines we adopt to this end are explained below:

1. For visualization purposes, we assume that an intention can select at most three measures ($1 \leq z \leq 3$) and three group-by levels ($1 \leq n \leq 3$). This is actually not a strong limitation, considering that a visualization of four or more dimensions and/or measures using a single table or chart is hardly feasible and definitely not intuitive.

2. Since we are focusing on intentions aimed at *describing* data, we believe that providing multiple visualizations from different points of view should be preferred to just picking the "most effective one". Indeed, the effectiveness of a visualization type largely depends on the skills and personal tastes of each user.

3. We restrict to considering visualization types that can be easily understood both by lay users and skilled users, and are suitable for multidimensional data.

4. Clearly, the effectiveness of a visualization type also depends on the features of the specific dataset. Using an unsuitable visualization can generate confusion and misunderstandings in users, and can lead them to wrong conclusions. Thus, for each intention we visualize only the charts that are recognized to be suitable given the characteristics of the data to be shown.

5. Models and components play a key role in the IAM approach. Thus, the visualizations we provide aims at showing not only dimension and measure values, but also the different components of a model using a color code. For the same reason, the interaction paradigm should be component-driven.

The visualization we provide for enhanced cube E based on guidelines (ii) and (v) includes three distinct but inter-related areas: a *table* area that shows the cube cells using a pivot table; a *chart* area that complements the table area by representing the cube cells through one or more charts; a *component* area that shows a list of model components sorted by their interestingness. The chart types we consider following guidelines (i) and (iii) are multiple line graphs, grouped column charts, heat maps, bubble graphs, and scatter plots. The heuristics we adopt to decide whether using or not each chart type for a given enhanced cube E (guideline (iv)) was inspired by [114]. The features of E we take into account to this end are the number n of dimensions, the number z of measures, and the domain cardinality and type of the dimensions. The pseudocode is shown in Algorithm 8; an example of heuristics we use is: if $n = 2$ and $z = 1$, draw a bubble graph using the X and Y axes for the two dimensions, the bubble size for the measure, and the bubble color for the model components.

The interaction paradigm we adopt is component-driven (guideline (v)). Specifically, clicking on one component c in the component area leads to emphasizing the corresponding

Algorithm 8 Chart area creation

INPUT $D = \{d_1, \ldots, d_n\}$: sets of dimensions, $M = \{m_1, \ldots, m_z\}$: set of measures, $T = \langle t_1, \ldots, t_r \rangle$: list of models sorted by interestingness

1: **if** $n = 1$ and $isDate(d_1)$ **then** ▷ Visualize multiple line graph
2: $MultipleLineGraph(X : d_1, Y_1 : m_1, \ldots, Y_z : m_z)$
3: **if** $z = 1$ and $card(d_1) \leq 50$ **then** ▷ Visualize grouped column chart
4: **if** $n = 1$ **then**
5: $GroupedColumnChart(X : d_1, height : m_1, color : t_1)$
6: **else**
7: **if** $n = 2$ and $card(d_2) \leq 8$ **then**
8: $GroupedColumnChart(X : d_1, height : m_1, color : d_2)$
9: **if** $n = 2$ and $z = 1$ **then** ▷ Visualize heat map
10: $HeatMap(X : d_1, Y : d_2, color : m_1)$
11: **else**
12: **if** $n = 1$ **then**
13: $HeatMap(X : d_1, Y_1 : m_1, \ldots, Y_z : m_z)$
14: **if** $z = 1$ **then** ▷ Visualize bubble chart
15: **switch** n **do**
16: **case** 2
17: $BubbleChart(X : d_1, Y : d_2, size : m_1, color : t_1)$
18: **case** 3
19: $BubbleChart(X : d_1, Y : d_2, Z : d_3, size : m_1, color : t_1)$
20: **switch** z **do** ▷ Visualize scatter plot
21: **case** 2
22: $ScatterPlot(X : m_1, Y : m_2, color : t_1)$
23: **case** 3
24: $ScatterPlot(X : m_1, Y : m_2, Z : m_3, color : t_1)$

cube cells (i.e., those that map to c via function μ) both in the table area and in the chart area. The highlight is the top component in the list and is selected by default.

Example 34 *Figure 11.3 shows the visualization obtained when a session of the two following intentions is formulated:*

with SALES describe storeCost by month
with SALES describe storeCost by category

On the top left, the table area; on the right, the chart area; on the bottom left, the component area (the three areas have been repositioned in the figure to save space). Here it is $n = 2$ and $z = 1$, so a heat map and a bubble chart have been selected. The top-interestingness

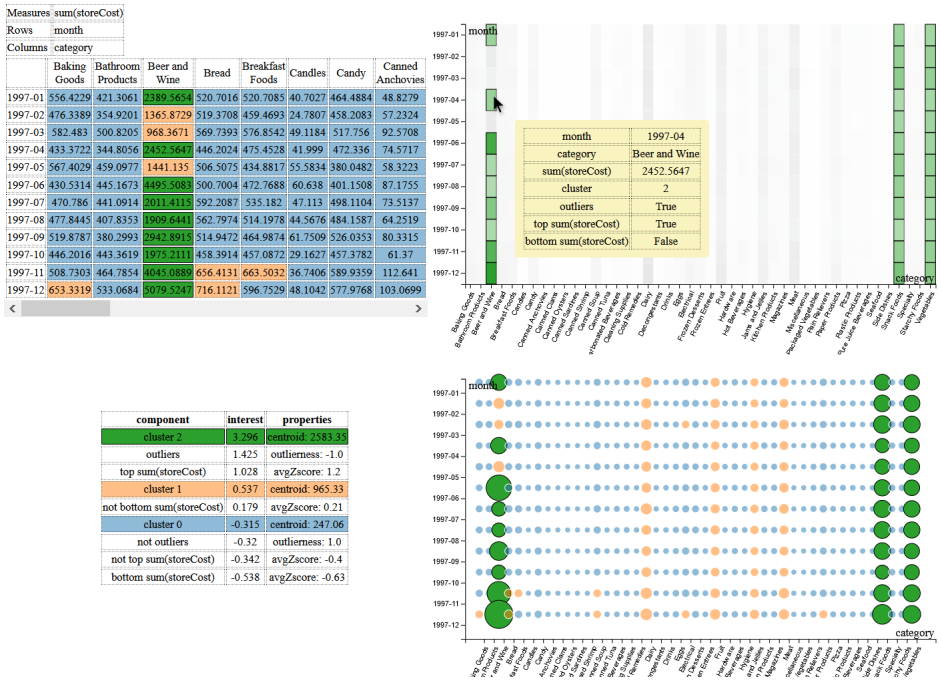


Figure 11.3: The visualization obtained for the session in Example 34

component is a cluster, so a color has been assigned to each component of clustering (i.e., to each cluster) and is uniformly used in all three areas. The highlight (in green) is currently selected and is emphasized using a thicker border in all areas. Note that a tooltip with all the details about a single cell is also shown (in yellow).

11.7 Related works

The idea of coupling data and analytical models was born in the 90's with inductive databases, where data were coupled with patterns meant as generalizations of the data [229]. Later on, data-to-model unification was addressed in MauveDB [66], which provides a language for specifying model-based views of data using common statistical models. However, achieving a unified view of data and models was still seen as a research challenge in business intelligence a few years later [221]. More recently, Northstar [159] has been proposed as a system to support interactive data science by enabling users to switch between data exploration and model building, adopting a real-time strategy for hyper-parameter tuning. Finally, the coupling of data and models is at the core of the IAM vision [275], on which this chapter relies. The three basic pillars of IAM are (i) the redefinition of query as expressing the user's intention rather than explicitly declaring what data are to be retrieved, (ii) the extension of

query results from plain data cubes to cubes enhanced with models and highlights, and (iii) the characterization of model components in terms of their interestingness to users.

The coupling of the OLAP paradigm and data mining to create an approach where concise patterns are extracted from multidimensional data for user's evaluation, was the goal of some approaches commonly labeled as OLAM [127]. In this context, k-means clustering is used in [25] to dynamically create semantically-rich aggregates of facts other than those statically provided by dimension hierarchies. Similarly, the shrink operator is proposed in [111] to compute small-size approximations of a cube via agglomerative clustering. Other operators that enrich data with knowledge extraction results are DIFF [243], which returns a set of tuples that most successfully describe the difference of values between two cells of a cube, and RELAX [245], which verifies whether a pattern observed at a certain level of detail is also present at a coarser level of detail, too. Finally, in [45] the OLAP paradigm is reused to explore prediction cubes, i.e., cubes where each cell summarizes a predictive model trained on the data corresponding to that cell. The IAM approach can be regarded as OLAM since, like the approaches mentioned above, it relies on mining techniques to enhance the cube resulting from an OLAP query. However, while each of the approaches above uses one single technique (e.g., clustering) to this end, the IAM leans on multiple mining techniques to give users a wider variety of insights, using the interestingness measure to select the most relevant ones.

In the same direction, in [244] the authors describe a method that profiles the exploration of a user and uses the Maximum Entropy principle to recommend which unvisited parts of the cube can be the most surprising in a subsequent query. The Cinecubes method [107, 108] aims at providing automated reporting as a result of an original OLAP query. The proposed method enriches an original OLAP query with auxiliary queries to aid (a) the comparison and assessment of the result of the query to similar data and (b) the explanation of the result with values at the most detailed level. So, the results of the Cinecubes system can coarsely be grouped as the result of two operators: the first one computes queries for values similar to ones defining the selection filters of the original query; the second one by drilling down into the dimensions of the result, one dimension at a time.

The characteristics of the different approaches for visualizing data and interacting with them have been deeply explored in the literature, also with reference to their suitability for datasets with different features and users with varying skills and goals. In [32], the author surveys the classifications proposed in the literature for visualization types and integrates them into a single comprehensive framework. Abela [3] proposes a decision tree to select the best visualization according to the user's goal and to the main features of data. More recently, SkyViz —to which our approach is inspired— starts from a visualization context based on

Table 11.1: Execution time in seconds for increasing cube cardinalities

Cardinality	Query	Model	Interestingness	Pivot	Total
323	0.88	1.45	0.03	0.03	2.39
77832	0.64	3.61	0.39	0.51	5.14
86829	0.69	3.66	0.48	1.56	6.38

seven coordinates for assessing the user's objectives and describing the data to be visualized. Then it uses skyline-based techniques to translate a visualization context into a set of suitable visualization types and to find the best bindings between the columns of the dataset and the graphical coordinates used by each visualization type.

11.8 Evaluation and conclusion

In this chapter, we have proposed a visual metaphor to display enhanced cubes for describe intentions according to the IAM vision. The prototype implementation can be accessed at http://semantic.csr.unibo.it/describe/. It uses the simple multidimensional engine described in [88], which in turn relies on the MySQL DBMS to execute queries on a star schema based on multidimensional metadata (in principle, the prototype could work on top of any other multidimensional engine). The mining models are imported from the Scikit-Learn Python library. Finally, the web-based visualization is implemented in JavaScript and exploits the D3 library for chart visualization.

To evaluate the feasibility of our approach from the computational point of view, we made some preliminary performance tests; to this end we populated the SALES cube using the FoodMart data (https://github.com/julianhyde/foodmart-data-mysql). We considered the worst case, in which all five models are computed on cubes obtained by progressively including in the group-by set the three dimensions with highest cardinality. The tests were run on an Intel(R) Core(TM)i7-6700 CPU@3.40GHz CPU with 8GB RAM. Table 11.1 shows the time necessary to query the base cube, to compute the models, to measure the interestingness, and to generate the pivot table returned to the browser. Remarkably, it turns out that at most 6.38 seconds are necessary to retrieve and visualize an enhanced cube of 86829 cells.

To evaluate the formulation effort without the intentional model and the describe intention, we asked five Ph.D. students in computer science to use Python to manually extract insights from a 2000 tuples bidimensional cube using only two types of models (outliers and clustering). This real-world cube was created from the COVID dataset made available by the

Table 11.2: Time (minutes) and formulation effort (numbers of characters for manual model extraction)

Student Id	Skill	Time	Models	Length
A	advanced	45	clustering	3479
B	advanced	51	both	1777
C	intermediate	25	outliers	935
D	advanced	59	both	1150
E	advanced	90	outliers	2627

European Center for Disease Prevention and Control[1]. Table 11.2 shows, for each student, her skill in Python (beginner/intermediate/advanced), the time taken for doing the exercise (in minutes), the models she extracted, and the ASCII character length of the Python code she wrote, disregarding the quality of the models extracted. We remark that even skilled students needed quite a long time for extracting both models, and had to write substantial Python programs.

Finally, the main directions for future research we wish to pursue are: (i) evaluate the effectiveness of the approach by experimenting it with real users; (ii) extend the approach to the other intention operators and to operate with *dashboards* of enhanced cubes; and (iii) experiment other interestingness metrics [43].

[1] www.ecdc.europa.eu/en/geographical-distribution-2019-ncov-cases

Chapter 12

Multidimensional Summarization: Itemsets as a Case Study

12.1 Introduction

Frequent itemset (FI) mining is an unsupervised mining technique used in descriptive analytics to uncover frequent correlations in (possibly huge) transactional datasets [9]. In its initial formulation for market basket analysis, items correspond to products and a transaction corresponds to a set of products bought together by a customer; a set of items that appear together in many transactions is said to be frequent. More recently FI mining has been applied to other contexts, where items are not necessarily homogeneous (as is the case for plain products) but, aimed at providing a richer description of subjects and events, they are modeled as multi-dimensional and multi-level objects. In a *multi-dimensional* item [129, 40] an event is described by multiple features (e.g., product and customer age); an item is *multi-level* [129, 22] if it can be described using a hierarchy with different levels of details (e.g., products and product types). The mining of multi-dimensional and multi-level items is now well understood and extends traditional FI mining with the possibility of aggregating infrequent specific itemsets into frequent generic itemsets [129, 22].

The exponential nature of FIs —n items may lead to 2^n FIs— makes it difficult for analysts to explore their information content. Figure 12.1 gives a qualitative picture of the containment lattice for itemsets [129]; itemsets in the upper part of the lattice appear more frequently together in transactions. To effectively explore the lattice, data analysts normally define a frequency threshold (or even a frequency range) to cut irrelevant itemsets. Setting a low threshold (Figure 12.1(a)) gives a broad picture of the correlations, but the FIs selected are too many for a well-focused analysis. On the other hand, setting a high

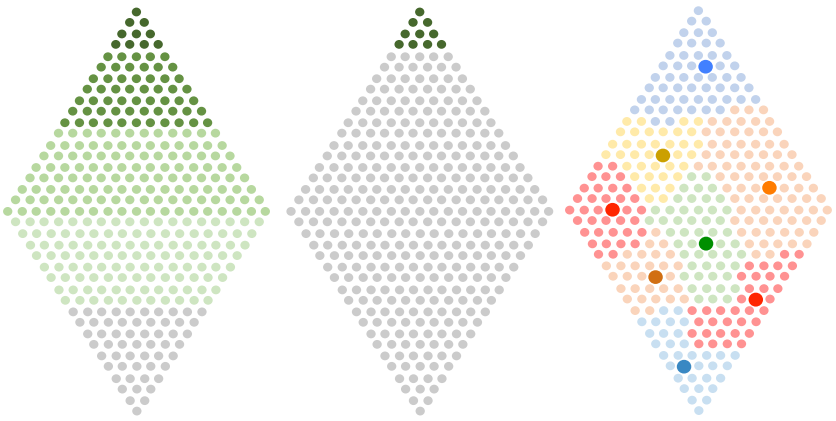


Figure 12.1: Itemset lattice, with dark itemsets being more frequent: FI mining with a low frequency threshold (a) and a high frequency threshold (b); summarization, with big dots corresponding to cluster representatives

threshold (Figure 12.1(b)) gives a strong focus on the most frequent FIs but may lead to missing interesting correlations. Besides, in both cases, the FIs selected are very similar to each other, so most of them do not really give useful information to the analyst. In [306], the authors highlight the need for *summaries* to overcome these problems. Summarization selects a minimal subset of representative FIs that describe the entire population while maximizing the diversity of these representatives (Figure 12.1(c)). Some approaches to summarization have been proposed in the past (e.g., [40, 223]), however most of them do not consider the multi-level and multi-dimensional natures of FIs and do not provide user-friendly ways to explore summaries.

In this chapter, we propose *SUSHI* (*SUmmarization and viSualization of Hierarchical Itemsets*), a comprehensive framework for analyzing multi-level and multi-dimensional FIs based on original techniques for summarization and visual exploration. In SUSHI, summarization operates in synergy with FI mining to empower and simplify analysis: indeed, it makes the approach more robust with reference to the frequency threshold and enables both specific and general patterns to be discovered. On top of that, visual exploration unveils and highlights hidden information, supports the process of understanding and decision making, and allows analysts to focus their attention on what is important. Note that some visual representations for FIs have been previously proposed (e.g., [166, 33]); however, none of them enables an in-depth and multi-resolution navigation of summaries.

Our work is inspired by a compelling case study focused on customer profiling starting from the tracking of daily GPS trajectories. Each transaction describes a mall customer by means of several features (e.g., where she lives, where she works, how much she earns),

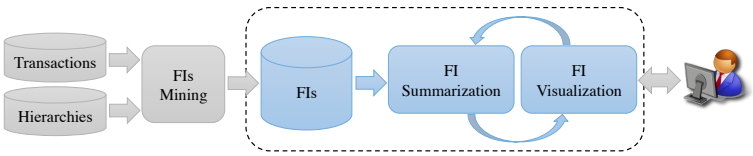


Figure 12.2: A functional architecture of SUSHI

and each feature is described at different, hierarchically-organized levels of details (e.g., she lives close to the Whitney Museum, which is located in the Greenwich Village district, which belongs to Manhattan). In this context, a FI describes the profile of a group of people sharing the same features/behavior. Multi-level and multi-dimensional FIs enable a rich representation of the behavior of customer groups; however, the huge number of mined FIs encourages the adoption of effective summarization and visualization techniques to provide decision makers with useful information for marketing and advertising.

As shown in Figure 12.2, SUSHI works independently of the algorithm applied for generating the FIs taken in input (e.g., Apriori [9], FP-Growth [130]). The summarization and the visualization components work jointly to show analysts the relevant information at multiple levels of detail; analysts iteratively create and visualize new summaries that better meet their needs by adjusting a set of parameters, and navigate them to get insights over summarized data.

In a preliminary paper [94] we have proposed an extension of the definition of containment to multi-level itemsets and a similarity function for FIs which takes into account both their extensional (support-based) and intensional (feature-based) natures. Relying on that background, we offer the following original contributions:

1. a clustering algorithm that uses this similarity function to create a hierarchical summary to shrink a set of FIs down to a set of relevant patterns; to define the representative FI of each cluster, we discuss different strategies;

2. theoretical results concerning antimonotonicity of support and similarity in multi-level settings; these results allow the reduction of the search space for the clustering algorithm;

3. an evaluation of the search space and of the computational complexity of the clustering algorithm;

4. two integrated approaches to summary visualization and exploration: a graph-based one, which highlights inter-cluster relationships, and a tree-based one, which emphasizes the relationships between the representative of each cluster and the other FIs in that cluster;

5. a comprehensive set of tests for evaluating the efficiency and effectiveness of our approach against others in the literature; we also made some tests with data science students to evaluate how well SUSHI enhances the analysts' capabilities in discovering relevant patterns.

The remainder of the chapter is organized as follows. After discussing the related approaches to summarization and visualization in Section 12.2, in Section 12.3 we provide a formal background for multi-level and multi-dimensional itemsets. Then, in Section 12.4 we propose a definition of itemset similarity. Section 12.5 describes our approach to building summaries of FIs, while Section 12.6 explains how we visualize them. Finally, Section 12.7 presents the results of the experimental tests and Section 12.8 concludes the chapter.

12.2 Related works

12.2.1 Mining and summarization of FIs

FI mining has been applied to multiple fields and types of items [128]. Items can be either binary (e.g., product purchases in market basket analysis) or categorical (e.g., IP addresses in a network table). Besides, they can be homogeneous (items from a single dimension at the same abstraction level, e.g., products) or multi-dimensional (items taken from multiple dimensions, e.g., products and customers), and they can be multi-level (items described at different levels of abstraction, e.g., products and product categories).

Since the application of the Apriori algorithm in market basket analysis [9], scholars defined many performant algorithms to extract FIs [180]; among them, FP-Growth [130] and very recently CBPM [70] and GMiner [52]. FI mining is currently witnessing a plethora of declinations [180], among them: high-utility itemsets (i.e., the extraction of FIs whose utility is higher than a given score [213]); colossal itemset mining (i.e., the extraction of FIs in datasets with few transactions containing even millions of items [273]); and frequent trajectory mining (i.e., the extraction of common routes in urban areas [299]). Given the possibility to mine FIs at different granularities [129], external knowledge can be used to generalize infrequent (specific) itemsets into frequent (generic) itemset [22]. For instance, two distinct products *milk* and *beer* can be grouped together into a generic FI only knowing that they both have type *beverage* [22]. While these algorithms address the mining of either flat or multi-dimensional and multi-level itemsets, they do not provide any form of summaries.

Since the number of FIs is exponential [9], summarization is necessary to support an effective data analysis [10]. Itemset summarization enables an effective explorations of large

datasets by replacing groups of similar FIs with their representatives. Classical approaches to summarization reduce the exploration space by limiting the retrieved FIs to maximal-covered [122], closed-covered [220], and δ-covered [263] FIs. However, these approaches do not consider the exponential cardinality of the summary, its pertinence with respect to the exploration focus, and the diversity of the FIs retrieved in the summary, which are indeed well-known driving criteria for data condensation [37]. To overcome these limitations, among the summarization techniques that have been well surveyed in [10], two approaches are well-known. In BUS [40], summarization exploits the definition of *itemset containment* (i.e., a FI can be summarized into another if the latter is a superset of the former) to uncover clusters of FIs by the greedy optimization of *compaction gain* and *information loss* metrics. BUS has been extended to MBUS [137] by the retrieval of *interesting* and *intelligible* itemset representatives. Further summarization techniques have been achieved by generalizing similar FIs belonging to sets [279, 8] or hyperrecatangles [283]; by computing an optimal number of probability vectors for cluster partitions [290, 223]; or by minimizing the restoration error [150]. In [307], the authors propose a summarization suitable only for high-utility itemsets; when itemsets are considered with equal utility, the summarization criterion boils down to the itemset containment used in [40, 137]. Overall, these contributions consider neither multi-dimensional nor multi-level FIs, nor they provide summary visualization and navigation. A schematic comparison between SUSHI and the summarization approaches for FIs is shown in Table 12.1.

As representative FIs are elected in place of a group of similar FIs, a key issue in summarization lies in the definition of itemset similarity. Similarity has been defined in terms of itemset intersection (e.g., [40, 137]) and support (i.e., the frequency of two itemsets within the dataset [284, 173]). However, these approaches *do not* exploit multi-level knowledge to aggregate FIs in intensionally-coherent groups.

From the computational point of view, summarization (seen as the problem of finding the minimal set approximating a given collection) is a NP-hard problem [8]. In SUSHI we adopt a greedy strategy based on a similarity function accounting for both the extensional and intensional natures of FIs. We also exploit the mathematical properties of our similarity function (specifically, its antimonotonicity) to sensibly prune the algorithm search space.

With respect to the above-mentioned contributions, in SUSHI we (i) introduce a novel similarity function based on the concepts of *relevance* and itemset containment, properly extended to deal with multi-dimensional and multi-level itemsets; (ii) introduce multiple strategies to shrink the exponential number of FIs to a meaningful *hierarchical* summary; (iii) provide a theoretical foundation to exploit the antimonotonicity of our similarity function to prune the algorithmic space; and (iv) propose an end-to-end framework supporting the

Table 12.1: Approaches to FI summarization classified by similarity function, representative cluster element, clustering type (E=Exclusive, O=Overlapping, T=Total, P=Partial), multi-dimensional (MD) and multi-level (ML) data, and presence of summary visualization

Appr.	Similarity function	Representative	Type	MD	ML	Vis
[8]	Maximum coverage	FI	E, P	✗	✗	✗
[290]	Kullback-Leibler divergence	Probab. profile	E, T	✗	✗	✗
[279]	Maximum coverage	FI	O, T	✗	✗	✗
[40]	Compaction, inf. loss	FI	E, T	✗	✗	✗
[137]	Compaction, inf. loss, interestingness	FI	E, T	✗	✗	✗
[150]	Restoration error	FI	E, T	✗	✗	✗
[223]	Compressible profile	Probab. profile	E, T	✗	✗	✗
[283]	Coverage cost function	Hyperrectangle	E, T	✗	✗	✗
[173]	Support	FI	E, T	✗	✗	✗
SUSHI	Support, relevance	FI	E, T	✓	✓	✓

navigation of the produced hierarchical summary. Finally, our contribution differs from approaches such as top-k (e.g., [316]) and minimum description length (i.e., compressing a dataset by exploiting regularities among data as in [186]), as our goal is not only to reduce and summarize the FIs but also to ensure an in-depth summary exploration. Furthermore, while a research branch close to FI mining is association rule mining, the summarization of association rules (e.g., [69]) relies on metrics (e.g., confidence, lift) that cannot be mapped to FIs.

12.2.2 Visual exploration

To be useful, a summary should give data analysts an overview of salient information and enable in-depth explorations of FI groups. This goal is not achievable by just displaying a plain list of FIs, nor by arranging all the available FIs in space. Visual techniques have high potential impact on dataset exploration [152], and should not require complex skills from decision makers.

Scholars have addressed the curse of FI and association rule cardinality by providing visualizations based on polylines [166], circular graph layout [33], treemaps [255], and parallel coordinates reflecting the item taxonomy [292]. The analyst is also included in the loop to visually mine FIs and association rules [171] by providing a visual representation of the algorithm search space and by enabling an interactive pruning. However, with increasing numbers of FIs, the proposed visualizations hardly scale up to human-understandable visualizations, and prevent analysts from perceiving itemset similarity at first sight.

In SUSHI, consistently with the graph visualization proposed in [252] and abiding by the "overview first, zoom and filter, then details-on-demand" mantra [252], we complement summarization with a hierarchical visualization enabling user-friendly and multi-resolution explorations of summaries.

12.3 Formal background

To define multi-level itemsets, we recall the notation of hierarchy from Definition 12. Such allows information to be provided at a level coarser than the one of dimensions. In practice, hierarchies can be incomplete, i.e., some levels may be missing in the part-of partial order. In this case, the techniques proposed in [113] can be used to balance hierarchies by filling the missing values.

The itemsets we consider are also multi-dimensional, i.e., they describe events along different features (e.g., worksIn). Each feature corresponds to a hierarchy (e.g., Location) and defines the semantics carried by an item at one specific level of that hierarchy.

Definition 33 (Domain Schema) *A* domain schema *is a triple* $\mathscr{D} = (H, \mathscr{E}, \mu)$ *where* H *is a set of hierarchies,* $\mathscr{E}$ *is a set of* features*, and* μ *is a function mapping each feature onto one hierarchy.*

Example 35 *Our working example relies on the* Profiling *domain schema, which describes the customers who regularly visit a mall and includes two hierarchies as shown in Figure 12.3. One hierarchy is rooted in the* Location *dimension and has two branches that describe locations from the geographical point of view and based on their characteristics, respectively. The roll-up partial order states, for instance, that* Neighborhood $\succeq_{\mathsf{Location}}$ Borough*; the part-of partial order states that* Harlem $\geq_{\mathsf{Location}}$ Manhattan*. The second hierarchy is rooted in the* Income *dimension and classifies incomes based on their ranges. The features of* Profiling *are* worksIn*,* frequents*, and* earns*; specifically* $\mu(\mathsf{worksIn}) = \mu(\mathsf{frequents}) = \mathsf{Location}$*,* $\mu(\mathsf{earns}) = \mathsf{Income}$*.*

We are now ready to define an item as a couple of a feature and a value taken from the domain of one of the levels of the corresponding hierarchy. Itemsets are non-redundant sets of items, i.e., two items in an itemset cannot be defined on values related in the part-of partial order (e.g., GreenwichVillage and Manhattan). Finally, transactions are itemsets whose items are all defined on dimension values, i.e., at the top level of a hierarchy (e.g., Macy's).

Definition 34 (Itemset and Transaction) *Given domain schema* $\mathscr{D} = (H, \mathscr{E}, \mu)$*, an* item *of* $\mathscr{D}$ *is a couple* $i = (f, v)$ *where* $f \in \mathscr{E}$*,* $v \in Dom(l)$*, and* l *is a level of hierarchy* $\mu(f)$*. An*

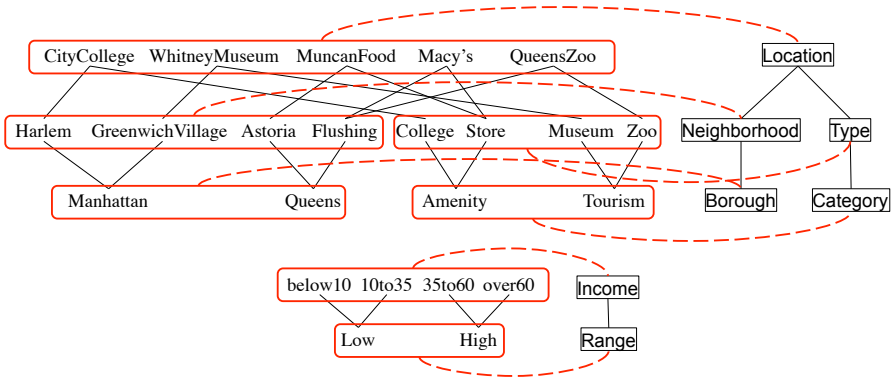


Figure 12.3: Hierarchies (right) and values (left) for the Profiling domain schema; in red, the level-domain mappings

itemset *I of* $\mathscr{D}$ *is a set of distinct items of* $\mathscr{D}$ *where, for each* $i, i' \in I$ *such that* $i = (f, v)$ *and* $i' = (f, v')$*, it is* $v \not\succeq_{\mu(f)} v'$ *and* $v' \not\succeq_{\mu(f)} v$*. We call a* transaction *an itemset only including items defined over dimensions of* H*.*

Example 36 *With respect to the* Profiling *multi-dimensional and multi-level domain, examples of an itemset* I *and a transaction* T *are:*

$$I = \{(\mathsf{worksIn}, \mathrm{Harlem}), (\mathsf{frequents}, \mathrm{Museum}), (\mathsf{frequents}, \mathrm{Macy's}), (\mathsf{earns}, \mathrm{High})\}$$
$$T = \{(\mathsf{worksIn}, \mathrm{CityCollege}), (\mathsf{frequents}, \mathrm{WhitneyMuseum}),$$
$$(\mathsf{frequents}, \mathrm{Macy's}), (\mathsf{earns}, 35\mathrm{to}60)\}$$

Note that, for semantic reasons, some features are actually disjunctive, meaning that they cannot realistically take two or more values at the same time; for instance, this is the case for earns. Others are not; for instance, a person can frequent several places. In practice it is not necessary to model this property of features, since real (correct) transactions will always have at most one single value for a disjunctive feature, so the same will hold for all FIs as well.

12.4 Working with itemsets

Working with multi-dimensional and multi-level items requires classic definitions related to FI mining to be extended. We start by defining a notion of *containment* between itemsets that generalizes set containment taking hierarchies into account; based on itemset containment we can then define the *support* of itemsets, which in turn is necessary to select FIs out of the set of all itemsets.

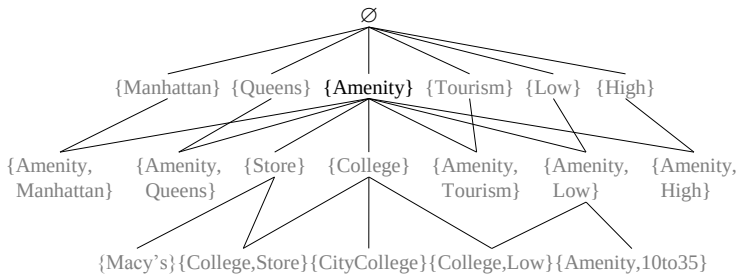


Figure 12.4: A portion of the containment lattice for the Profiling; itemsets in gray are partially expanded

Definition 35 ((Multi-Level) Itemset Containment) *Given two itemsets I and I', we say that I is* contained *in I' (denoted $I \sqsubseteq I'$) if for each item $i \in I$, $i = (f,v)$, there is an item $i' \in I'$, $i' = (f,v')$ such that $v' \geq_{\mu(f)} v$. Given a set $\mathscr{F}$ of itemsets, an itemset $I \in \mathscr{F}$ is* directly contained in $I' \in \mathscr{F}$ with reference to $\mathscr{F}$ *(denoted $I \dot{\sqsubseteq}_{\mathscr{F}} I'$) if there is no other itemset $I'' \in \mathscr{F}$ such that $I \sqsubseteq I'' \sqsubseteq I'$.*

Let $\mathscr{I}$ denote the set of all items of a schema domain and $2^{\mathscr{I}}$ denote the set of all itemsets as in Definition 34. The containment relationship is reflexive, antisymmetric, and transitive, and for each pair of itemsets in $\mathscr{I}$ there are a least upper bound and a greatest lower bound; so relationship $\sqsubseteq$ induces a lattice on $2^{\mathscr{I}}$, whose top and bottom elements are the empty itemset and $\mathscr{I}$, respectively. Given two itemsets I and I', we denote with $lub(I,I')$ and $glb(I,I')$ their least upper bound (i.e., the least element in $2^{\mathscr{I}}$ that is greater than or equal to both I and I') and greatest lower bound (i.e., the greatest element in $2^{\mathscr{I}}$ that is less than or equal to both I and I') in the lattice.

Example 37 *Figure 12.4 shows a portion of the containment lattice for the* Profiling *domain schema; for simplicity we focus on features* frequents *and* earns *and denote items by their value only. For instance, for* frequents *it is* $\{\text{Amenity}\} \sqsubseteq \{\text{College}, \text{Store}\}$ *and* $\{\text{Amenity}\} \dot{\sqsubseteq}_{2^{\mathscr{I}}} \{\text{College}\}$. *Intuitively,* $\{\text{Store}\}$ *and* $\{\text{College}\}$ *arise by describing* $\{\text{Amenity}\}$ *at a higher level of detail;* $\{\text{Amenity}, \text{Manhattan}\}$ *arises from a branch of the* Location *hierarchy; and* $\{\text{Amenity}, \text{Low}\}$ *arises by extending* $\{\text{Amenity}\}$ *with the additional item* $\{\text{Low}\}$.

We say that transaction T *supports* itemset I if $I \sqsubseteq T$. For instance, in Example 36, T supports I. Given a set of transactions $\mathscr{T}$, the set of transactions that support I is denoted by $\mathscr{T}_I \subseteq \mathscr{T}$. At this stage we can introduce a numerical property of itemsets, their *support*.

Definition 36 (Itemset Support) *Given itemset I, its support $sup(I)$ within a set of transactions $\mathscr{T}$ is defined as $sup(I) = |\mathscr{T}_I|/|\mathscr{T}|$.*

An itemset I is called *frequent* if its support is greater or equal to a given threshold. Since the containment relationship induces a lattice on the set $2^{\mathscr{I}}$ of all possible itemsets, it also induces a partial order over the set $\mathscr{F} \subseteq 2^{\mathscr{I}}$ of FIs. Thus, from now on we will say that $\mathscr{F}$ is a partially ordered set (POS).

A valuable property of the support for non-hierarchical itemsets is antimonotonicity along the containment relationship: for each I, I' such that $I \subseteq I'$ it clearly is $sup(I) \geq sup(I')$ [9]. Remarkably, this property also holds for our hierarchical itemsets along the multidimensional containment relationship introduced in Definition 35, as proved by the following theorem.

Theorem 1 ((Multi-Level) Antimonotonicity of Support) *Given two itemsets I and I' such that $I \sqsubseteq I'$, for all sets of transactions it is $\mathscr{T}_{I'} \subseteq \mathscr{T}_I$ and $sup(I) \geq sup(I')$.*

Proof: To prove that $\mathscr{T}_{I'} \subseteq \mathscr{T}_I$ it is sufficient to prove that for each transaction T that supports I', T also supports I (for Definition 36). For Definition 34, if T supports I' then for each item $i' \in I'$, $i' = (f, v')$, there is at least one item $i_0 \in T$, $i_0 = (f, v_0)$, such that $v_0 \geq_{\mu(f)} v'$. But since $I \sqsubseteq I'$, we know by Definition 35 that for each item $i \in I$, $i = (f, v)$, there is an item $i' \in I'$, $i' = (f, v')$ such that $v' \geq_{\mu(f)} v$. The part-of partial order $\geq_{\mu(f)}$ is transitive, so $v_0 \geq_{\mu(f)} v$; then we obtain that for each $i \in I$ there is at least one item $i_0 \in T$ such that $v_0 \geq_{\mu(f)} v'$, in other words that T supports I. From here, it immediately follows that $sup(I) \geq sup(I')$. □

Note that, due to this result, the transactions supporting I' also support I.

In Section 12.5 we will compute summaries using clustering, which clearly requires the definition of a similarity function to measure how "close" two FIs are. To this end, we start by introducing *feature-based* similarity, which increases with the number of features (i.e., semantics) shared by two FIs. Clearly, if two FIs include two distinct groups of transactions, feature-based similarity is not meaningful, so it is associated with a *support-based* similarity which increases with the percentage of transactions supporting both FIs. There is no obvious correlation between these two aspects of similarity; for instance, support-based similarity can be low even if feature-based similarity is high when non-shared features are rare and supported by a small fraction of transactions.

In a multi-level and multi-dimensional domain, defining feature-based similarity only by counting the common items between two FIs would be reductive; in fact, the informative value carried by these items in terms of level of detail should be considered as well. For instance, knowing that a person frequents Macy's is more relevant than knowing that she frequents a generic amenity. Intuitively, an FI is more *relevant* than another if it includes a larger number of distinct features; in turn, the relevance of a feature increases with the level of detail at which it is expressed.

Definition 37 (Itemset Relevance) *Given itemset I, its* relevance *is*

$$rel(I) = \sum_{f \in Feat(I)} \left(rel(f) + \sum_{l \in Lev_f(I)} rel(l) \right)$$

where $Feat(I)$ is the set of distinct features of the items in I, $Lev_f(I)$ is the set of levels of the values coupled with feature f in the items of I, $rel(f)$ is the relevance of f, and $rel(l)$ is the relevance of level l.

Assuming that higher levels of detail carry more informative content and relevance, means assuming that $rel(l) \geq rel(l')$ if $l \succeq_{\mu(f)} l'$. As a consequence, given any two itemsets such that $I \sqsubseteq I'$, we always have $rel(I') \geq rel(I)$.

We can now define the similarity between two itemsets as a linear combination of a support-based and a feature-based similarity.

Definition 38 (Itemset Similarity) *Given a set of transactions $\mathscr{T}$, a POS of FIs $\mathscr{F}$, two FIs I and I' supported by $\mathscr{T}$, and a coefficient $\lambda \in [0..1]$, the* similarity *of I and I' is defined as $sim(I,I') = \lambda sim_{sup}(I,I') + (1-\lambda) sim_{fea}(I,I')$, where*

$$sim_{sup}(I,I') = \begin{cases} \frac{sup(glb(I,I'))}{sup(I)+sup(I')-sup(glb(I,I'))}, \text{ if } glb(I,I') \in \mathscr{F} \\ 0, \text{ otherwise} \end{cases} \tag{12.1}$$

$$sim_{fea}(I,I') = \begin{cases} \frac{rel(lub(I,I'))}{rel(glb(I,I'))}, \text{ if } lub(I,I'), glb(I,I') \in \mathscr{F} \\ 0, \text{ otherwise} \end{cases} \tag{12.2}$$

Both sim_{sup} and sim_{fea} range in $[0..1]$; they can be explained as follows:

- sim_{sup} is the ratio between the number of transactions that support both I and I' and the number of transactions that support either I or I'. The higher the portion of transactions supporting both I and I', the higher sim_{sup} as requested by the the support-based principle.

- sim_{fea} is the ratio between the relevance of the $lub()$ and $glb()$ of the two FIs. When the features belonging to the two FIs I, I' are dissimilar, (i) the relevance of the $lub(I,I')$ will be low since most of the features in I, I' will be dropped or their level will be less detailed; (ii) the relevance of the $glb(I,I')$ will be high since most of the features in I' must be added to those in I. This behavior satisfies the feature-based principle.

Clearly, since the lub and glb operators are commutative, it is always $sim(I,I') = sim(I',I)$.

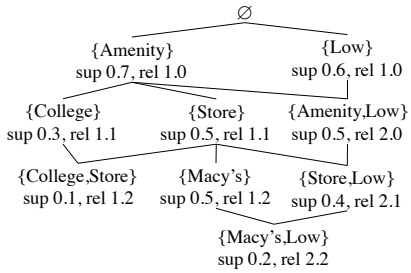


Figure 12.5: The POS $\mathscr{F}$ of FIs for Examples 38, 39 and 41 (each FI is annotated with its support and relevance)

Example 38 *With reference to the hierarchies of Figure 12.3, let the POS of FIs $\mathscr{F}$ be the one shown in Figure 12.5, restricted to features* frequents *and* earns*. We assume that (i) for each feature f it is $rel(f) = 1$, and (ii) relevance increases by 0.1 for each level of detail. Given FIs $I = \{(\mathsf{frequents}, \mathrm{Store})\}$ and $I' = \{(\mathsf{frequents}, \mathrm{Amenity}), (\mathsf{earns}, \mathrm{Low})\}$, it is $lub(I,I') = \{(\mathsf{frequents}, \mathrm{Amenity})\}$, $glb(I,I') = \{(\mathsf{frequents}, \mathrm{Store}), (\mathsf{earns}, \mathrm{Low})\}$, and $sim(I,I') = 0.57$.*

To decrease the complexity of summary creation, in Section 12.5 we will leverage on the properties of itemset similarity and containment. First of all, Theorem 2 proves that itemset similarity is antimonotonic.

Theorem 2 (Antimonotonicity of Itemset Similarity) *Given three FIs I, I', and I'' such that $I \sqsubseteq I' \sqsubseteq I''$, for all sets of transactions and all $\lambda \in [0..1]$ it is $sim(I,I') \geq sim(I,I'')$.*

Proof: Support-based similarity is antimonotonic along the containment relationship. This can be easily proved by considering that, in Equation (12.1), since $I \sqsubseteq I' \sqsubseteq I''$ it is $glb(I,I') = I'$ and $glb(I,I'') = I''$, and by recalling that function *sup* is antimonotonic. Similarly, feature-based similarity is antimonotonic along the containment relationship. This can be easily proved by considering that, in Equation (12.2), since $I \sqsubseteq I' \sqsubseteq I''$ it is $lub(I,I') = lub(I,I'') = I$, and by recalling that $rel(I') \leq rel(I'')$ by assumption. But then, *sim* is antimonotonic for any λ in that it is a linear combination of antimonotonic functions. □

The next property, formalized by Theorem 3, states that direct itemset containment (see Definition 35) always entails higher similarity, i.e., in case an itemset I' is directly contained in an itemset I''', the similarity between I' and I''' is greater than the similarity of I' to any other itemset I'' that does not directly contain I'. To prove this result we rely on the following lemma.

Lemma 1 *Given a POS of FIs $\mathscr{F}$ and three FIs $I',I'',I''' \in \mathscr{F}$ such that $I''' = glb(I',I'')$, $I' \not\sqsubseteq I''$, $I'' \not\sqsubseteq I'$, for all sets of transactions and all $\lambda \in [0..1]$ it is $sim(I',I''') \geq sim(I',I'')$ and $sim(I'',I''') \geq sim(I',I'')$.*

Proof: We prove the thesis by contradiction, assuming that $sim(I',I'') > sim(I',I''')$. We consider sim_{sup} first:

$$\frac{sup(glb(I',I''))}{sup(I')+sup(I'')-sup(glb(I',I''))} > \frac{sup(glb(I',I'''))}{sup(I')+sup(I''')-sup(glb(I',I'''))}$$

From here, recalling that $glb(I',I''') = I'''$, we get

$$\frac{sup(I''')}{sup(I')+sup(I'')-sup(I''')} > \frac{sup(I''')}{sup(I')}$$

Since $sup(I'') \geq sup(I''')$ because of Theorem 1, this inequality admits no solution. This also holds for the relevance component of the similarity function, sim_{fea}. Let $I = lub(I',I'')$:

$$\frac{rel(lub(I',I''))}{rel(glb(I',I''))} > \frac{rel(lub(I',I'''))}{rel(glb(I',I'''))} \Leftrightarrow \frac{rel(I)}{rel(I''')} > \frac{rel(I')}{rel(I''')}$$

But since $I \sqsubseteq I'$, for Definition 37 we have $rel(I') \geq rel(I)$, then the inequality admits no solution and we have a contradiction. From the two contradictions it follows that $sim(I',I''') \geq sim(I',I'')$ and $sim(I'',I''') \geq sim(I',I'')$. □

Theorem 3 *Given a POS of FIs $\mathscr{F}$ and any two FIs $I',I'' \in \mathscr{F}$, for each $I''' \in \mathscr{F}$ such that $I' \dot{\sqsubseteq}_{\mathscr{F}} I'''$, it is $sim(I',I'') \leq sim(I',I''')$.*

Proof: The proof comes by applying Theorem 2 to the result of Lemma 1. □

This theorem allows to sensibly reduce the summarization space in Section 12.5, narrowing the number of cluster comparisons down to the pairs of clusters whose representatives are directly contained into one another (i.e., it prevents an all-against-all comparison).

12.5 Summarizing frequent itemsets

Since SUSHI supports interactive exploration and navigation, it gives summaries a hierarchical organization to let them be analyzed at different levels of detail. To create summaries we adopt an agglomerative hierarchical clustering technique, which progressively merges couples of clusters starting from singletons until one single cluster is obtained. The result

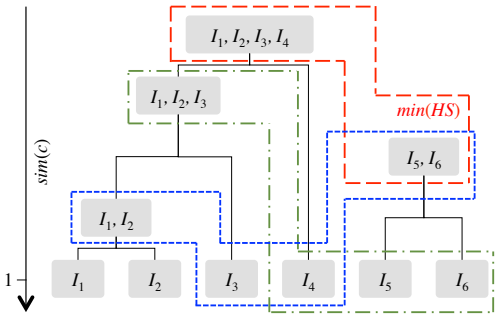


Figure 12.6: A simple h-summary with three possible summaries, including the minimum one

of hierarchical clustering is commonly represented using a dendrogram, i.e., a binary tree structure containing a k-block set partition for each $1 \leq k \leq n$, where n is the number of objects to be clustered [62]. In our context, due to the constraints we pose on mergeability, in some cases it is impossible to merge all the FIs into one single cluster, so the dendrogram will actually not be a tree but a forest of trees.

12.5.1 Summaries and representatives

Definition 39 (H-Summary and Summary) *Given a POS of FIs $\mathscr{F}$, a* hierarchical summary *(briefly,* h-summary*) of $\mathscr{F}$ is a (directed) forest HS where each node is a cluster $c \subseteq \mathscr{F}$ and each leaf is a singleton cluster. Each node c is labeled with the similarity $sim(c)$ of the representatives of the two clusters that were merged to obtain c (conventionally, $sim(c) = 1$ for singleton clusters). A* summary *S of HS is any set of clusters in HS that define a (complete and disjoint) partition of $\mathscr{F}$.*

Among all possible summaries for HS, the one including all and only the roots of HS is denoted as $min(HS)$ and called *minimum summary* (see Figure 12.6).

Note that, while traditional dendrograms are typically cut at a given similarity level to produce a clustering, in our definition a summary can include clusters at different similarity levels. This is to allow analysts to interactively choose which cluster they want to analyze in more detail during visualization (see Section 12.6).

In a summary, each cluster is represented by a single FI. To define the representative $rep(c)$ of cluster c, we provide three different strategies:

- *Top*: the representative is the most general FI in c, $rep(c) \sqsubseteq I, \forall I \in c$
- *Bottom*: the representative is the most specific FI in c, $I \sqsubseteq rep(c), \forall I \in c$
- *Medoid*: the representative is the cluster "center", $rep(c) = med(c)$

The Top strategy is the one most commonly used in the literature [40], while Bottom is related to the notion of *maximal FI* (i.e., a FI for which none of its immediate supersets is frequent [122]). As confirmed by both our tests and assessment with real users (see Section 12.7), Top often lacks in properly characterizing the clusters. The reason for this is that, as the cluster cohesion decreases, one or more features appearing in some of the cluster FIs may disappear from the representative, which leads to very low relevance. Conversely, when the Bottom representative is adopted, all the features appearing in at least one FI of the cluster are included. Obviously, both Top and Bottom are *extreme* representatives of clusters, thus they often do not show high similarity with the other FIs in the the cluster. To overcome this problem, we alternatively propose the Medoid strategy; to the best of our knowledge, no previous approach relies on medoid representatives for FIs. A medoid is defined as follows:

Definition 40 (Medoid) *Given a summary HS and a cluster $c \in HS$, a FI $\bar{I} \in c$ is a* candidate medoid *of c if, for each other FI $I^* \in c$, it is $sim(\bar{I},I^*) > 0$ and $\sum_{I \in c} sim(\bar{I},I) \geq \sum_{I \in c} sim(I^*,I)$. If c has at least one candidate medoid, then the* medoid *of c, denoted $med(c)$, is the one with the highest relevance and, at the same relevance, with the highest support.*

Given two clusters c', c'', computing the medoid of $c' \cup c''$ requires all the pairwise similarities between the FIs to be estimated. This entails the computation of a quadratic number of similarities in the cardinality of $c' \cup c''$ (each FI against all the others), which can be unfeasible for large datasets:

$$med(c) = argmax_{\bar{I} \in c' \cup c''} \sum_{I \in c' \cup c''} sim(\bar{I},I)$$

To cut this complexity down to linear, while merging two clusters we can approximate the optimal medoid based on the cluster representatives:

$$\begin{aligned} med^*(c',c'') &= argmax_{\bar{I} \in c' \cup c''} sim(\bar{I},rep(c')) \cdot |c'| + sim(\bar{I},rep(c'')) \cdot |c''| \\ &\simeq argmax_{\bar{I} \in c' \cup c''} \sum_{I \in c'} sim(\bar{I},I) + \sum_{I \in c''} sim(\bar{I},I) = med(c' \cup c'') \end{aligned}$$

Thanks to the last step, for each candidate medoid $\bar{I}$ we can replace the sum of the similarities between $\bar{I}$ and all other FIs with the similarity between $\bar{I}$ and the two representative medoids.

Depending on the strategy adopted for picking cluster representatives, different mergeability constraints arise. Indeed, two clusters can actually be merged only if the resulting cluster is well-formed, i.e., if it has a representative itself. Note that, for a singleton cluster $c = \{I\}$, it is $rep(c) = I$ for all strategies.

Definition 41 (Mergeability) *Two clusters c and c' are mergeable (denoted $c \leftrightarrow c'$) if: $rep(c) \sqsubseteq rep(c')$ or $rep(c') \sqsubseteq rep(c)$, in case of Top or Bottom strategies; the union of c and c' has a medoid, in case of Medoid strategy.*

Example 39 *With reference to the POS of FIs $\mathscr{F}$ in Figure 12.5, let the following clusters be given: $c_1 = \{\{\mathsf{College}\}, \{\mathsf{College}, \mathsf{Store}\}\}$, $c_2 = \{\{\mathsf{Amenity}\}\}$, $c_3 = \{\{\mathsf{Low}\}\}$. According to the Top strategy, it is $rep(c_1) = \{\mathsf{College}\}$; c_1 can be merged with c_2 but not with c_3 since $\{\mathsf{College}\} \not\sqsubseteq \{\mathsf{Low}\}$ and $\{\mathsf{Low}\} \not\sqsubseteq \{\mathsf{College}\}$. According to the Bottom strategy, it is $rep(c_1) = \{\mathsf{College}, \mathsf{Store}\}$; c_1 can be merged with c_2 but not with c_3 since $\{\mathsf{College}, \mathsf{Store}\} \not\sqsubseteq \{\mathsf{Low}\}$ and $\{\mathsf{Low}\} \not\sqsubseteq \{\mathsf{College}, \mathsf{Store}\}$. According to the Medoid strategy, it is $rep(c_1) = \{\mathsf{College}, \mathsf{Store}\}$; c_1 can be merged with c_2 but not with c_3 since $glb(\{\mathsf{College}, \mathsf{Store}\}, \{\mathsf{Low}\}) \notin \mathscr{F}$, hence, $sim(\{\mathsf{College}, \mathsf{Store}\}, \{\mathsf{Low}\}) = 0$ and $c_1 \cup c_3$ has no candidate medoids.*

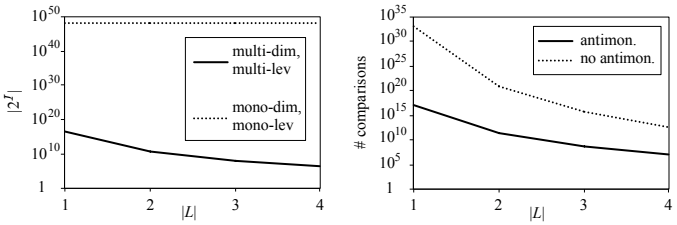
12.5.2 Search space for multi-dimensional and multi-level FIs

Hierarchical clustering on a set $\mathscr{F}$ of FIs entails $|\mathscr{F}|(|\mathscr{F}|-1)/2$ comparisons between FIs; so, reasoning about the cardinality of $\mathscr{F}$ is crucial. The actual cardinality of $\mathscr{F}$ clearly depends on the number of transactions in the dataset and on the frequency threshold. To discuss the impact of the multi-dimensional and multi-level setting on $|\mathscr{F}|$ in general terms, we consider the worst possible case, in which $\mathscr{F} = 2^{\mathscr{I}}$ because there is a transaction for each possible itemset and the threshold is 1 (all itemsets are frequent).

When working with mono-dimensional and mono-level items, the total number of possible (non-empty) itemsets is $|2^{\mathscr{I}}| = 2^n - 1$, where $n = |\mathscr{I}|$ is the total number of items. In the multi-dimensional case these n items are not homogeneous, i.e., they are picked from the domains of multiple dimensions. If v is the cardinality of these domains, which we assume for simplicity to be constant, the number of possible itemsets cuts down to

$$|2^{\mathscr{I}}| = \sum_{k=1}^{n/v} \binom{n/v}{k} \cdot v^k = (v+1)^{n/v} - 1 \tag{12.3}$$

where n/v is the number of dimensions (assuming for simplicity that all features are disjunctive). Finally, in the multi-level case, some meta-knowledge of part-of relationships between items is available and expressed in the form of hierarchies as in Definition 33, so some more combinations of items become unfeasible: indeed, we recall from Definition 34 that there cannot be any part-of relationship between the values of the items included in an itemset (e.g., $\{(\mathsf{worksIn}, \mathrm{Harlem}), (\mathsf{worksIn}, \mathrm{Manhattan})\}$ is not an itemset). In this case, the number



(a) (b)

Figure 12.7: (a) Search space measured as the number of itemsets in function of the number of levels; (b) number of comparisons at the first iteration in function of the number of levels

of possible itemsets is further reduced to

$$|2^{\mathscr{I}}| = \sum_{k=1}^{n/(\nu|L|)} \binom{n/(\nu|L|)}{k} \cdot |L|^k \cdot \nu^k = (\nu|L|+1)^{n/(\nu|L|)} - 1 \tag{12.4}$$

where $|L|$ is the number of levels in each hierarchy (assumed for simplicity to be the same for all hierarchies) and $n/(\nu|L|) = |H|$ is the number of hierarchies. Clearly, if $|L| = 1$ (flat hierarchies), Equation (12.4) boils down to Equation (12.3).

Figure 12.7(a) shows how the number of itemsets computed with Equation (12.4) changes when the number of levels $|L|$ changes from 1 (mono-level items) to 4 (multi-level items), with $n = 160$ and $\nu = 10$. As a reference, also the number of itemsets in the case of mono-dimensional and mono-level items is plotted. Clearly, when dimensions are structured in hierarchies the number of FIs is significantly smaller than in the "flat" case.

Even if $|\mathscr{F}|$ is significantly smaller for multi-level items than for mono-level ones, executing all $|\mathscr{F}|(|\mathscr{F}|-1)/2$ comparison would be expensive. Fortunately, this is not the case in our approach due to Theorem 3 that allows to compare only the couples of clusters whose representatives are directly contained into one another with reference to the set of all clusters representatives. The most expensive iteration for agglomerative hierarchical clustering algorithm is the first one, when all clusters are singleton so there are $|\mathscr{F}|$ clusters. Assuming for simplicity that all hierarchies are linear (i.e., no diamond-like hierarchies are present), an itemset including k items has exactly k fathers in the containment lattice. The total number of comparisons required at this stage is:

$$\sharp \text{ comparisons} = \sum_{k=1}^{n/(\nu|L|)} \binom{n/(\nu|L|)}{k} \cdot |L|^k \cdot \nu^k \cdot k = n(\nu|L|+1)^{n/(\nu|L|)-1}$$

Figure 12.7(b) plots this formula for increasing numbers of hierarchy levels, distinguishing whether the antimonotonicity of the similarity function is considered or not. Besides the

reduction of the number of comparisons due to antimonotonicity exploitation, the plot, as in Figure 12.7(a), also emphasizes how multi-level containment further reduces the number of comparisons. In the subsequent iterations of the clustering algorithm, the number of comparisons varies depending on the actual shape of the hierarchies and on the strategy chosen for cluster representatives, but it is always smaller than the first one since (i) the number of clusters decreases at each iteration, and (ii) not all pairs of clusters can actually be merged due to the mergeability constraints.

Example 40 *Given $n = 120$, $l = 3$, $v = 10$ and a null support threshold (i.e., all itemsets are frequent) it is $|2^{\mathscr{I}}| = |\mathscr{F}| = 923520$. The number of comparisons is $36 \cdot 10^5$, that is less than the number of required comparisons required by mono-dimensional and mono-level itemset clustering, $|\mathscr{F}|(|\mathscr{F}|-1)/2 = 43 \cdot 10^{10}$.* □

12.5.3 Building h-summaries

In this section we explain how h-summaries can be efficiently computed by agglomerative clustering. Considering the drastic reduction in the number of useful comparisons enabled by the antimonotonicity of the similarity function and by mergeability constraints, the efficiency of the agglomerative clustering algorithm can be greatly improved by storing the set of *useful comparisons* at each iteration within a graph $UC = (C, M)$, where C is the current set of clusters and M stores an arc for each couple of clusters whose comparison is actually useful[1]. Keeping the arcs in M sorted by decreasing similarity values ensures that the most similar couple of clusters can be found in constant time.

The pseudocode for summarizing FIs is shown in Algorithm 9. HS stores the h-summary being built by agglomerative hierarchical clustering. Graph $UC = (C, M)$ keeps track of cluster mergeability: at each iteration C is the set of clusters in the minimum summary $min(HS)$ (clearly, $min(HS)$ changes at each iteration as clusters are progressively merged and HS is expanded), while M stores an arc for each couple of candidate mergeable clusters, i.e., clusters with direct containment (Line 2). We recall that safely restricting to consider direct containment is possible thanks to Theorem 3; indeed, at each step, Algorithm 9 merges the two most similar clusters, which are necessarily two clusters related by direct containment. We use the term *candidate* since, while with the Top and Bottom strategies mergeability is ensured by the building rules of UC, the Medoid strategy requires a further check since the changes occurred in the cluster composition may affect mergeability.

[1] Since the notions of multidimensional cube and measure are not formally referred here, in order to keep a user-friendly notation, we reassigned letters C, c and M respectively a set of clusters, a cluster, and a set of mergeability arcs.

Algorithm 9 Create H-Summary

INPUT $\mathscr{F}$: POS of FIs, k: number of clusters, *strat* representative strategy
OUTPUT HS: h-summary

1: $C = \{\{I\}$ s.t. $I \in \mathscr{F}\}$ ▷ Set of singleton clusters, one for each FI
2: $M \leftarrow \{(c',c'')$ s.t. $c' \in C \wedge c'' \in C \wedge rep(c') \dot{\sqsubseteq}_{\mathscr{F}} rep(c'')\}$ ▷ Couples of clusters with direct containment
3: $HS \leftarrow C$ ▷ HS is initialized
4: $UC \leftarrow (C,M)$ ▷ UC is initialized
5: **while** $(M \neq \varnothing) \wedge (|min(HS)| > k)$ **do** ▷ While the h-summary is expandable...
6: $(\overline{c}',\overline{c}'') \leftarrow argmax_{\{(c',c'')\in M\}} sim(rep(c'),rep(c''))$ ▷ ...find the pair of candidate mergeable clusters with most similar representatives;
7: **if** $\overline{c}' \leftrightarrow \overline{c}''$ **then** ▷ if they are actually mergeable according to *strat*...
8: $\overline{c} \leftarrow \overline{c}' \cup \overline{c}''$ ▷ ...merge them,
9: $AddFather(HS,\overline{c},\overline{c}',\overline{c}'')$ ▷ update HS,
10: $Update(UC,\overline{c},\overline{c}',\overline{c}'',strat)$ ▷ and update UC
11: **return** HS

The algorithm starts by initializing HS with singleton clusters, one for each FI (Line 3); this can be seen as a degenerate h-summary since no merge took place so far. At Line 4 we initialize $UC = (C,M)$: C is the set of singleton clusters, while M stores the couples of singletons whose FIs are directly contained into one another with reference to $\mathscr{F}$.

Then, while the cardinality of $min(HS)$ is above the target cardinality k and some pairs of mergeable clusters exist (Line 5), we pick the pair of mergeable clusters $\overline{c}'$ and $\overline{c}''$ whose representatives have maximum similarity (Line 6); since the arcs in M are sorted by descending similarity, this simply means getting the first arc. These two clusters are then merged into cluster $\overline{c}$ (Line 8) and the h-summary HS is updated by adding the $\overline{c}$ as the father of $\overline{c}'$ and $\overline{c}''$ (Line 9). Finally, at Line 10, UC is updated as described by Algorithm 10.

Algorithm 10 updates UC according to the representative strategy adopted. Figure 12.8 shows which arcs are kept and which ones are dropped when $\overline{c}'$ and $\overline{c}''$ are merged. Note that arc $(\hat{c},\overline{c}')$ cannot belong to M since otherwise it would be $rep(\hat{c}) \sqsubseteq rep(\overline{c}') \sqsubseteq rep(\overline{c}'')$, but then $(\hat{c},\overline{c}'')$ would not be a direct containment (with reference to the set of all cluster representatives), which is impossible by construction (see Line 12). Symmetrically, the same holds for $(\check{c},\overline{c}'')$. Arc $(\overline{c}',\overline{c}'')$ is dropped in all the strategies; moreover: in Top, arc $(\hat{c},\overline{c}'')$ is dropped since with this strategy the representative of $\overline{c}$ is the representative of $\overline{c}'$ (i.e., the most general one), thus $\hat{c}$ and $\overline{c}$ are not mergeable since $rep(\overline{c}) = rep(\overline{c}') \not\sqsubseteq rep(\hat{c})$; Bottom is the dual case of *Top*; in Medoid, containment of representatives is not necessary, so all clusters remain potentially mergeable and no more arcs are removed.

Example 41 *Here we show some steps of Algorithm 9 using the Bottom strategy, with reference to the POS of FIs shown in Figure 12.5. The h-summary HS is initialized with 9 singleton clusters (gray boxes in Figure 12.9(a)). Among the 11 pairs of mergeable clusters whose FIs are directly contained one into the other with reference to $\mathscr{F}$ (the pairs*

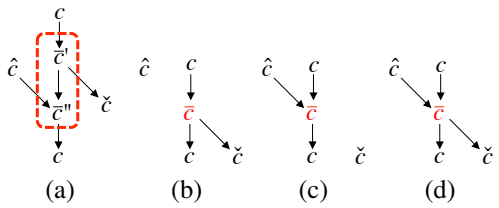


Figure 12.8: Changes in UC when clusters $\overline{c}'$ and $\overline{c}''$ are merged: before merging (a), after Top merging (b), after Bottom merging (c), and after Medoid merging (d)

Algorithm 10 Update

INPUT $UC = (C, M)$: graph of useful comparisons, $\overline{c}', \overline{c}''$: clusters to be merged (with $(\overline{c}', \overline{c}'') \in M$), $\overline{c}$: cluster resulting from the merge, *strat*: representative strategy
OUTPUT UC: updated graph of useful comparisons

1: $C \leftarrow C \cup \overline{c}$
2: **switch** *strat* **do**
3: **case** *Top*
4: $M \leftarrow M \cup \{(\overline{c}, c) \text{ s.t. } (\overline{c}'', c) \in M \vee (\overline{c}', c) \in M\}$ ▷ $\overline{c}$ inherits outgoing arcs of $\overline{c}''$ and $\overline{c}'$
5: $M \leftarrow M \cup \{(c, \overline{c}) \text{ s.t. } (c, \overline{c}') \in M\}$ ▷ $\overline{c}$ inherits the incoming arcs of $\overline{c}'$
6: **case** *Bottom*
7: $M \leftarrow M \cup \{(c, \overline{c}) \text{ s.t. } (c, \overline{c}') \in M \vee (c, \overline{c}'') \in M\}$ ▷ $\overline{c}$ inherits incoming arcs of $\overline{c}'$ and $\overline{c}''$
8: $M \leftarrow M \cup \{(\overline{c}, c) \text{ s.t. } (\overline{c}'', c) \in M\}$ ▷ $\overline{c}$ inherits the outgoing arcs of $\overline{c}''$
9: **case** *Medoid*
10: $M \leftarrow M \cup \{(c, \overline{c}) \text{ s.t. } (c, \overline{c}') \in M \vee (c, \overline{c}'') \in M\}$ ▷ $\overline{c}$ inherits incoming arcs of $\overline{c}'$ and $\overline{c}''$
11: $M \leftarrow M \cup \{(\overline{c}, c) \text{ s.t. } (\overline{c}', c) \in M \vee (\overline{c}'', c) \in M\}$ ▷ $\overline{c}$ inherits outgoing arcs of $\overline{c}'$ and $\overline{c}''$
12: $Clear(UC, \overline{c}', \overline{c}'')$ ▷ Remove $\overline{c}'$ and $\overline{c}''$ from C and all arcs in M that (i) involve $\overline{c}'$ or $\overline{c}''$, (ii) do not represent direct containment
13: **return** UC

connected by arrows in Figure 12.9(a)), the pair $(\{\mathsf{Store}\}, \{\mathsf{Macy's}\})$ *is the one with highest similarity (0.96). So these two clusters are merged; the representative of the new cluster is FI* $\{\mathsf{Macy's}\}$ *as shown in Figure 12.9(b). At the second iteration, among the 8 pairs of mergeable clusters, the one with highest similarity is* $(\{\mathsf{Amenity}, \mathsf{Low}\}, \{\mathsf{Store}, \mathsf{Low}\})$ *(Figure 12.9(b)); the representative of the merged cluster is* $\{\mathsf{Store}, \mathsf{Low}\}$*. After 7 iterations, two clusters are obtained (shown in Figure 12.10(b)); since these two clusters are not mergeable, the algorithm stops and returns an h-summary including these two roots. Figure 12.10 also shows the minimum summaries obtained using the Top and Medoid strategies, emphasizing the differences between the composition and the representative of the resulting clusters.*

12.5.4 Complexity

The computational complexity of hierarchical clustering has a two-faceted nature: initialization and iteration. While the initialization cost (Algorithm 1, Line 2) has been discussed in Section 12.5.2, we now consider the iteration cost. Note that the complexity of our algorithms does not depend on the number of transactions in the dataset, but it depends on the number

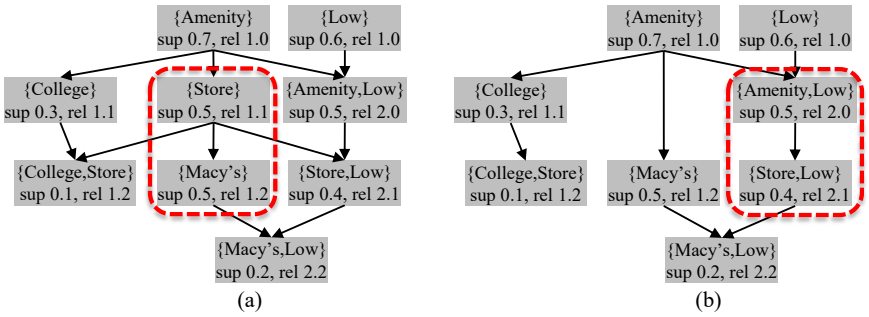


Figure 12.9: Clustering steps shown on graph UC using the bottom strategy; first (a) and second (b) iterations (each cluster is shown by its representative, arrows represent cluster mergeability, dashed boxes show the next couple of clusters to be merged)

of FIs (i.e., on $|\mathscr{F}|$). For instance, in the case of colossal trajectory mining [273, 302], even only a few dozens of transactions can produce a huge number of FIs.

By relying on a priority queue (Algorithm 9, Line 6), the worst-case complexity of hierarchical clustering is $O(|\mathscr{F}|^2 log|\mathscr{F}|)$ [63]: clustering requires $|\mathscr{F}|-1$ merging steps, each of which requires $|\mathscr{F}|$ similarity computations to be stored in the priority queue. Editing a priority queue has a $log|\mathscr{F}|$ complexity. However, merging two clusters entails different complexities according to the adopted strategy. On the one hand, Bottom and Top require no validation for merged cluster, so the actual complexity of clustering is $O(|\mathscr{F}|^2 log|\mathscr{F}|)$. On the other hand, the computation of a feasible medoid for two clusters c' and c'' has quadratic complexity, resulting in a complexity $O(|\mathscr{F}|^3))$. Although medoid approximations exist (e.g., [211, 20]), they require statistical assumptions that are not know while merging the clusters[2]. Remarkably, the medoid approximation med^* proposed in Section 12.5.1 has linear complexity, resulting in an overall complexity $O(|\mathscr{F}|^2 log|\mathscr{F}|)$.

Note that in our setting the worst-case scenario is actually quite unlikely, since it requires that: (i) the goal is to aggregate all FIs into a single cluster; (ii) only in the last iteration of the algorithm (when there is a single cluster) the optimal and approximated medoid complexities are $O(|\mathscr{F}|^3)$ and $O(|\mathscr{F}|^2 log|\mathscr{F}|)$; and (iii) no direct containment relationships between FIs are present, so that the the antimonoticity of the similarity function cannot be used to prune the search space.

As to space complexity, hierarchical clustering requires to store both the graph of useful comparisons (UC in Algorithm 9) and the priority queue that sorts the mergeability arcs. Storing UC requires to store both its set of nodes, C, and its set of arcs, M. As to C, all the FIs in $\mathscr{F}$ (i.e., the initial singleton clusters) are stored in memory; if f is the average space

[2]The estimation of statistical parameters proved to produce "bad" medoids when the cluster population was not sufficient to provide a precise estimation or when global parameters where extended even to small clusters.

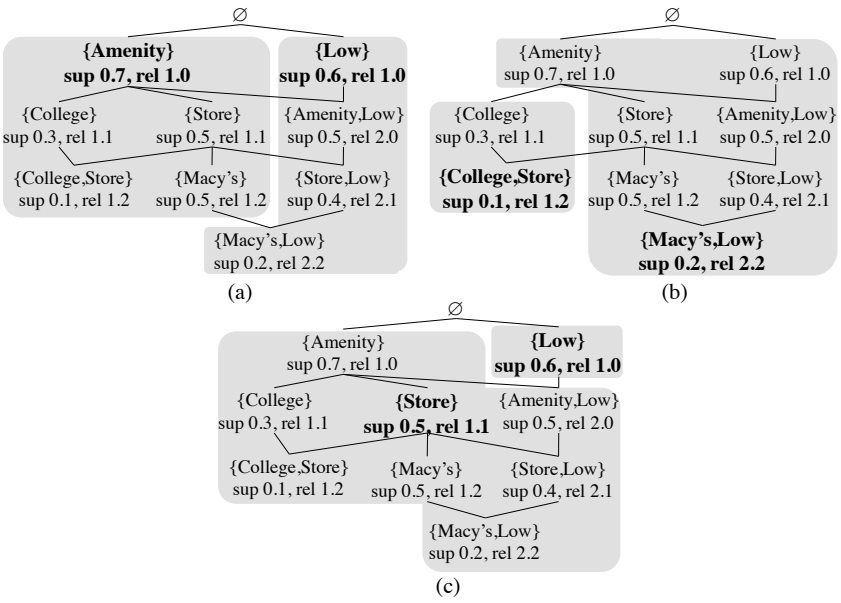


Figure 12.10: Minimum summaries obtained using the Top (a), Bottom (b), and Medoid (c) strategies shown on $\mathscr{F}$ (gray areas represent clusters, their representatives are in bold)

taken by an FI (f depends on the size of each item and on the number of items per FI), storing C requires $|\mathscr{F}| \cdot f$ bytes. As to M, each mergeability arc includes a reference to the two clusters (8 bytes in total) and the similarity between their representatives (4 bytes), yielding $|M| \cdot 12$ bytes overall. Finally, the priority queue contains the references to all mergeability arcs, overall $|M| \cdot 4$ bytes. Thus, the total space taken in memory is $|\mathscr{F}| \cdot f + |M| \cdot 16$ bytes. Differently from the computational time, this space does not depend on the summarization strategy as the clustering process does not require to copy new objects in memory, but only to partition graph UC (see Section 12.7).

12.6 Visualizing and exploring summaries

Though several summarization approaches have been devised as discussed in Section 12.2, most of them do not provide user-friendly ways to visually explore the summaries obtained. To make data analysis more effective, SUSHI builds on an interactive visual interface (available at http://semantic.csr.unibo.it/sushi) that enables analysts to unveil information hidden in summaries.

Initially, when visualizing an h-summary HS, an overview of the minimum summary $min(HS)$ is provided by showing the top k clusters sorted by cardinality. If $|min(HS)| > k$, to avoid scaling issues the remaining clusters are grouped into a fictitious "other" cluster.

SUSHI then provides two different visual approaches that fulfill orthogonal requirements to navigate HS:

- The *graph-based* approach overviews the *entire* h-summary by always showing a complete summary, as in the classical approaches to FI visualization discussed in Section 12.2, and enables analysts to expand/collapse clusters. This approach highlights inter-cluster relationships (e.g., in our profiling case study, how common behaviors relate to each other).

- In the *tree-based* approach, the context of visualization is a single cluster, shown with its children and grandchildren within the h-summary; thus, the relationships between its representative and the other FIs in the cluster are emphasized (e.g., in our profiling case study, how common a behavior is). Analysts can zoom in and out one cluster.

These two approaches are pursued using well-known visualization layouts, respectively, directed acyclic graphs (DAGs) [102] and treemaps [24] (Figure 12.11). In both approaches, colors code both the feature with the highest relevance of the cluster representative (hue) and its support (saturation).

Both visualization approaches share two interaction possibilities (see Section 12.6):

- *options*: analysts are allowed to select the data source (e.g., a file containing the FIs), the representative strategy (Bottom, Top, or Medoid), the desired number of clusters k, and the similarity coefficient λ;

- *filter*: the data source can be filtered by applying a support threshold and by specifying the specific items the analyst is interested in (e.g., visualizing only the FIs that contain items (worksIn,Harlem) or (frequents,Museum) with a support between 0.25 and 0.5).

However, as discussed in the following subsections, they differ in the primitives enabling analysts to navigate h-summaries.

12.6.1 Graph-based visualization

DAGs gracefully represent inter-cluster relationships within a summary in terms of direct containment between cluster representatives. At each iteration, analysts can expand a thick-bordered cluster or collapse a cluster. Thin-bordered clusters represents singleton clusters that cannot be further expanded. At the maximum level of detail, i.e., when all clusters have been completely expanded, the summary including all singleton clusters is displayed. More precisely, given the summary S currently displayed through the DAG and a cluster $c \in S$: (i) the expansion of c replaces c with its k most similar descendants in HS which completely

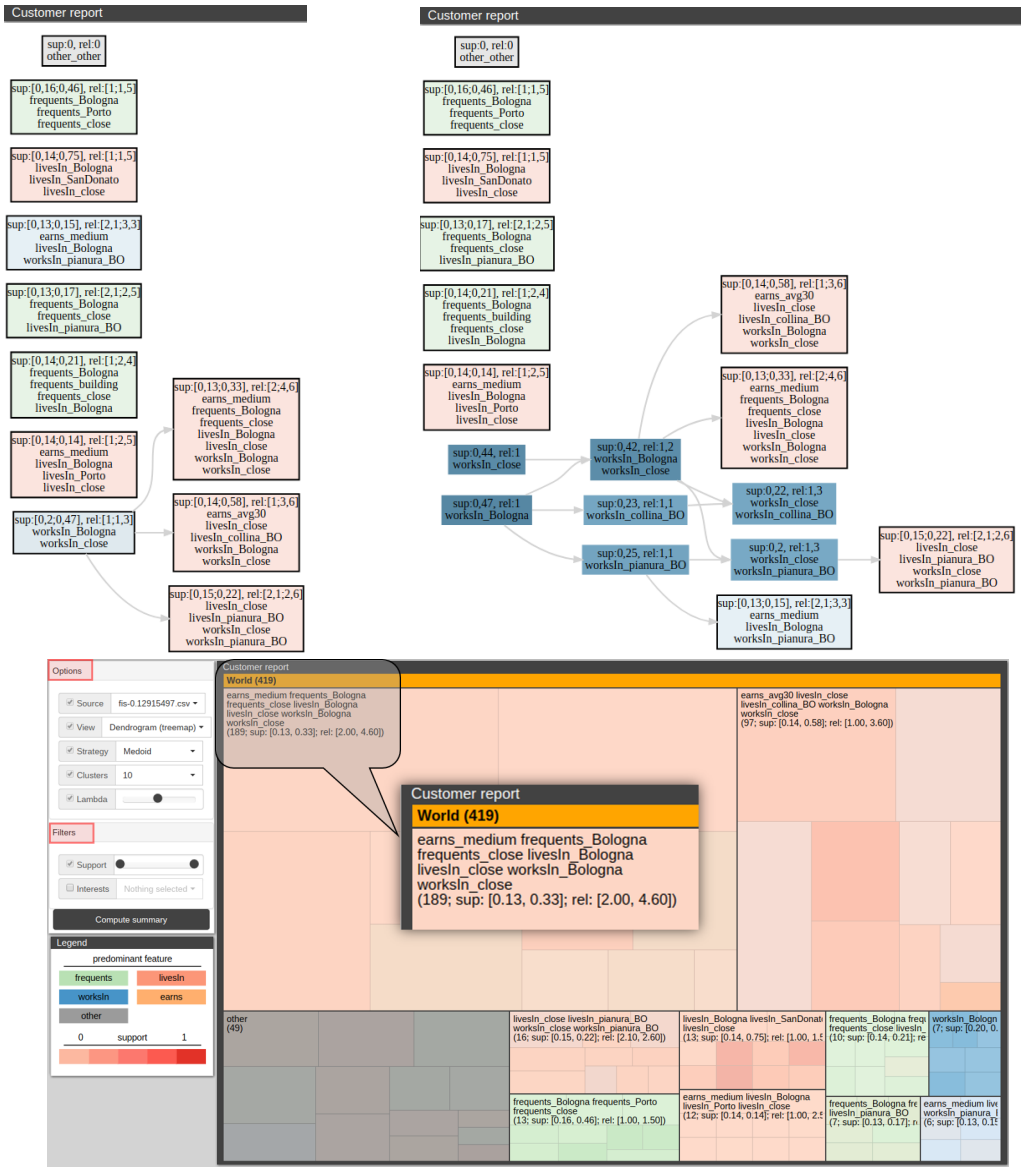


Figure 12.11: Graph-based visualization, before (a) and after (b) the expansion of cluster $\{(\mathsf{worksIn}, \mathsf{Bologna}), (\mathsf{worksIn}, \mathsf{close})\}$; tree-based visualization (c)

and disjointly cover the FIs in c, so as to obtain a new summary S'; (ii) the collapse of c undoes its expansion (similarly to the zoom-in, extend, and zoom-out primitives in [252]).

Section 12.6 shows an example of the DAG representing $min(HS)$ ($k = 10$), while Section 12.6 shows how the graph changes when a cluster is expanded. DAG visualization is based on the Graphviz open source library [79]. The DAG should be examined from left to right; leftmost clusters have representatives with low item cardinality (high support, low relevance), while rightmost clusters have representatives with high item cardinality (low support, high relevance). This helps analysts in finding containment paths between cluster

representatives, and is the reason for keeping FIs sorted in space due to the antimonotonic and monotonic properties of $sup()$ and $rel()$, respectively.

12.6.2 Tree-based visualization

Ordered treemaps are popular visualization tools for large hierarchical datasets, and we use them to map h-summaries into 2D areas [251]. Section 12.6 shows a treemap (drawn using the D3 library) in which the visualization area is partitioned into $k = 10$ (plus one "others") rectangles, each corresponding to a cluster in $min(HS)$. The area of rectangles is proportional to the cluster cardinality, hence, this visualization emphasizes the volume of FIs summarized by each representative. Each rectangle is divided into sub-rectangles corresponding to its k descendants with highest similarity $sim(c)$ in HS. By hovering over these sub-rectangles, analysts can get further details. Physical nesting is used instead of arcs to represent inter-cluster relationships, so the visualization is clearer and easier to interpret, and scalability is improved.

At each iteration, analysts can zoom-in or zoom-out a cluster c. More precisely, (i) a zoom-in of c displays its k most similar descendants in HS which completely and disjointly cover FIs in c; (ii) a zoom-out of c undoes the last zoom-in (similarly to the zoom-in, filter, and zoom-out primitives in [252]).

12.7 Experimental tests

Summarization can be evaluated in terms of (i) effectiveness (summary compaction, cohesion, information loss, and interestingness), (ii) efficiency (computational performance), and (iii) understandability of the summary. We evaluate SUSHI using two datasets: a real one, called ProfilingDS, related to the Profiling domain schema, and a synthetic one, called SyntheticDS.

- ProfilingDS describes the behavior of 20000 mall customers in the city of Bologna (Italy). The profile of each customer is obtained from her daily GPS trajectories, and modeled through a transaction set $\mathscr{T}$ using multiple features (where she lives and works, the places she frequents, and how much she earns). Transactions are then enriched with multi-level and multi-dimensional knowledge from external open data sources, namely, Open Street Map (https://www.openstreetmap.org/) and the Italian Statistic Institute ISTAT (https://www.istat.it/); the classical Apriori algorithm [9] is applied to extract the set of FIs $\mathscr{F}$ out of $\mathscr{T}$.

- SyntheticDS includes multiple POSs of FIs. While all these POSs share the same number of hierarchies ($|H| = 4$) and values per level ($\nu = 20$), they have different

Table 12.2: Notation summary

Notation	Meaning
$\|H\| \in [3,6]$	Number of hierarchies
$\|L\| \in [2,5]$	Number of levels per hierarchy
$v = 20$	Number of values per level
$\lambda \in [0,0.6]$	Similarity constant
$k = 10$	Number of desired clusters
$\mathscr{F}$	Set of FIs
HS	H-summary
$min(HS)$	Minimum summary
$\|M\|$	Number of mergeability comparisons

cardinalities $|\mathscr{F}|$ and different hierarchy depths $|L|$. Additional details will be given in Section 12.7.1.

Finally, for both datasets, $\mathscr{F}$ is summarized into an h-summary HS. The notation we adopt is summarized in Table 12.2.

12.7.1 Effectiveness of the summarization strategies

Effectiveness is evaluated from different perspectives: compaction gain, information loss, interestingness, and cluster cohesion. The first two have been defined in [40] to evaluate a summary S of the set of FIs $\mathscr{F}$, the third one in [137][3]:

- *Compaction gain*: the reduction with respect to $\mathscr{F}$,

$$Gain(S) = |\mathscr{F}|/|S|$$

- *Information loss*: the total amount of information missing from S,

$$Loss(S) = \sum_{c \in S} \sum_{I \in c} \sum_{f \in Feat(I)} loss_f(I,c)$$

where $loss_f(I,c)$ counts the number of values for feature f that are present in I but not in $rep(c)$.

[3] In [137], the authors also provide a metric for intelligibility which is not applicable to our approach since it works on itemsets with a fixed schema, while we use schemaless itemsets.

- *Interestingness*: how different the summary elements are (intuitively, the more unbalanced the summary, the higher its interestingness):

$$Int(S) = \sum_{c \in S} \frac{|c| \cdot (|c| - 1)}{|\mathscr{F}| \cdot (|\mathscr{F}| - 1)}$$

Compaction gain, information loss, and interestingness do not measure how similar the elements of each cluster are to the cluster representative; for this reason we complement them with the *cohesion* of the clusters in S, defined as:

$$Coh(S) = \frac{1}{|S|} \sum_{c \in S} \sum_{I \in c, I \neq rep(c)} \frac{sim(rep(c), I)}{|c| - 1}$$

Figure 12.12 compares —in terms of cohesion, compaction gain, information loss, and interestingness — the minimum summaries $min(HS)$ produced by the Bottom, Medoid, and Top strategies of SUSHI for ProfilingDS. We comment below the main outcomes:

- Bottom and Medoid outperform Top in terms of cohesion and information loss as a direct consequence of how they are conceived. As to interestingness, Medoid outperforms both other strategies, while Bottom achieves a lower interestingness than Top as it produces more fragmented clusters.

- Higher cohesion values could be obtained by increasing the number of clusters in the h-summary, $|min(HS)|$, which however might make the summary very complex. Figure 12.13 compares the three strategies for increasing numbers of clusters, showing that Medoid outperforms Top since it produces higher cohesion and lower loss of information; additionally, Medoid can be applied even for more compact summaries (i.e., for low values of $|min(HS)|$). Conversely, Bottom shows its limitations: since the number of maximal FIs is a lower bound to $|min(HS)|$, it always produces a larger number of clusters.

- SUSHI allows users to tune the relative weight λ of support-based and feature-based similarity when computing itemset relevance. Indeed, as claimed in Section 12.4, clusters with similar features but different support might underlie different correlations (in ProfilingDS, different customer behaviors). The impact of λ is more apparent for Medoid, while Top and Bottom are less sensitive to it.

- As to information loss, for a fixed summary cardinality, Top and Bottom are at the highest and lowest ends of the range. This is not surprising since, intuitively, Top and Bottom tend to generalize and to preserve the summarized information, respectively.

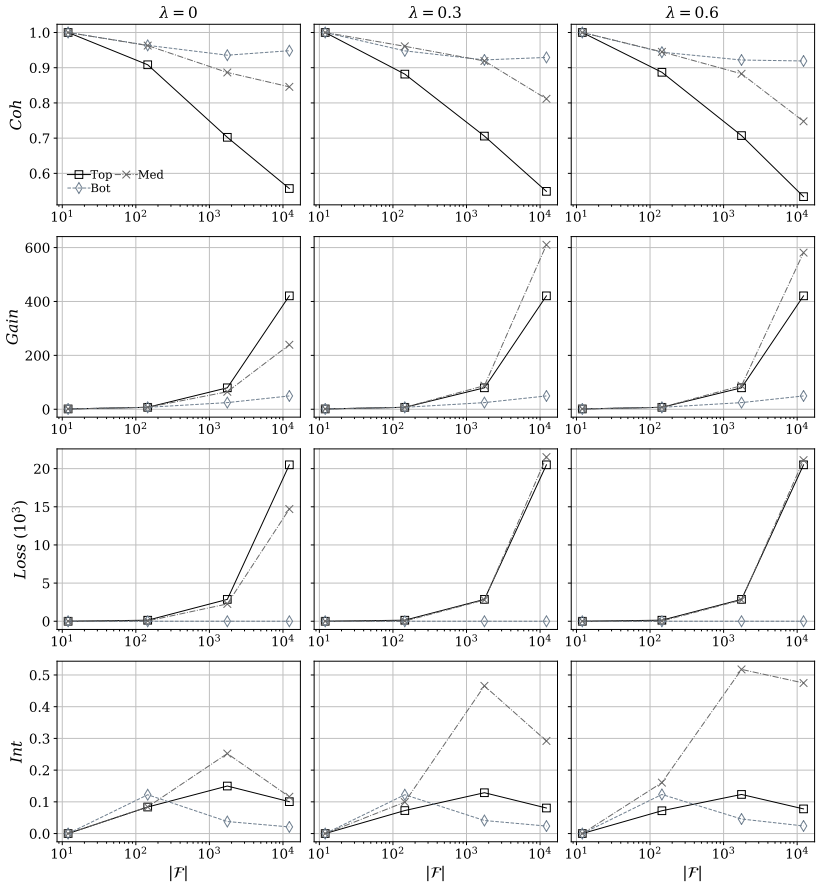


Figure 12.12: Comparison of the Bottom, Medoid, and Top strategies in terms of effectiveness (with $k = 20$) for ProfilingDS

SUSHI explicitly keeps hierarchy into account, thus it is interesting to analyze how the hierarchy structure impacts on effectiveness. To this end we rely on our synthetic dataset, SyntheticDS. First of all, in Figure 12.14 we compare the three strategies on four POSs including 5000 FIs when the depth $|L|$ of the four hierarchies changes from 2 to 5. Medoid outperforms Bottom and Top in terms of compaction gain and interestingness. For shallow hierarchies, Top and Medoid provide the most compact summaries but, as expected, Medoid produces more cohesive summaries. Conversely, Bottom produces clusters with high cohesion and no information loss (by definition) at the cost of a lower compaction gain (i.e., several small clusters). Noticeably, Medoid —which is always better than Top in terms of cohesion and information loss— also overcomes Top in terms of gain for deep hierarchies since, in this case, each medoid is similar to several FIs (see Definition 38).

The h-summaries returned by SUSHI are inherently hierarchical. While the tests described above only analyze the properties of the minimum summary, it is also interesting to analyze

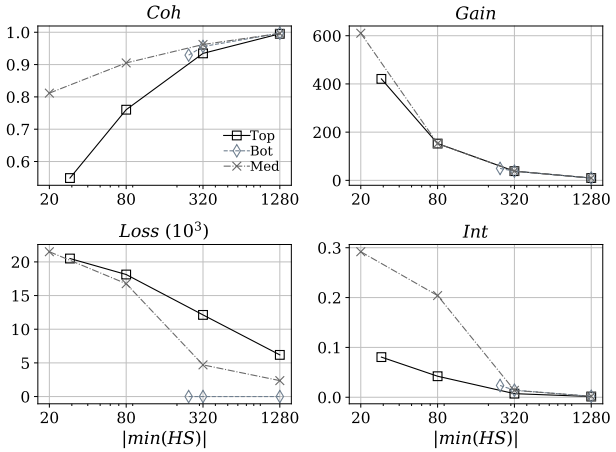


Figure 12.13: Comparison of the Bottom, Medoid, and Top strategies in terms of effectiveness (with $|\mathscr{F}| = 12000$, $\lambda = 0.3$) for ProfilingDS

how the properties of summaries change when moving from large and very detailed ones to compact and less informative ones. To do this, we analyze how the properties of the minimum summary change as Algorithm 9 reduces its cardinality. In particular, Figure 12.15 shows how cohesion, number $|M|$ of mergeability comparisons, compaction gain, information loss, and interestingness evolve for increasing iteration steps. Not surprisingly, Bottom and Top prune more mergeability arcs than Medoid, with Bottom being the strategy that requires less steps to conclude the algorithm. The three strategies merge progressively less cohesive clusters as iterations proceed. While at early iterations the three strategies merge small clusters, towards the end (approximately after the first 5000 iterations) the pruning of mergeability arcs favors the creations of larger clusters.

Overall, we can conclude that the Top strategy produces compact but not cohesive summaries (because it tends to generalize), the Bottom strategy produces cohesive but not compact summaries (because it preserves details), while the Medoid strategy achieves the best balance between compactness and cohesion. In the process of FI exploration, the three strategies provide different insights: Top enables users to uncover general behaviors supported by a large FI population; Bottom uncovers idiosyncratic behaviors supported by a population of cohesive FIs; Medoid mitigates these effects, uncovering behaviors supported by the most cohesive FIs.

Optimal vs. approximated medoid

As described in Section 12.5.1, for the sake of scalability, we also implemented an approximated version of the optimal medoid. At every merging step, the percentage variation

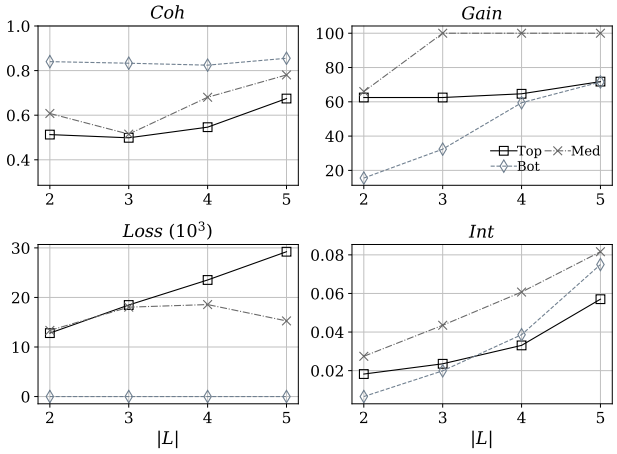


Figure 12.14: Comparison of the Bottom, Medoid, and Top strategies in terms of effectiveness (with $k = 20$, $\lambda = 0.3$) for SynteticDS

in cluster cohesion is less than 1% (i.e., even when the approximated medoid is different from the optimal one, they are still highly similar). However, following an iterative process, changing the representative of even a few clusters might produce different summary results. Figure 12.16 shows how the summaries produced by the approximated and optimal medoids affect the Medoid strategy for increasing summary cardinalities. Noticeably, the provided summaries have the same compaction gain. By obtaining summaries with the same number of clusters and similar cluster medoids, the variation in information loss is also minimal. The summary produced by the approximated strategy tends to be slightly more unbalanced as the medoid is not always centered in the cluster, resulting in higher interestingness (due to the quadratic effect of cluster cardinality, small cardinality variations are amplified in the interestingness metric). Due to unbalancing, the approximated medoid produces a larger number of smaller clusters than the optimal medoid, resulting in a slightly higher average cohesion (cohesion only differs by 0.05 in the worst case). We can conclude that the approximated medoid does represent a valid alternative to the optimal one.

12.7.2 Efficiency of the summarization strategies

We ran the tests on a machine equipped with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz CPU and 4GB RAM; all measures are in seconds. We emphasize that we implemented SUSHI in a centralized and sequential architecture; an implementation in a big data distributed solution is out of the chapter scope.

As depicted in Figure 12.17-left and Figure 12.18-left, SUSHI runs in near-real time even when thousands of FIs are considered. It is apparent that Medoid is computationally

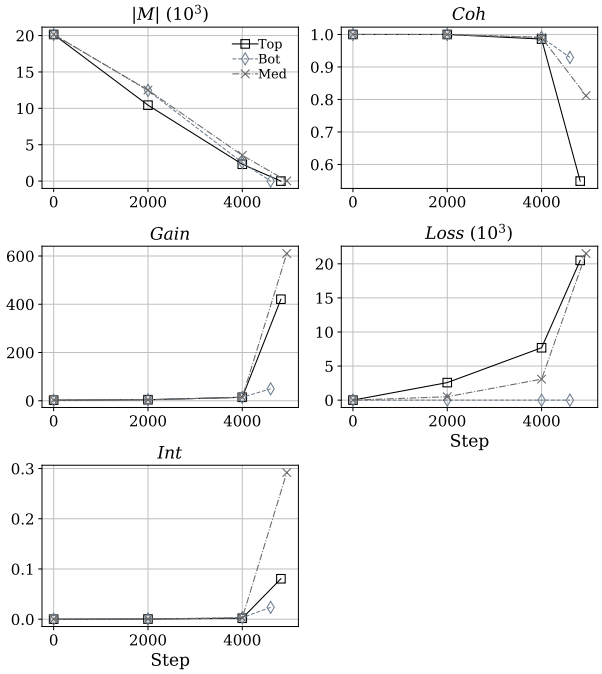


Figure 12.15: Comparison of the Bottom, Top, and Medoid strategies while building the h-summary ($k = 20$, $|\mathscr{F}| = 12000$, $\lambda = 0.3$) for ProfilingDS

heavier than Bottom and Top, due to the quadratic complexity required to compute the medoids. Remarkably, when the approximated medoid is used, the time for summarization is comparable to that of Top and Bottom. The computational gap becomes larger as the average cluster size grows at the different steps of Algorithm 9 (see Figure 12.17-right). Overall, the performances of the summarization strategies are not sensible to the hierarchy structure (Figure 12.18-right).

As shown in Figure 12.19-left, the memory usage clearly increases with the number of FIs, $|\mathscr{F}|$. However, summarizing 10^5 FIs only requires 400MB. This happens also in Figure 12.19-right by increasing the number of hierarchy levels $|L|$ and by keeping a fixed amount of itemsets, $|\mathscr{F}| = 5000$: in fact, deeper hierarchies produce more mergeability arcs in Algorithm 9. As anticipated in Section 12.5.4, the memory usage does not depend on the summarization strategies as all the produced summaries are hierarchical partitions of the existing $\mathscr{F}$.

12.7.3 Comparison against BUS and MBUS

To the best of our knowledge, no previous approaches address the summarization of multi-level and multi-dimensional FIs. The closest contribution to SUSHI are [40] and its extension

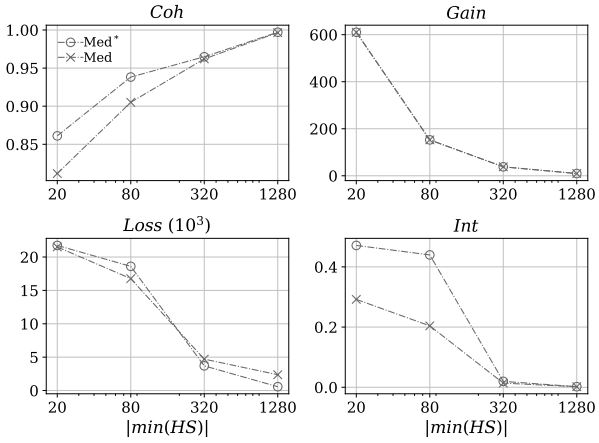


Figure 12.16: Comparison of the optimal (Med) and approximated (Med*) Medoid strategies in terms of effectiveness ($|\mathscr{F}| = 12000$, $\lambda = 0.3$) for ProfilingDS

[137], in which the authors introduce the BUS and MBUS algorithms to summarize transactions with a fixed schema. Before performing a quantitative comparison, we report the key differences between SUSHI and (M)BUS:

1. SUSHI relies on a multi-dimensional and multi-level similarity (Definition 38), while in (M)BUS similarity is expressed in terms of set containment (i.e., an FI I summarizes I' if I is a subset of I').

2. SUSHI relies on hierarchical summaries expressed as dendrograms, which natively code containment and similarity relationships between clusters; this allows to interactively navigate h-summaries (i.e., expansion/collapse in Section 12.6.1 and zoom-in/zoom-out in Section 12.6.2) by following containment paths. Conversely, (M)BUS generates "flat" summaries (i.e., plain sets of FIs); in (M)BUS, summary navigation requires multiple runs of the algorithm with different values of $|min(HS)|$, which does not preserve the relationships among clusters.

3. SUSHI implements three summarization strategies, while (M)BUS implements only the Top strategy.

Due to these differences, BUS and MBUS cannot be directly compared to SUSHI. Thus, to compare the three approaches in terms of effectiveness we limit SUSHI to the Top strategy and only consider the minimum summaries it produces. Figure 12.20 compares —in terms of cohesion, compaction gain, information loss, and interestingness— the minimum summaries $min(HS)$ produced by the Top strategy of SUSHI and the summaries produced by BUS and MBUS. Clearly, while the three approaches are comparable as to compaction gain,

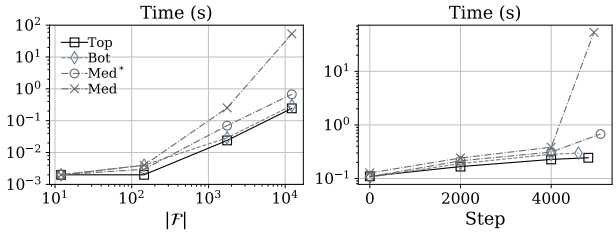


Figure 12.17: Efficiency in function of (left) the number of FIs $|\mathscr{F}|$ ($k = 20$, $\lambda = 0.3$) and (right) the number of algorithm steps ($k = 20$, $\lambda = 0.3$, $|\mathscr{F}| = 12000$) for ProfilingDS

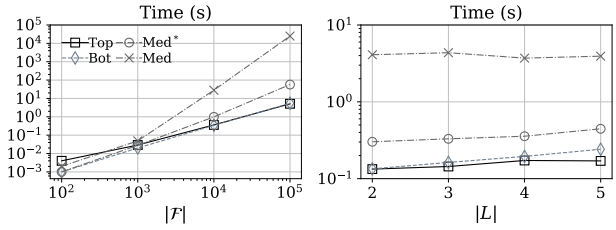


Figure 12.18: Efficiency in function of (left) the number of FIs $|\mathscr{F}|$ and (right) the hierarchy depth $|L|$ ($k = 20$, $\lambda = 0.3$) for SyntheticDS

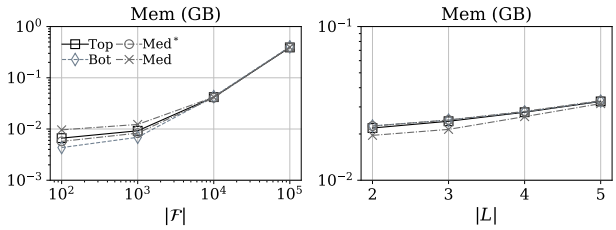


Figure 12.19: Memory usage in function of (left) the number of FIs $|\mathscr{F}|$ and (right) the hierarchy depth $|L|$ ($k = 20$, $\lambda = 0.3$) for SyntheticDS

information loss, and interestingness, SUSHI significantly outperforms the others in terms of cohesion. This happens because the SUSHI similarity function captures hierarchical similarity between FIs, while in (M)BUS similarity is limited to set containment. So, for instance, in (M)BUS the two FIs $I = \{(\mathsf{frequents}, \mathrm{Store})\}$ and $I' = \{(\mathsf{frequents}, \mathrm{Macy's})\}$ have null similarity, hence, they cannot be clustered together.

As to efficiency, Figure 12.21 shows that all three SUSHI strategies outperform BUS and MBUS by orders of magnitude. The execution of the algorithms was stopped when $|\mathscr{F}| = 1100$, since obtaining larger numbers of FIs would require hours. The dramatic improvement of SUSHI is strictly related to its capability of reducing the number of comparisons between FIs by exploiting hierarchies, as formally described in Section 12.5.2.

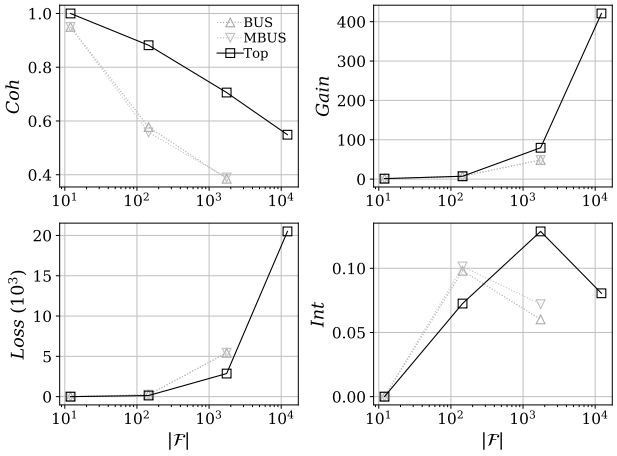


Figure 12.20: Comparison of the Top strategy, BUS ,and MBUS in terms of effectiveness (with $k = 20$, $\lambda = 0.3$) for ProfilingDS

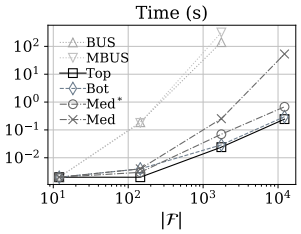


Figure 12.21: Comparison of SUSHI, BUS, and MBUS in terms of efficiency in function of the number of FIs ($k = 20$, $\lambda = 0.3$) for ProfilingDS

To sum up, from the discussion above it clearly emerges that without a dedicated approach, such as SUSHI, the summarization of multi-level and multi-dimensional FIs yields poor performances from both points of view of effectiveness and efficiency.

12.7.4 Summary understandability

To assess the quality of the summary and the visual experience with SUSHI, we conducted a set of tests with 15 users, mainly master students in data science with basic or advanced knowledge of FI mining. After a 15-minutes introduction to SUSHI and to our profiling case study, the users were asked to answer 6 analytical questions in 6 minutes each, and finally to fill out a qualitative questionnaire. Of the 6 questions, two were answered using a spreadsheet software with a plain CSV file, two using graph-based visualization, and two using tree-based visualization; the dataset included 419 FIs (analyzing a larger dataset in a plain file would have been too complex). Here is an example of question:

Which common behavior do the mall customers show among the following ones?

Table 12.3: Outcome of user evaluation

	Spreadsheet	**Graph-based**	**Tree-based**
Preferences	2	3	10
Complexity	7	6.7	5.8
Time (s)	262	308	231
Score	0.64	0.63	0.7

1. $\{(\mathsf{livesIn}, \mathrm{Bologna}), (\mathsf{livesIn}, \mathrm{close}), (\mathsf{earns}, \mathrm{Medium})\}$
2. $\{(\mathsf{livesIn}, \mathrm{Bologna}), (\mathsf{worksIn}, \mathrm{far}), (\mathsf{earns}, \mathrm{High})\}$
3. $\{(\mathsf{livesIn}, \mathrm{Bologna}), (\mathsf{earns}, \mathrm{10to35}), (\mathsf{frequents}, \mathrm{HillsInBologna})\}$

Table 12.3 summarizes the quantitative results. From the comments made in the questionnaire, it emerges that the spreadsheet is too dispersive and hardly manageable for large datasets; graph-based visualization provides an intuitive overview of the summary and well highlights FI relationships, though navigation is difficult when several clusters are displayed; tree-based visualization gives an intuitive overview of the summary, a focused navigation, and good readability.

Overall, users mostly appreciated tree-based visualization, which yields the lowest complexity in understanding summaries, friendly in-depth navigation, and the lowest time per answer. Both SUSHI visualizations provide intuitive overviews of the summary, with tree-based visualization allowing the most focused navigation. The representativeness of the visualized FI with respect to the FIs within the same group is also well perceived. Though the overall scores are comparable, spreadsheet analysis is deemed to be too dispersive and not effective for FI summarization. The longer answering time with graph-based visualization with respect to spreadsheet is explained by considering that the brief training made for SUSHI could not balance the previous experience of users with spreadsheets.

12.8 Conclusion

The new applications emerging in the era of analytics and big data ask for the study of new machine learning techniques as well as for revamping established ones. In particular, our work has been inspired by a real profiling study based on the tracking of GPS positions of people. Although multi-level and multi-dimensional FIs are a perfect way to represent the behavior of clusters of customers, the huge number of FIs mined hinders their effective analysis. For this reason we proposed SUSHI, an original approach to FI summarization and visualization based on an innovative similarity function. SUSHI turned out to outperform

previous approaches both from the efficiency and effectiveness points of view. Moreover, the proposed similarity and visualization techniques were successful in the tests with real users, proving to be a valuable tool to expedite the analysis of FIs.

Chapter 13

Conclusion and Research Directions

In the second part of the thesis, we investigated how advanced analytics can ease climbing the knowledge pyramid, helping data scientists with no vertical knowledge of computer science and data engineering to access data. In particular, through augmented, conversational, and intentional OLAP, we provided advanced analytic abstractions at a higher level than formal queries and programming languages; the three contributions answer to three different needs: (i) having frameworks that proactively recommend analytics queries based on the task at hand, (ii) providing a user-friendly interface to formal queries through natural language, and (iii) providing high-level analytics intentions. Finally, with multidimensional summarization, we addressed the compression of OLAM results (i.e., with frequent itemsets as a case study) by exploiting multidimensional similarity. In this direction, the research challenges are plentiful.

Sensors produce valuable contextual data that, in Augmented OLAP, have been exploited to recommend interesting analytical queries. From a broader perspective, it is interesting to correlate context-awareness to data quality issues. It has been recognized that contextual assessments can be as important as objective quality indicators because they affect which information gets used for decision making tasks [280]. Specifically, the data quality dimensions impacted by contextual data are relevancy, value-added, timeliness, completeness, and volume [258].

Conversational and Intentional OLAP synergically work on providing higher-level interfaces and analytic abstractions. Also, they overcome the idea of querying a cube to get a plain result (e.g., a pivot table). On the one hand, Conversational OLAP allows data scientists to access data through natural language, overcoming the need of formal languages. In this context, our goal is to extract a meaningful representation that, in turn, allows a succinct vocalization of the results. On the other hand, Intentional OLAP introduces high-level user-friendly abstractions (e.g., describe, assess, and explain) to perform analytic tasks

mixing formal queries, data mining, and machine learning tasks. In this context, while we designed and implemented the describe intention, our goal is to extend the approach to the other intention operators to produce enhanced cubes (i.e., cubes augmented with interesting highlights coming from mining and machine learning models).

Finally, the summarization of multidimensional results plays an important role in all the explored directions, since they benefit from succinct visualization metaphor (Augmented OLAP), vocalization (Conversational OLAP), and insight extraction (Intentional OLAP). In this context, while we focused on the intrinsic multidimensional nature of frequent itemsets, our goal is to extend our approach to multidimensional cubes, allowing the summarization of results based both on pattern similarity and novel interestingness measures.

Bibliography

[1] (2018). Extracting business value from the 4 v's of big data. https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data. [Online; accessed 21-June-2019].

[2] (Accessed 09/12/2019). Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng.

[3] Abela, A. (2008). *Advanced presentations by design*. Pfeiffer.

[4] Abul, O., Bonchi, F., and Nanni, M. (2008). Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proc. ICDE*, pages 376–385. IEEE Computer Society.

[5] Abul, O., Bonchi, F., and Nanni, M. (2010). Anonymization of moving objects databases by clustering and perturbation. *Inf. Syst.*, 35(8):884–910.

[6] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145.

[7] Affolter, K., Stockinger, K., and Bernstein, A. (2019). A comparative survey of recent natural language interfaces for databases. *VLDB J.*, 28(5):793–819.

[8] Afrati, F. N., Gionis, A., and Mannila, H. (2004). Approximating a collection of frequent sets. In *Proc. KDD*, pages 12–19.

[9] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proc. VLDB*, pages 487–499.

[10] Ahmed, M. (2019). Data summarization: a survey. *Knowl. Inf. Syst.*, 58(2):249–273.

[11] Ajanki, A., Billinghurst, M., Jrvenpaa, T., Kandemir, M., Kaski, S., Koskela, M., Kurimo, M., Laaksonen, J., Puolamaki, K., Ruokolainen, T., and Tossavainen, T. (2010). Contextual information access with augmented reality. In *Proc. Int. Work. on Machine Learning for Signal Processing*, pages 95–100.

[12] Al-Hussaeni, K., Fung, B. C. M., Iqbal, F., Dagher, G. G., and Park, E. G. (2018). Safepath: Differentially-private publishing of passenger trajectories in transportation systems. *Comput. Networks*, 143:126–139.

[13] Aligon, J., Gallinucci, E., Golfarelli, M., Marcel, P., and Rizzi, S. (2015). A collaborative filtering approach for recommending OLAP sessions. *Decis. Support Syst.*, 69:20–30.

[14] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E. (2014). Similarity measures for OLAP sessions. *Knowl. Inf. Syst.*, 39(2):463–489.

[15] Aligon, J., Marcel, P., and Negre, E. (2011). Summarizing and querying logs of OLAP queries. In Guillet, F., Pinaud, B., Venturini, G., and Zighed, D. A., editors, *Advances in Knowledge Discovery and Management*, volume 3, pages 99–124. Springer.

[16] Almeida, A. M. R., Lima, M. I. V., de Macêdo, J. A. F., and Machado, J. C. (2016). DMM: A distributed map-matching algorithm using the mapreduce paradigm. In *Proc. ITSC*, pages 1706–1711, Rio de Janeiro.

[17] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proc. AFIPS*, pages 483–485, Atlantic City, New Jersey, USA.

[18] Ashbrook, D. and Starner, T. (2002). Learning significant locations and predicting user movement with GPS. In *Proc. ISWC*, pages 101–108, Seattle, WA, USA.

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

[20] Bagaria, V. K., Kamath, G. M., Ntranos, V., Zhang, M. J., and Tse, D. (2018). Medoids in almost-linear time via multi-armed bandits. In *Proc. AISTATS*, volume 84, pages 500–509. PMLR.

[21] Bahrainian, S. A. and Crestani, F. (2017). Towards the next generation of personal assistants: Systems that know when you forget. In *Proc. ICTIR*, pages 169–176, Amsterdam, The Netherlands.

[22] Baralis, E., Cagliero, L., Cerquitelli, T., D'Elia, V., and Garza, P. (2010). Support driven opportunistic aggregation for generalized itemset extraction. In *Proc. IS*, pages 102–107, London, UK.

[23] Beatty, J. C. (1982). On the relationship between LL(1) and LR(1) grammars. *J. ACM*, 29(4):1007–1022.

[24] Bederson, B. B., Shneiderman, B., and Wattenberg, M. (2002). Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854.

[25] Bentayeb, F. and Favre, C. (2009). RoK: Roll-up with the k-means clustering method for recommending OLAP queries. In *Proc. DEXA*, pages 501–515.

[26] Beresford, A. R. and Stajano, F. (2004). Mix zones: User privacy in location-aware services. In *Proc. PerCom Workshops*, pages 127–131. IEEE Computer Society.

[27] Bie, T. D. (2013). Subjective interestingness in exploratory data mining. In *Proc. IDA*, pages 19–31.

[28] Bilogrevic, I., Huguenin, K., Jadliwala, M., Lopez, F., Hubaux, J., Ginzboorg, P., and Niemi, V. (2013). Inferring social ties in academic networks using short-range wireless communications. In *Proc. WPES*, pages 179–188. ACM.

[29] Bindschaedler, V. and Shokri, R. (2016). Synthesizing plausible privacy-preserving location traces. In *Proc. IEEE Symposium on Security and Privacy*, pages 546–563. IEEE Computer Society.

[30] Bindschaedler, V., Shokri, R., and Gunter, C. A. (2017). Plausible deniability for privacy-preserving data synthesis. *Proc. VLDB*, 10(5):481–492.

[31] Blunschi, L., Jossen, C., Kossmann, D., Mori, M., and Stockinger, K. (2012). SODA: generating SQL for business users. *Proc. VLDB*, 5(10):932–943.

[32] Börner, K. (2015). *Atlas of knowledge: anyone can map*. MIT Press.

[33] Bothorel, G., Serrurier, M., and Hurter, C. (2013). Visualization of frequent itemsets with nested circular layout and bundling algorithm. In *Proc. ISVC*, pages 396–405.

[34] Brant, K. and Sicular, S. (2018). Hype cycle for artificial intelligence, 2018. http://www.gartner.com/en/documents/3883863/hype-cycle-for-artificial-intelligence-2018. [Online; accessed 21-June-2019].

[35] Bulling, A., Cakmakci, O., Kunze, K., and Rehg, J. M. (2016). Eyewear computing - augmenting the human with head-mounted wearable assistants. *Dagstuhl Reports*, 6(1):160–206.

[36] Büschel, W., Mitschick, A., and Dachselt, R. (2018). Here and now: Reality-based information retrieval: Perspective paper. In *Proc. CHIIR*, pages 171–180, New Brunswick, USA.

[37] Carbonell, J. G. and Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. SIGIR*, pages 335–336.

[38] Cecaj, A., Mamei, M., and Zambonelli, F. (2016). Re-identification and information fusion between anonymized CDR and social network data. *J. Ambient Intell. Humaniz. Comput.*, 7(1):83–96.

[39] Chandler, T., Morgan, T., and Kuhlen, T. W. (2018). Exploring immersive analytics for built environments. In Marriott, K. et al., editors, *Immersive Analytics*, volume 11190 of *LNCS*, pages 331–357. Springer.

[40] Chandola, V. and Kumar, V. (2007). Summarization — compressing data into an informative representation. *Knowl. Inf. Syst.*, 12(3):355–378.

[41] Chang, K., Wei, L., Yeh, M., and Peng, W. (2011a). Discovering personalized routes from trajectories. In Jensen, C. S., Lee, W., Zheng, Y., and Mokbel, M. F., editors, *Proc. LBSN*, pages 33–40, Chicago, IL, USA. ACM.

[42] Chang, W., Wu, J., and Tan, C. C. (2011b). Friendship-based location privacy in mobile social networks. *Int. J. Secur. Networks*, 6(4):226–236.

[43] Chanson, A., Crulis, B., Drushku, K., Labroche, N., and Marcel, P. (2019). Profiling user belief in BI exploration for measuring subjective interestingness. In *Proc. DOLAP*.

[44] Chédin, A., Francia, M., Marcel, P., Peralta, V., and Rizzi, S. (2020). The tell-tale cube. In *Proc. ADBIS*, volume 12245 of *LNCS*, pages 204–218, Lyon, France. Springer.

[45] Chen, B., Chen, L., Lin, Y., and Ramakrishnan, R. (2005). Prediction cubes. In *Proc. VLDB*, pages 982–993.

[46] Chen, M., Sun, J., Ni, X., and Chen, Y. (2011). Improving context-aware query classification via adaptive self-training. In *Proc. CIKM*, pages 115–124, Glasgow, United Kingdom.

[47] Chen, R., Ács, G., and Castelluccia, C. (2012a). Differentially private sequential data publication via variable-length n-grams. In *Proc. CCS*, pages 638–649, Raleigh, NC, USA. ACM.

[48] Chen, R., Fung, B. C. M., Desai, B. C., and Sossou, N. M. (2012b). Differentially private transit data publication: a case study on the montreal transportation system. In *Proc. KDD*, pages 213–221. ACM.

[49] Chen, R., Fung, B. C. M., Mohammed, N., Desai, B. C., and Wang, K. (2013). Privacy-preserving trajectory data publishing by local suppression. *Inf. Sci.*, 231:83–97.

[50] Chiba, T., Sei, Y., Tahara, Y., and Ohsuga, A. (2019). Trajectory anonymization: Balancing usefulness about position information and timestamp. In *Proc. NTMS*, pages 1–6. IEEE.

[51] Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks. In *Proc. KDD*, pages 1082–1090. ACM.

[52] Chon, K., Hwang, S., and Kim, M. (2018). Gminer: A fast gpu-based frequent itemset mining method for large-scale data. *Inf. Sci.*, 439-440:19–38.

[53] Chu, C., Kim, S. K., Lin, Y., Yu, Y., Bradski, G. R., Ng, A. Y., and Olukotun, K. (2006). Map-reduce for machine learning on multicore. In *Proc. NIPS*, pages 281–288. MIT Press.

[54] Cicek, A. E., Nergiz, M. E., and Saygin, Y. (2014). Ensuring location diversity in privacy-preserving spatio-temporal data publishing. *VLDB J.*, 23(4):609–625.

[55] Colliat, G. (1996). Olap, relational, and multidimensional database systems. *SIGMOD Rec.*, 25(3):64–69.

[56] Croatti, A. and Ricci, A. (2017). Towards the web of augmented things. In *Proc. ICSA*, pages 80–87, Gothenburg, Sweden.

[57] Cumin, J., Petit, J., Scuturici, V., and Surdu, S. (2017). Data exploration with SQL using machine learning techniques. In *Proc. EDBT*, pages 96–107, Venice, Italy.

[58] Czauderna, T., Haga, J., Kim, J., Klapperstück, M., Klein, K., Kuhlen, T. W., Oeltze-Jafra, S., Sommer, B., and Schreiber, F. (2018). Immersive analytics applications in life and health sciences. In Marriott, K. et al., editors, *Immersive Analytics*, volume 11190 of *LNCS*, pages 289–330. Springer.

[59] Dai, J., Yang, B., Guo, C., and Ding, Z. (2015). Personalized route recommendation using big trajectory data. In Gehrke, J., Lehner, W., Shim, K., Cha, S. K., and Lohman, G. M., editors, *Proc. ICDE*, pages 543–554, Seoul, South Korea. IEEE Computer Society.

[60] Dai, Y., Shao, J., Wei, C., Zhang, D., and Shen, H. T. (2018). Personalized semantic trajectory privacy preservation through trajectory reconstruction. *World Wide Web*, 21(4):875–914.

[61] Dalenius, T. (1986). Finding a needle in a haystack or identifying anonymous census records. *J. of Official Statistics*, 2(3):329.

[62] Davidson, I. and Ravi, S. S. (2009). Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. *Data Min. Knowl. Discov.*, 18(2):257–282.

[63] Day, W. H. and Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

[64] De Montjoye, Y. A., Hidalgo, C. A., Verleysen, M., and Blondel, V. D. (2013). Unique in the Crowd: The privacy bounds of human mobility. *Scientific Reports*, 3.

[65] Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113.

[66] Deshpande, A. and Madden, S. (2006). MauveDB: supporting model-based user views in database systems. In *Proc. SIGMOD*, pages 73–84.

[67] Dhamdhere, K., McCurley, K. S., Nahmias, R., Sundararajan, M., and Yan, Q. (2017). Analyza: Exploring data with conversation. In *Proc. IUI*, pages 493–504, New York, NY, USA. ACM.

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

[69] Djenouri, Y., Drias, H., and Bendjoudi, A. (2014). Pruning irrelevant association rules using knowledge mining. *IJBIDM*, 9(2):112–144.

[70] Djenouri, Y., Lin, J. C., Nørvåg, K., and Ramampiaro, H. (2019). Highly efficient pattern mining based on transaction decomposition. In *Proc. ICDE*, pages 1646–1649. IEEE.

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

[72] Domingo-Ferrer, J. and Trujillo-Rasua, R. (2012). Microaggregation- and permutation-based anonymization of movement data. *Inf. Sci.*, 208:55–80.

[73] Dong, Y. and Pi, D. (2018). Novel privacy-preserving algorithm based on frequent path for trajectory data publishing. *Knowl.-Based Syst.*, 148:55–65.

[74] Douriez, M., Doraiswamy, H., Freire, J., and Silva, C. T. (2016). Anonymizing NYC taxi data: Does it matter? In *Proc. DSAA*, pages 140–148. IEEE.

[75] Drushku, K., Aligon, J., Labroche, N., Marcel, P., and Peralta, V. (2019). Interest-based recommendations for business intelligence users. *Inf. Syst.*, 86:79–93.

[76] Dwork, C. (2006). Differential privacy. In *Proc. ICALP*, volume 4052 of *LNCS*, pages 1–12. Springer.

[77] Dwyer, T., Riche, N. H., Klein, K., Stuerzlinger, W., and Thomas, B. H. (2016). Immersive analytics. *Dagstuhl Reports*, 6(6):1–9.

[78] Eddy, S. R. (1996). Hidden markov models. *Current opinion in structural biology*, 6(3):361–365.

[79] Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., and Woodhull, G. (2001). Graphviz - open source graph drawing tools. In *Graph Drawing*, pages 483–484.

[80] ElSayed, N. A. M., Thomas, B. H., Marriott, K., Piantadosi, J., and Smith, R. T. (2016). Situated analytics: Demonstrating immersive analytical tools with augmented reality. *J. Vis. Lang. Comput.*, 36:13–23.

[81] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, pages 226–231, Portland, Oregon, USA.

[82] Etcheverry, L. and Vaisman, A. A. (2012). QB4OLAP: A vocabulary for OLAP cubes on the semantic web. In *Proc. COLD*, volume 905 of *CEUR Workshop Proceedings*, Aachen, DEU. CEUR-WS.org.

[83] Fiore, M., Katsikouli, P., Zavou, E., Cunche, M., Fessant, F., Hello, D. L., Aivodji, U. M., Olivier, B., Quertier, T., and Stanica, R. (2019). Privacy of trajectory micro-data: a survey. *arXiv preprint arXiv:1903.12211*.

[84] Forney, G. D. (1973). The viterbi algorithm. *Proc. of the IEEE*, 61(3):268–278.

[85] Fortunato, S. (2009). Community detection in graphs. *CoRR*, abs/0906.0612.

[86] Franceschi-Bicchierai, L. (2015). Redditor cracks anonymous data trove to pinpoint muslim cab drivers. *Online at: http://mashable. com/2015/01/28/redditor-muslim-cab-drivers*.

[87] Francia, M., Gallinucci, E., and Golfarelli, M. (2019a). Social BI to understand the debate on vaccines on the web and social media: unraveling the anti-, free, and pro-vax communities in Italy. *Social Netw. Analys. Mining*, 9(1):46:1–46:16.

[88] Francia, M., Gallinucci, E., and Golfarelli, M. (2020a). Towards conversational OLAP. In *Proc. DOLAP@EDBT/ICDT*, volume 2572 of *CEUR Workshop Proceedings*, pages 6–15, Copenhagen, Denmark. CEUR-WS.org.

[89] Francia, M., Gallinucci, E., and Golfarelli, M. (2021a). Conversational OLAP in action. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 646–649.

[90] Francia, M., Gallinucci, E., and Golfarelli, M. (2021b). Cool: A framework for conversational OLAP. *Information Systems*, page 101752.

[91] Francia, M., Gallinucci, E., Golfarelli, M., and Santolini, N. (2020b). Dart: De-anonymization of personal gazetteers through social trajectories. *Journal of Information Security and Applications*, 55:102634.

[92] Francia, M., Gallinucci, E., and Vitali, F. (2019b). Map-matching on big data: a distributed and efficient algorithm with a hidden markov model. In *Proc. MIPRO*, pages 1238–1243, Opatija, Croatia. IEEE.

[93] Francia, M., Golfarelli, M., Marcel, P., Rizzi, S., and Vassiliadis, P. (2021c). Assess queries for interactive analysis of data cubes. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 121–132.

[94] Francia, M., Golfarelli, M., and Rizzi, S. (2018). A similarity function for multi-level and multi-dimensional itemsets. In *Proc. SEBD*, volume 2161 of *CEUR Workshop Proceedings*, pages 1–8, Castellaneta Marina, Taranto, Italy. CEUR-WS.org.

[95] Francia, M., Golfarelli, M., and Rizzi, S. (2019c). Augmented business intelligence. In *Proc. DOLAP@EDBT/ICDT*, volume 2324 of *CEUR Workshop Proceedings*, pages 1–10, Lisbon, Portugal. CEUR-WS.org.

[96] Francia, M., Golfarelli, M., and Rizzi, S. (2020c). A-BI$^+$: A framework for augmented business intelligence. *Inf. Syst.*, 92:101520.

[97] Francia, M., Golfarelli, M., and Rizzi, S. (2020d). Summarization and visualization of multi-level and multi-dimensional itemsets. *Inf. Sci.*, 520:63–85.

[98] Freudiger, J., Shokri, R., and Hubaux, J. (2011). Evaluating the privacy risk of location-based services. In *Proc. Financial Cryptography*, volume 7035 of *LNCS*, pages 31–46, Gros Islet, St. Lucia. Springer.

[99] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

[100] Fung, B. C. M., Wang, K., Chen, R., and Yu, P. S. (2010). Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53.

[101] Gambs, S., Killijian, M., and del Prado Cortez, M. N. (2011). Show me how you move and I will tell you who you are. *Trans. Data Privacy*, 4(2):103–126.

[102] Gansner, E. R., Koutsofios, E., North, S. C., and Vo, K. (1993). A technique for drawing directed graphs. *IEEE Trans. Software Eng.*, 19(3):214–230.

[103] Gedik, B. and Liu, L. (2005). Location privacy in mobile systems: A personalized anonymization model. In *ICDCS*, pages 620–629. IEEE Computer Society.

[104] Geetha, P., Naikodi, C., and Setty, S. L. N. (2020). Design of big data privacy framework—a balancing act. In Jain, V., Chaudhary, G., Taplamacioglu, M. C., and Agarwal, M. S., editors, *Advances in Data Sciences, Security and Applications*, pages 253–265, Singapore. Springer Singapore.

[105] Giacometti, A., Marcel, P., Negre, E., and Soulet, A. (2011). Query recommendations for OLAP discovery-driven analysis. *IJDWM*, 7(2):1–25.

[106] Gil, Y., Pierce, S. A., Babaie, H. A., Banerjee, A., Borne, K. D., Bust, G., Cheatham, M., Ebert-Uphoff, I., Gomes, C., Hill, M., Horel, J., Hsu, L., Kinter, J., Knoblock, C. A., Krum, D., Kumar, V., Lermusiaux, P., Liu, Y., North, C., Pankratius, V., Peters, S., Plale, B., Pope, A., Ravela, S., Restrepo, J., Ridley, A. J., Samet, H., and Shekhar, S. (2019). Intelligent systems for geosciences: an essential research agenda. *Commun. ACM*, 62(1):76–84.

[107] Gkesoulis, D. and Vassiliadis, P. (2013). CineCubes: cubes as movie stars with little effort. In *Proceedings of DOLAP*, pages 3–10, San Francisco, CA, USA.

[108] Gkesoulis, D., Vassiliadis, P., and Manousis, P. (2015). CineCubes: Aiding data workers gain insights from OLAP queries. *Inf. Syst.*, 53:60–86.

[109] Goga, O., Lei, H., Parthasarathi, S. H. K., Friedland, G., Sommer, R., and Teixeira, R. (2013). Exploiting innocuous activity for correlating users across sites. In *Proc. WWW*, pages 447–458, Rio de Janeiro, Brazil.

[110] Goh, C. Y., Dauwels, J., Mitrovic, N., Asif, M. T., Oran, A., and Jaillet, P. (2012). Online map-matching based on hidden markov model for real-time traffic sensing applications. In *Proc. ITSC*, pages 776–781, Anchorage, AK, USA.

[111] Golfarelli, M., Graziani, S., and Rizzi, S. (2014). Shrink: An OLAP operation for balancing precision and size of pivot tables. *Data Knowl. Eng.*, 93:19–41.

[112] 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.

[113] Golfarelli, M. and Rizzi, S. (2009). *Data warehouse design: Modern principles and methodologies*. McGraw-Hill, Inc.

[114] Golfarelli, M. and Rizzi, S. (2020). A model-driven approach to automate data visualization in big data analytics. *Information Visualization*, 19(1).

[115] Golfarelli, M. and Saltarelli, E. (2003). The workload you have, the workload you would like. In *Proc. DOLAP*, pages 79–85, New Orleans, USA.

[116] Golle, P. and Partridge, K. (2009). On the anonymity of home/work location pairs. In *Proc. Pervasive*, pages 390–397, Nara, Japan.

[117] Gouineau, F., Landry, T., and Triplet, T. (2016). Patchwork, a scalable density-grid clustering algorithm. In *Proc. ACM Symposium on Applied Computing*, pages 824–831, Pisa, Italy.

[118] Gramaglia, M. and Fiore, M. (2015). Hiding mobile traffic fingerprints with GLOVE. In *Proc. CoNEXT*, pages 26:1–26:13. ACM.

[119] Gramaglia, M., Fiore, M., Tarable, A., and Banchs, A. (2017). Preserving mobile subscriber privacy in open datasets of spatiotemporal trajectories. In *Proc. INFOCOM*, pages 1–9. IEEE.

[120] Guha, R. V., Gupta, V., Raghunathan, V., and Srikant, R. (2015). User modeling for a personal assistant. In *Proc. WSDM*, pages 275–284, New York, NY, USA. ACM.

[121] Guilly, M. L., Petit, J., and Scuturici, V. (2018). SQL query completion for data exploration. *CoRR*, abs/1802.02872.

[122] Gunopulos, D., Khardon, R., Mannila, H., and Toivonen, H. (1997). Data mining, hypergraph transversals, and machine learning. In *Proc. PODS*, pages 209–216.

[123] Gunopulos, D., Kollios, G., Tsotras, V. J., and Domeniconi, C. (2005). Selectivity estimators for multidimensional range queries over real attributes. *VLDB J.*, 14(2):137–154.

[124] Gupta, A., Harinarayan, V., and Quass, D. (1995). Aggregate-query processing in data warehousing environments. In *Proc. VLDB*, pages 358–369, San Francisco, CA, USA. Morgan Kaufmann.

[125] Gursoy, M. E., Liu, L., Truex, S., and Yu, L. (2019). Differentially private and utility preserving publication of trajectory data. *IEEE Trans. Mob. Comput.*, 18(10):2315–2329.

[126] Han, B., Liu, L., and Omiecinski, E. (2012). NEAT: road network aware trajectory clustering. In *Proc. ICDCS*, pages 142–151, Macau, China.

[127] Han, J. (1997). OLAP mining: Integration of OLAP with data mining. In *Proc. Working Conf. on Database Semantics*, pages 3–20.

[128] Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86.

[129] Han, J. and Fu, Y. (1999). Mining multiple-level association rules in large databases. *IEEE Trans. Knowl. Data Eng.*, 11(5):798–804.

[130] Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87.

[131] Han, Q., Lu, D., Zhang, K., Du, X., and Guizani, M. (2018). Lclean: A plausible approach to individual trajectory data sanitization. *IEEE Access*, 6:30110–30116.

[132] He, J., Chu, W. W., and Liu, Z. (2006). Inferring privacy information from social networks. In Mehrotra, S., Zeng, D. D., Chen, H., Thuraisingham, B. M., and Wang, F., editors, *Proc. ISI*, volume 3975 of *LNCS*, pages 154–165, San Diego, CA, USA. Springer.

[133] He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C. M., and Srivastava, D. (2015). DPT: differentially private trajectory synthesis using hierarchical reference systems. *Proc. VLDB*, 8(11):1154–1165.

[134] He, X., Zhao, K., and Chu, X. (2021). Automl: A survey of the state-of-the-art. *Knowl. Based Syst.*, 212:106622.

[135] Hern, A. (2018). Fitness tracking app strava gives away location of secret us army bases. *The Guardian*, 28.

[136] Hochbaum, D. S. (1997). Approximating covering and packing problems: Set cover, vertex cover, independent set, and related problems. In Hochbaum, D. S., editor, *Approximation Algorithms for NP-hard Problems*, pages 94–143. PWS Publishing Co., Boston, MA, USA.

[137] Hoplaros, D., Tari, Z., and Khalil, I. (2014). Data summarization for network traffic monitoring. *J. Network and Computer Applications*, 37:194–205.

[138] Hu, Z., Yang, J., and Zhang, J. (2018). Trajectory privacy protection method based on the time interval divided. *Computers & Security*, 77:488–499.

[139] Hua, J., Gao, Y., and Zhong, S. (2015). Differentially private publication of general time-serial trajectory data. In *Proc. INFOCOM*, pages 549–557. IEEE.

[140] Hube, N. and Müller, M. (2018). The data in your hands: Exploring novel interaction techniques and data visualization approaches for immersive data analytics. In *Proc. VisBIA*, pages 12–21, Castiglione della Pescaia, Italy.

[141] Huo, Z., Huang, Y., and Meng, X. (2011). History trajectory privacy-preserving through graph partition. In *Proc. MLBS*, pages 71–78. ACM.

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

[143] iConsulting S.p.A. (2020). Indyco wewb site. https://www.indyco.com/. [Online; accessed 21-June-2019].

[144] Inmon, W. H. (2005). *Building the data warehouse*. John wiley & sons.

[145] Jerbi, H., Ravat, F., Teste, O., and Zurfluh, G. (2009). Preference-based recommendations for OLAP analysis. In *Proc. DaWaK*, pages 467–478, Linz, Austria.

[146] Jiang, D., Leung, K. W., Vosecky, J., and Ng, W. (2014). Personalized query suggestion with diversity awareness. In *Proc. ICDE*, pages 400–411, Chicago, USA.

[147] Jiang, K., Shao, D., Bressan, S., Kister, T., and Tan, K. (2013). Publishing trajectories with differential privacy guarantees. In Szalay, A., Budavari, T., Balazinska, M., Meliou, A., and Sacan, A., editors, *Proc. SSDBM*, pages 12:1–12:12, Baltimore, MD, USA. ACM.

[148] Jin, F., Hua, W., Xu, J., and Zhou, X. (2019). Moving object linking based on historical trace. In *Proc. ICDE*, pages 1058–1069. IEEE.

[149] Jin, F., Hua, W., Zhou, T., Xu, J., Francia, M., Orowska, M., and Zhou, X. (2020). Trajectory-based spatiotemporal entity linking. *IEEE Transactions on Knowledge and Data Engineering*.

[150] Jin, R., Abu-Ata, M., Xiang, Y., and Ruan, N. (2008). Effective and efficient itemset pattern summarization: regression-based approaches. In *Proc. KDD*, pages 399–407.

[151] Kato, R., Iwata, M., Hara, T., Suzuki, A., Xie, X., Arase, Y., and Nishio, S. (2012). A dummy-based anonymization method based on user trajectory with pauses. In *Proc. SIGSPATIAL*, pages 249–258. ACM.

[152] Keim, D. A. (2002). Information visualization and visual data mining. *IEEE Trans. Vis. Comput. Graph.*, 8(1):1–8.

[153] Kelleher, J. D. and Tierney, B. (2018). *Data science*. MIT Press.

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

[155] Kido, H., Yanagisawa, Y., and Satoh, T. (2005). Protection of location privacy using dummies for location-based services. In *Proc. ICDE Workshops*, page 1248. IEEE Computer Society.

[156] Komishani, E. G., Abadi, M., and Deldar, F. (2016). PPTD: preserving personalized privacy in trajectory data publishing by sensitive attribute generalization and trajectory local suppression. *Knowl.-Based Syst.*, 94:43–59.

[157] Kong, X., Li, M., Ma, K., Tian, K., Wang, M., Ning, Z., and Xia, F. (2018). Big trajectory data: A survey of applications and services. *IEEE Access*, 6:58295–58306.

[158] Kopanaki, D., Theodossopoulos, V., Pelekis, N., Kopanakis, I., and Theodoridis, Y. (2016). Who cares about others' privacy: Personalized anonymization of moving object trajectories. In *Proc. EDBT*, pages 425–436. OpenProceedings.org.

[159] Kraska, T. (2018). Northstar: An interactive data science system. *Proc. VLDB*, 11(12):2150–2164.

[160] Krumm, J. and Rouhana, D. (2013). Placer: semantic place labels from diary data. In *Proc. UbiComp*, pages 163–172. ACM.

[161] Kuchmann-Beauger, N., Brauer, F., and Aufaure, M. (2013). QUASL: A framework for question answering and its application to business intelligence. In *Proc. RCIS*, pages 1–12. IEEE.

[162] Kuhn, H. W. (2010). The hungarian method for the assignment problem. In *50 Years of Integer Programming*, pages 29–47. Springer.

[163] Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T. M. T., Dousse, O., Eberle, J., and Miettinen, M. (2013). From big smartphone data to worldwide research: The mobile data challenge. *Pervasive and Mobile Computing*, 9(6):752–771.

[164] Lee, J., Han, J., Li, X., and Cheng, H. (2011). Mining discriminative patterns for classifying trajectories on road networks. *IEEE Trans. Knowl. Data Eng.*, 23(5):713–726.

[165] Lei, P., Peng, W., Su, I., and Chang, C. (2012). Dummy-based schemes for protecting movement trajectories. *J. Inf. Sci. Eng.*, 28(2):335–350.

[166] Leung, C. K. and Carmichael, C. L. (2009). FpVAT: a visual analytic tool for supporting frequent pattern mining. *SIGKDD Explorations*, 11(2):39–48.

[167] Li, F., Gao, F., Yao, L., and Pan, Y. (2016). Privacy preserving in the publication of large-scale trajectory databases. In *Proc. BigCom*, volume 9784 of *LNCS*, pages 367–376. Springer.

[168] Li, F. and Jagadish, H. V. (2016). Understanding natural language queries over relational databases. *SIGMOD Record*, 45(1):6–13.

[169] Li, M., Zhu, L., Zhang, Z., and Xu, R. (2017). Achieving differential privacy of trajectory data publishing in participatory sensing. *Inf. Sci.*, 400:1–13.

[170] Li, N., Li, T., and Venkatasubramanian, S. (2007). t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. ICDE*, pages 106–115. IEEE Computer Society.

[171] Lim, S. J. (2009). On a visual frequent itemset mining. In *Proc. ICDIM*, pages 46–51.

[172] Lin, M., Cao, H., Zheng, V. W., Chang, K. C., and Krishnaswamy, S. (2015). Mobile user verification/identification using statistical mobility profile. In *Proc. BIGCOMP*, pages 15–18, Jeju, South Korea.

[173] Liu, G., Zhang, H., and Wong, L. (2014). A flexible approach to finding representative pattern sets. *IEEE Trans. Knowl. Data Eng.*, 26(7):1562–1574.

[174] Liu, X., Chen, J., Xia, X., Zong, C., Zhu, R., and Li, J. (2019). Dummy-based trajectory privacy protection against exposure location attacks. In *Proc. WISA*, volume 11817 of *LNCS*, pages 368–381. Springer.

[175] Liu, X., Wang, L., and Zhu, Y. (2018). SLAT: sub-trajectory linkage attack tolerance framework for privacy-preserving trajectory publishing. In *Proc. NaNA*, pages 298–303. IEEE.

[176] Liu, X., Zhao, H., Pan, M., Yue, H., Li, X., and Fang, Y. (2012). Traffic-aware multiple mix zone placement for protecting location privacy. In *Proc. INFOCOM*, pages 972–980. IEEE.

[177] Llavori, R. B. and Nebot, V. (2015). *Context-Aware Business Intelligence*, pages 87–110. Springer.

[178] Lomonaco, V., Maltoni, D., and Pellegrini, L. (2019). Fine-grained continual learning. *CoRR*, abs/1907.03799.

[179] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., and Huang, Y. (2009). Map-matching for low-sampling-rate GPS trajectories. In *Proc. GIS*, pages 352–361, Seattle, Washington, USA. ACM.

[180] Luna, J. M., Fournier-Viger, P., and Ventura, S. (2019). Frequent itemset mining: A 25 years review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 9(6).

[181] Luo, A., Chen, S., and Xv, B. (2017). Enhanced map-matching algorithm with a hidden markov model for mobile phone positioning. *ISPRS Int. J. Geo-Inform.*, 6(11):327.

[182] Lyons, G., Tran, V., Binnig, C., Çetintemel, U., and Kraska, T. (2016). Making the case for query-by-voice with echoquery. In *Proc. SIGMOD*, pages 2129–2132, New York, NY, USA. ACM.

[183] Ma, C. Y. T., Yau, D. K. Y., Yip, N. K., and Rao, N. S. V. (2013). Privacy vulnerability of published anonymous mobility traces. *IEEE/ACM Trans. Netw.*, 21(3):720–733.

[184] Machanavajjhala, A., Gehrke, J., Kifer, D., and Venkitasubramaniam, M. (2006). l-diversity: Privacy beyond k-anonymity. In *Proc. ICDE*, page 24. IEEE Computer Society.

[185] Mahdavifar, S., Abadi, M., Kahani, M., and Mahdikhani, H. (2012). A clustering-based approach for personalized privacy preserving publication of moving object trajectory data. In *Proc. NSS*, volume 7645 of *LNCS*, pages 149–165. Springer.

[186] Mampaey, M. and Vreeken, J. (2013). Summarizing categorical data by clustering attributes. *Data Min. Knowl. Discov.*, 26(1):130–173.

[187] Manning, C. D. and Schütze, H. (2001). *Foundations of statistical natural language processing*. MIT Press.

[188] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, Baltimore, Maryland. The Association for Computer Linguistics.

[189] Manousakas, D., Mascolo, C., Beresford, A. R., Chan, D., and Sharma, N. (2018). Quantifying privacy loss of human mobility graph topology. *Proc. PoPETs*, 2018(3):5–21.

[190] Maouche, M., Mokhtar, S. B., and Bouchenak, S. (2017). Ap-attack: A novel user re-identification attack on mobility datasets. In *Proc. MobiQuitous*, pages 48–57, Melbourne, Australia.

[191] Marcel, P. and Negre, E. (2011). A survey of query recommendation techniques for data warehouse exploration. In *Proc. EDA*, pages 119–134, Clermont-Ferrand, France.

[192] Marmasse, N. and Schmandt, C. (2000). Location-aware information delivery with *ComMotion*. In *Proc. HUC*, pages 157–171, Bristol, UK.

[193] Mattos, E. P. D., Domingues, A. C. S. A., and Loureiro, A. A. F. (2019). Give me two points and i'll tell you who you are. In *Proc. IV*, pages 1081–1087. IEEE.

[194] McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *Proc. FOCS*, pages 94–103. IEEE Computer Society.

[195] Mendes, R. and Vilela, J. P. (2017). Privacy-preserving data mining: Methods, metrics, and applications. *IEEE Access*, 5:10562–10582.

[196] Meyerson, A. and Williams, R. (2004). On the complexity of optimal k-anonymity. In *Proc. PODS*, pages 223–228. ACM.

[197] Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

[198] Mir, D. J., Isaacman, S., Cáceres, R., Martonosi, M., and Wright, R. N. (2013). DP-WHERE: differentially private modeling of human mobility. In *Proc. BigData*, pages 580–588. IEEE Computer Society.

[199] Monreale, A., Andrienko, G. L., Andrienko, N. V., Giannotti, F., Pedreschi, D., Rinzivillo, S., and Wrobel, S. (2010). Movement data anonymity through generalization. *Trans. Data Privacy*, 3(2):91–121.

[200] Monreale, A., Pinelli, F., Trasarti, R., and Giannotti, F. (2009). Wherenext: a location predictor on trajectory pattern mining. In *Proc. KDD*, pages 637–646, Paris, France.

[201] Monreale, A., Trasarti, R., Pedreschi, D., Renso, C., and Bogorny, V. (2011). C-safety: a framework for the anonymization of semantic trajectories. *Trans. Data Privacy*, 4(2):73–101.

[202] Moosavi, S., Omidvar-Tehrani, B., Craig, R. B., and Ramnath, R. (2017). Annotation of car trajectories based on driving patterns. *CoRR*, abs/1705.05219.

[203] Moosavi, S., Ramnath, R., and Nandi, A. (2016). Discovery of driving patterns by trajectory segmentation. In Hoel, E. G. and Eldawy, A., editors, *Proc. SIGSPATIAL PhD Symposium*, pages 4:1–4:4, Burlingame, California, USA. ACM.

[204] Mu, X., Zhu, F., Lim, E., Xiao, J., Wang, J., and Zhou, Z. (2016). User identity linkage by latent user space modelling. In *Proc. KDD*, pages 1775–1784, San Francisco, CA, USA.

[205] Mulder, Y. D., Danezis, G., Batina, L., and Preneel, B. (2008). Identification via location-profiling in GSM networks. In *Proc. WPES*, pages 23–32, Alexandria, VA, USA.

[206] Naini, F. M., Unnikrishnan, J., Thiran, P., and Vetterli, M. (2016). Where you are is who you are: User identification by matching statistics. *IEEE Trans. Information Forensics and Security*, 11(2):358–372.

[207] Narayanan, A. and Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. In *Proc. IEEE Symposium on Security and Privacy*, pages 111–125, Oakland, California, USA.

[208] Narayanan, A. and Shmatikov, V. (2009). De-anonymizing social networks. In *Proc. IEEE Symposium on Security and Privacy*, pages 173–187, Oakland, California, USA.

[209] Negre, E., Ravat, F., Teste, O., and Tournier, R. (2013). Cold-start recommender system problem within a multidimensional data warehouse. In *Proc. RCIS*, pages 1–8, Paris, France.

[210] Nergiz, M. E., Atzori, M., and Saygin, Y. (2008). Towards trajectory anonymization: a generalization-based approach. In *Proc. SPRINGL*, pages 52–61. ACM.

[211] Newling, J. and Fleuret, F. (2017). A sub-quadratic exact medoid algorithm. In *Proc. AISTATS*, volume 54, pages 185–193. PMLR.

[212] Newson, P. and Krumm, J. (2009). Hidden markov map matching through noise and sparseness. In *Proc. GIS*, pages 336–343, Seattle, Washington, USA.

[213] Nguyen, L. T. T., Vu, V. V., Lam, M. T. H., Duong, T. T. M., Manh, L. T., Nguyen, T. T. T., Vo, B., and Fujita, H. (2019). An efficient method for mining high utility closed itemsets. *Inf. Sci.*, 495:78–99.

[214] Ostrovsky, R. and III, W. E. S. (2007). A survey of single-database private information retrieval: Techniques and applications. In Okamoto, T. and Wang, X., editors, *Proc. PKC*, volume 4450 of *LNCS*, pages 393–411, Beijing, China. Springer.

[215] Outa, F. E., Francia, M., Marcel, P., Peralta, V., and Vassiliadis, P. (2020a). Supporting the generation of data narratives. In Michael, J. and Torres, V., editors, *ER Forum, Demo and Posters 2020 co-located with 39th International Conference on Conceptual Modeling (ER 2020), Vienna, Austria, November 3-6, 2020*, volume 2716 of *CEUR Workshop Proceedings*, pages 168–172. CEUR-WS.org.

[216] Outa, F. E., Francia, M., Marcel, P., Peralta, V., and Vassiliadis, P. (2020b). Towards a conceptual model for data narratives. In Dobbie, G., Frank, U., Kappel, G., Liddle, S. W., and Mayr, H. C., editors, *Proc. ER*, volume 12400 of *Lecture Notes in Computer Science*, pages 261–270, Vienna, Austria. Springer.

[217] Ouyang, Y., Xu, Y., Le, Z., Chen, G., and Makedon, F. (2008). Providing location privacy in assisted living environments. In *Proc. PETRA*, volume 282 of *ACM International Conference Proceeding Series*, page 39. ACM.

[218] Palanisamy, B. and Liu, L. (2011). Mobimix: Protecting location privacy with mix-zones over road networks. In *Proc. ICDE*, pages 494–505. IEEE Computer Society.

[219] Pan, F., Cong, G., Tung, A. K. H., Yang, J., and Zaki, M. J. (2003). Carpenter: finding closed patterns in long biological datasets. In *Proc. KDD*, pages 637–642, Washington DC, USA.

[220] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proc. ICDT*, pages 398–416.

[221] Pedersen, T. B. (2009). Warehousing the world: A vision for data warehouse research. In Kozielski, S. and Wrembel, R., editors, *New Trends in Data Warehousing and Data Analysis*, volume 3 of *Annals of Information Systems*, pages 1–17. Springer.

[222] Peixoto, D. A., Hung, N. Q. V., Zheng, B., and Zhou, X. (2019). A framework for parallel map-matching at scale using spark. *Distributed Parallel Databases*, 37(4):697–720.

[223] Poernomo, A. K. and Gopalkrishnan, V. (2009). CP-summary: a concise representation for browsing frequent itemsets. In *Proc. KDD*, pages 687–696.

[224] Poosala, V. and Ioannidis, Y. E. (1997). Selectivity estimation without the attribute value independence assumption. In *Proc. VLDB*, pages 486–495, Athens, Greece.

[225] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulalas-Divanis, A. (2014). Apriori-based algorithms for k^m-anonymizing trajectory data. *Trans. Data Privacy*, 7(2):165–194.

[226] Primault, V., Boutet, A., Mokhtar, S. B., and Brunie, L. (2019). The long road to computational location privacy: A survey. *IEEE Commun. Surv. Tutorials*, 21(3):2772–2793.

[227] Pyrgelis, A., Troncoso, C., and Cristofaro, E. D. (2018). Knock knock, who's there? membership inference on aggregate location data. In *Proc. NDSS*. The Internet Society.

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

[229] Raedt, L. D. (2002). A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77.

[230] Riederer, C. J., Kim, Y., Chaintreau, A., Korula, N., and Lattanzi, S. (2016). Linking users across domains with location data: Theory and validation. In *Proc. WWW*, pages 707–719. ACM.

[231] Rizzi, S., Golfarelli, M., and Graziani, S. (2015). An OLAM operator for multidimensional shrink. *IJDWM*, 11(3):68–97.

[232] Rizzi, S. and Saltarelli, E. (2003). View materialization vs. indexing: Balancing space constraints in data warehouse design. In *Proc. CAiSE*, pages 502–519.

[233] Roberts, F. and Tesman, B. (2009). *Applied combinatorics*. Chapman and Hall/CRC.

[234] Rossi, L. and Musolesi, M. (2014). It's the way you check-in: identifying users in location-based social networks. In *Proc. COSN*, pages 215–226, Dublin, Ireland.

[235] Rossi, L., Walker, J., and Musolesi, M. (2015). Spatio-temporal techniques for user identification by means of GPS mobility data. *EPJ Data Sci.*, 4(1):11.

[236] Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.*, 40(2):99–121.

[237] Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education.

[238] Sadilek, A. and Krumm, J. (2012). Far out: Predicting long-term human mobility. In *Proc. AAAI*, Toronto, Ontario, Canada.

[239] Saha, A., Khapra, M. M., and Sankaranarayanan, K. (2018). Towards building large scale multimodal domain-aware conversation systems. In *Proc. AAAI*, pages 696–704. AAAI Press.

[240] Saha, D., Floratou, A., Sankaranarayanan, K., Minhas, U. F., Mittal, A. R., and Özcan, F. (2016). ATHENA: an ontology-driven system for natural language querying over relational data stores. *Proc. VLDB*, 9(12):1209–1220.

[241] Salvador, S. and Chan, P. (2004). Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Proc. ICTAI*, pages 576–584.

[242] Samarati, P. and Sweeney, L. (1998). Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. PODS*, page 188. ACM Press.

[243] Sarawagi, S. (1999). Explaining differences in multidimensional aggregates. In *Proc. VLDB*, pages 42–53.

[244] Sarawagi, S. (2000). User-adaptive exploration of multidimensional data. In *Proceedings of VLDB*, pages 307–316, Cairo, Egypt.

[245] Sathe, G. and Sarawagi, S. (2001). Intelligent rollups in multidimensional OLAP data. In *Proc. VLDB*, pages 531–540.

[246] Satopaa, V., Albrecht, J. R., Irwin, D. E., and Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *Proc. ICDCS*, pages 166–171.

[247] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. (1979). Access path selection in a relational database management system. In *Proc. SIGMOD*, pages 23–34, Boston, USA.

[248] Sen, J., Ozcan, F., Quamar, A., Stager, G., Mittal, A. R., Jammi, M., Lei, C., Saha, D., and Sankaranarayanan, K. (2019). Natural language querying of complex business intelligence queries. In *Proc. SIGMOD*, pages 1997–2000, Amsterdam, The Netherlands. ACM.

[249] Shao, D., Jiang, K., Kister, T., Bressan, S., and Tan, K. (2013). Publishing trajectory with differential privacy: A priori vs. A posteriori sampling mechanisms. In *Proc. DEXA*, volume 8055 of *LNCS*, pages 357–365. Springer.

[250] Shawi, R. E., Maher, M., and Sakr, S. (2019). Automated machine learning: State-of-the-art and open challenges. *CoRR*, abs/1906.02287.

[251] Shneiderman, B. (1992). Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99.

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

[253] Sicat, R., Li, J., Choi, J., Cordeil, M., Jeong, W., Bach, B., and Pfister, H. (2019). DXR: a toolkit for building immersive data visualizations. *IEEE Trans. Vis. Comput. Graph.*, 25(1):715–725.

[254] Song, C., Qu, Z., Blumm, N., and Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968):1018–1021.

[255] Song, W. and Liu, M. (2014). A visualizer for high utility itemset mining. In *Proc. CSE*, pages 244–248.

[256] Srivatsa, M. and Hicks, M. (2012). Deanonymizing mobility traces: using social network as a side-channel. In *Proc. CCS*, pages 628–637, Raleigh, NC, USA.

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

[258] Strong, D. M., Lee, Y. W., and Wang, R. Y. (1997). Data quality in context. *Commun. ACM*, 40(5):103–110.

[259] Stuart, D. (2015). The data revolution: Big data, open data, data infrastructures and their consequences. *Online Inf. Rev.*, 39(2):272.

[260] Su, Y. and Grauman, K. (2016). Detecting engagement in egocentric video. In *Proc. ECCV*, pages 454–471.

[261] Sweeney, L. (2000). Simple demographics often identify people uniquely. *Health*, 671:1–34.

[262] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 10(5):557–570.

[263] Takigawa, I. and Mamitsuka, H. (2011). Efficiently mining δ-tolerance closed frequent subgraphs. *Machine Learning*, 82(2):95–121.

[264] Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., and Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9-12):3563–3576.

[265] Tarjan, R. E. (1972). Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160.

[266] Terrovitis, M., Poulis, G., Mamoulis, N., and Skiadopoulos, S. (2017). Local suppression and splitting techniques for privacy preserving publication of trajectories. *IEEE Trans. Knowl. Data Eng.*, 29(7):1466–1479.

[267] Terrovitis, M., Vassiliadis, P., Skiadopoulos, S., Bertino, E., Catania, B., Maddalena, A., and Rizzi, S. (2007). Modeling and language support for the management of pattern-bases. *Data Knowl. Eng.*, 62(2):368–397.

[268] Thollot, R., Kuchmann-Beauger, N., and Aufaure, M. (2012). Semantics and usage statistics for multi-dimensional query expansion. In *Proc. DASFAA*, pages 250–260, Busan, South Korea.

[269] Trummer, I., Wang, Y., and Mahankali, S. (2019). A holistic approach for query evaluation and result vocalization in voice-based OLAP. In *Proc. SIGMOD*, pages 936–953, Amsterdam, The Netherlands. ACM.

[270] Tu, Z., Zhao, K., Xu, F., Li, Y., Su, L., and Jin, D. (2019). Protecting trajectory from semantic attack considering k-anonymity, l-diversity, and t-closeness. *IEEE Trans. Network and Service Management*, 16(1):264–278.

[271] Ukkonen, E. (1985). Finding approximate patterns in strings. *J. Algorithms*, 6(1):132–137.

[272] van der Aalst, W. (2016). *Data Science in Action*, pages 3–23. Springer Berlin Heidelberg, Berlin, Heidelberg.

[273] Vanahalli, M. K. and Patil, N. (2019). An efficient parallel row enumerated algorithm for mining frequent colossal closed itemsets from high dimensional datasets. *Inf. Sci.*, 496:343–362.

[274] Vartak, M., Rahman, S., Madden, S., Parameswaran, A. G., and Polyzotis, N. (2015). SEEDB: efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13):2182–2193.

[275] Vassiliadis, P., Marcel, P., and Rizzi, S. (2019). Beyond roll-up's and drill-down's: An intentional analytics model to reinvent OLAP. *Inf. Syst.*, 85:68–91.

[276] Vieira, M. R., Razente, H. L., Barioni, M. C. N., Hadjieleftheriou, M., Srivastava, D., Jr., C. T., and Tsotras, V. J. (2011). On query result diversification. In *Proc. ICDE*, pages 1163–1174, Hannover, Germany.

[277] Vitali, G., Francia, M., Golfarelli, M., and Canavari, M. (2021). Crop management with the iot: An interdisciplinary survey. *Agronomy*, 11(1):181.

[278] Wang, H., Gao, C., Li, Y., Wang, G., Jin, D., and Sun, J. (2018). De-anonymization of mobility trajectories: Dissecting the gaps between theory and practice. In *Proc. NDSS*. The Internet Society.

[279] Wang, J. and Karypis, G. (2006). On efficiently summarizing categorical databases. *Knowl. Inf. Syst.*, 9(1):19–37.

[280] Watts, S., Shankaranarayanan, G., and Even, A. (2009). Data quality assessment in context: A cognitive perspective. *Decis. Support Syst.*, 48(1):202–211.

[281] Westin, A. F. (1968). Privacy and freedom. *Washington and Lee Law Review*, 25(1):166.

[282] Wu, X. and Sun, G. (2014). A novel dummy-based mechanism to protect privacy on trajectories. In *Proc. ICDM Workshops*, pages 1120–1125. IEEE Computer Society.

[283] Xiang, Y., Jin, R., Fuhry, D., and Dragan, F. F. (2011). Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Discov.*, 23(2):215–251.

[284] Xin, D., Han, J., Yan, X., and Cheng, H. (2005). Mining compressed frequent-pattern sets. In *Proc. VLDB*, pages 709–720.

[285] Xu, F., Tu, Z., Li, Y., Zhang, P., Fu, X., and Jin, D. (2017). Trajectory recovery from ash: User privacy is NOT preserved in aggregated mobility data. In *Proc. WWW*, pages 1241–1250. ACM.

[286] Xu, K., Zhu, M., Zhang, D., and Gu, T. (2008). Context-aware content filtering & presentation for pervasive & mobile information systems. In *Proc. ICST*, page 20, Quebec, Canada.

[287] Xue, A. Y., Zhang, R., Zheng, Y., Xie, X., Huang, J., and Xu, Z. (2013). Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proc. ICDE*, pages 254–265. IEEE Computer Society.

[288] Xue, W., Pung, H. K., Palmes, P. P., and Gu, T. (2008). Schema matching for context-aware computing. In *Proc. UbiComp*, pages 292–301, Seoul, Korea.

[289] Yaghmazadeh, N., Wang, Y., Dillig, I., and Dillig, T. (2017). Sqlizer: query synthesis from natural language. *PACMPL*, 1(OOPSLA):63:1–63:26.

[290] Yan, X., Cheng, H., Han, J., and Xin, D. (2005). Summarizing itemset patterns: a profile-based approach. In *Proc. KDD*, pages 314–323.

[291] Yan, X., Guo, J., and Cheng, X. (2011). Context-aware query recommendation by learning high-order relation in query logs. In *Proc. CIKM*, pages 2073–2076, Glasgow, United Kingdom.

[292] Yang, L. (2005). Pruning and visualizing generalized association rules in parallel coordinates. *IEEE Trans. Knowl. Data Eng.*, 17(1):60–70.

[293] Yang, Z., Zhong, S., and Wright, R. N. (2005). Anonymity-preserving data collection. In *Proc. KDD*, pages 334–343. ACM.

[294] Yao, L., Wang, X., Wang, X., Hu, H., and Wu, G. (2019). Publishing sensitive trajectory data under enhanced l-diversity model. In *Proc. MDM*, pages 160–169. IEEE.

[295] Yarovoy, R., Bonchi, F., Lakshmanan, L. V. S., and Wang, W. H. (2009). Anonymizing moving objects: how to hide a MOB in a crowd? In *Proc. EDBT*, volume 360 of *ACM International Conference Proceeding Series*, pages 72–83. ACM.

[296] You, S., Zhang, J., and Gruenwald, L. (2015). Large-scale spatial join query processing in cloud. In *Proc. ICDE Workshops*, pages 34–41, Seoul, South Korea.

[297] You, T., Peng, W., and Lee, W. (2007). Protecting moving trajectories with dummies. In *Proc. MDM*, pages 278–282. IEEE.

[298] Yu, J., Zhang, Z., and Sarwat, M. (2018). Spatial data management in apache spark: the geospark perspective and beyond. *GeoInformatica*, pages 1–42.

[299] Yu, W. (2019). Discovering frequent movement paths from taxi trajectory data using spatially embedded networks and association rules. *IEEE Trans. Intelligent Transportation Systems*, 20(3):855–866.

[300] Yu, Y., Cao, L., Rundensteiner, E. A., and Wang, Q. (2014). Detecting moving object outliers in massive-scale trajectory streams. In Macskassy, S. A., Perlich, C., Leskovec, J., Wang, W., and Ghani, R., editors, *Proc. KDD*, pages 422–431, New York, NY, USA. ACM.

[301] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proc. NSDI*, pages 15–28, San Jose, CA, USA.

[302] Zaki, F. A. M. and Zulkurnain, N. F. (2018). RARE: mining colossal closed itemset in high dimensional data. *Knowl.-Based Syst.*, 161:1–11.

[303] Zang, H. and Bolot, J. (2011). Anonymization of location data does not work: a large-scale measurement study. In *Proc. MobiCom*, pages 145–156. ACM.

[304] Zeidan, A., Lagerspetz, E., Zhao, K., Nurmi, P., Tarkoma, S., and Vo, H. T. (2018). Geomatch: Efficient large-scale map matching on apache spark. In *Proc. BigData*, pages 384–391, Seattle, WA, USA.

[305] Zhang, K. and Shasha, D. E. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.

[306] Zhang, S., Jin, Z., and Lu, J. (2010). Summary queries for frequent itemsets mining. *Journal of Systems and Software*, 83(3):405–411.

[307] Zhang, X. and Deng, Z. (2015). Mining summarization of high utility itemsets. *Knowl.-Based Syst.*, 84:67–77.

[308] Zhao, X., Dong, Y., and Pi, D. (2019). Novel trajectory data publishing method under differential privacy. *Expert Syst. Appl.*, 138.

[309] Zheng, K., Wang, H., Qi, Z., Li, J., and Gao, H. (2017). A survey of query result diversification. *Knowl. Inf. Syst.*, 51(1):1–36.

[310] Zheng, Y. (2015). Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3):29:1–29:41.

[311] Zheng, Y., Zhang, L., Ma, Z., Xie, X., and Ma, W. (2011). Recommending friends and locations based on individual location history. *ACM Trans. Web*, 5(1):5:1–5:44.

[312] Zheng, Y., Zhang, L., Xie, X., and Ma, W. (2009). Mining interesting locations and travel sequences from GPS trajectories. In Quemada, J., León, G., Maarek, Y. S., and Nejdl, W., editors, *Proc. WWW*, pages 791–800, Madrid, Spain. ACM.

[313] Zhong, Y., Yuan, N. J., Zhong, W., Zhang, F., and Xie, X. (2015). You are where you go: Inferring demographic attributes from location check-ins. In *Proc. WSDM*, pages 295–304, Shanghai, China.

[314] Zhou, B., Pei, J., and Luk, W. (2008). A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorations*, 10(2):12–22.

[315] Zhou, C., Frankowski, D., Ludford, P. J., Shekhar, S., and Terveen, L. G. (2004). Discovering personal gazetteers: an interactive clustering approach. In *Proc. GIS*, pages 266–273, Washington, DC, USA.

[316] Zihayat, M. and An, A. (2014). Mining top-k high utility patterns over data streams. *Inf. Sci.*, 285:138–161.