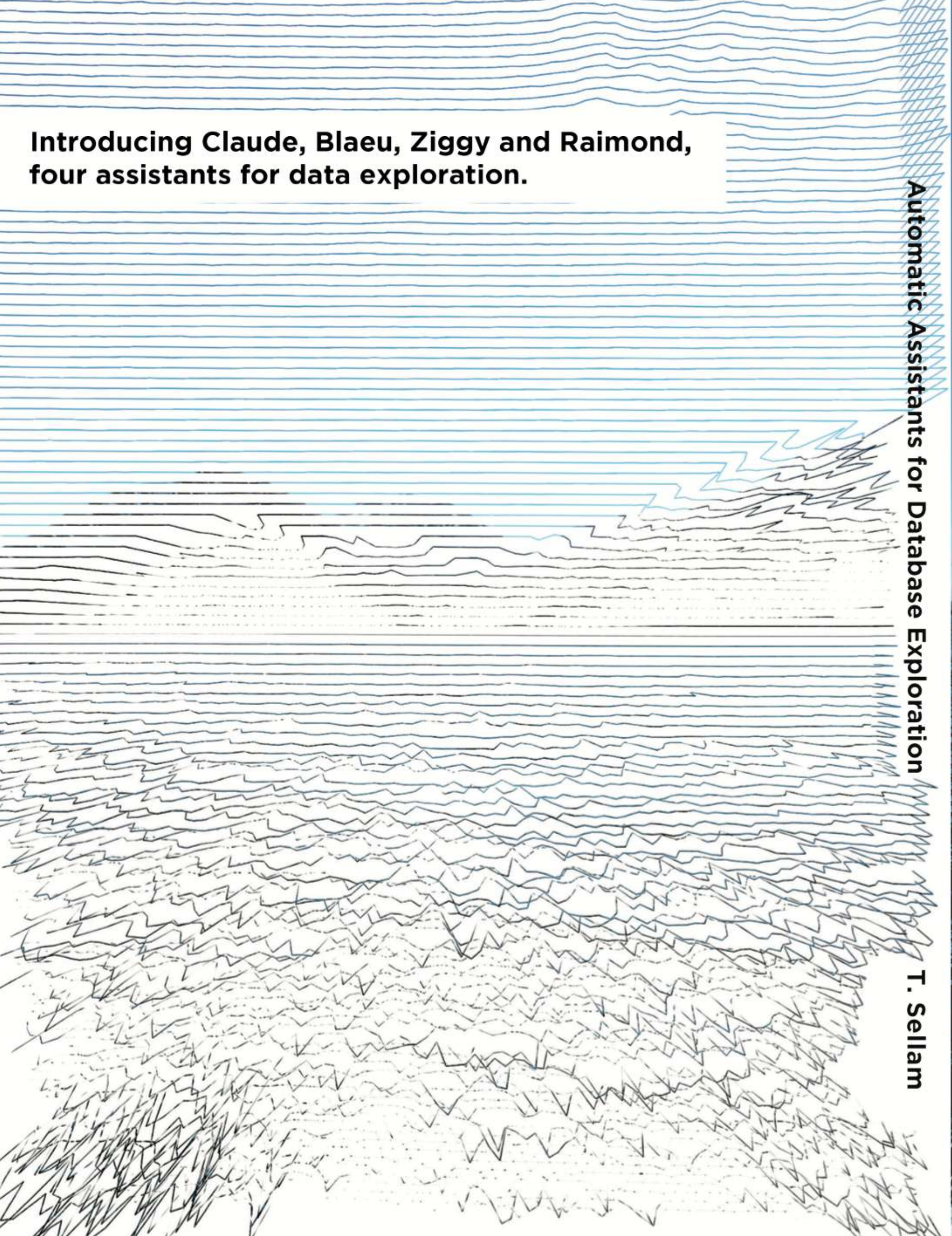
Automatic assistants for database exploration

Sellam, T.H.J.

Publication date
2016

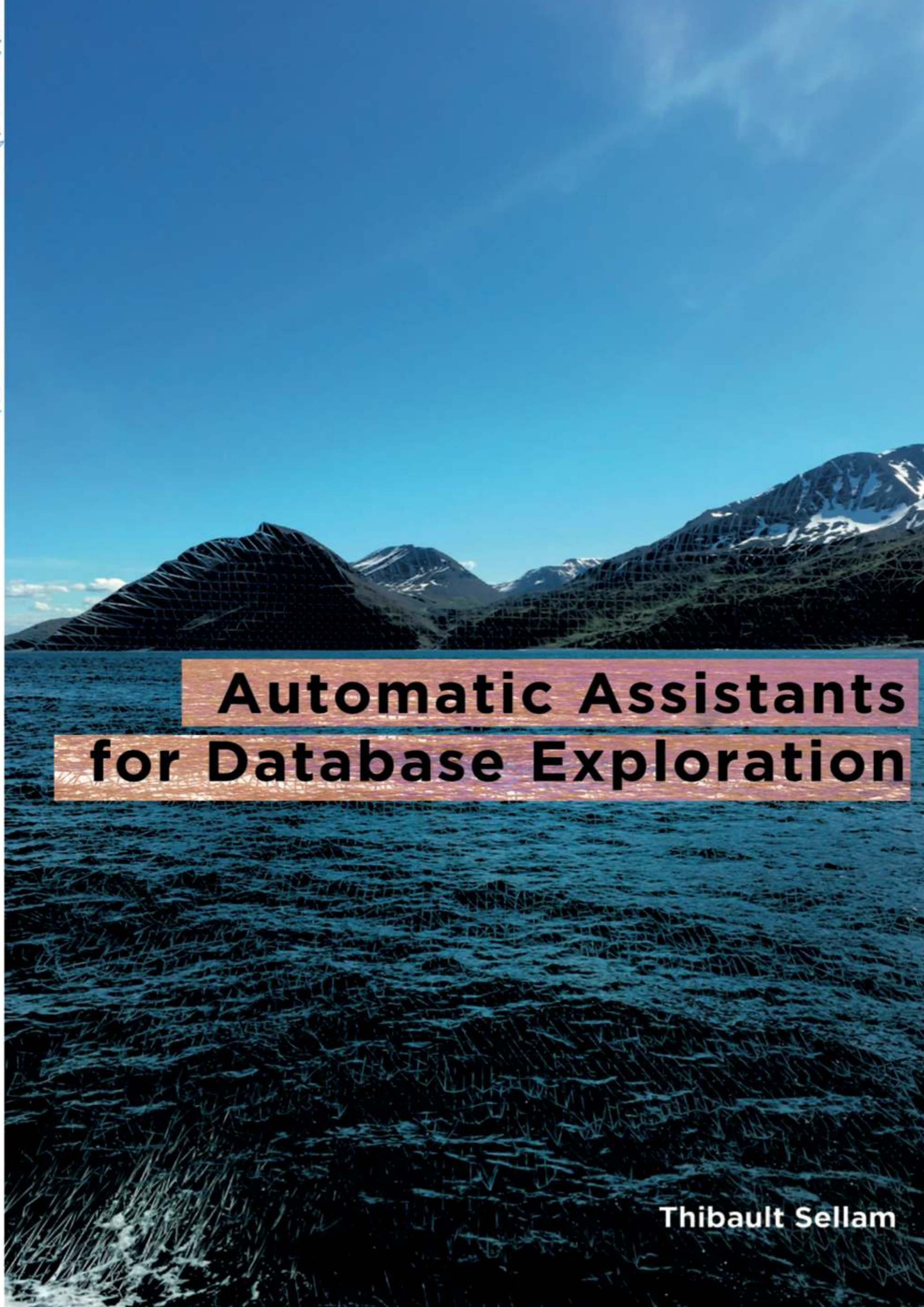
Document Version
Final published version



Introducing Claude, Blaeu, Ziggy and Raimond, four assistants for data exploration.

Automatic Assistants for Database Exploration

T. Sellam



Automatic Assistants for Database Exploration

Thibault Sellam

Automatic Assistants for Database Exploration

Thibault Sellam

Automatic Assistants for Database Exploration

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op donderdag 3 november 2016, te 10:00 uur

door

Thibault Henri Joseph Sellam

geboren te Créteil, Frankrijk.

Promotor:	Prof. dr. M.L. Kersten	Universiteit van Amsterdam
Overige Leden:	Prof. dr. M. de Rijke	Universiteit van Amsterdam
	Prof. dr. P.W. Adriaans	Universiteit van Amsterdam
	Prof. dr. M. Worring	Universiteit van Amsterdam
	Prof. dr. G. Weikum	Max Planck Inst. Informatics
	Prof. dr. B. Goethals	Universiteit Antwerpen

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica

CWI
Centrum Wiskunde & Informatica

Most of the research reported in this thesis was carried out at CWI, the Dutch National Center for Mathematics and Computer Science.

COMMIT/

The research reported in this thesis was supported by the Dutch national program COMMIT.

Microsoft

Part of the research reported in this thesis was carried out at Microsoft Research, in Mountain View, CA.

SIKS

SIKS Dissertation Series No. 2016-44
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN: 9789462955202
Cover design by Noki Powlonski

Acknowledgments

I am deeply grateful to the many people who supported me during my doctoral studies. First and foremost, I am indebted to prof. dr. Martin Kersten, who advised, supported and challenged me energetically during those five years. I would need an entire new Acknowledgment section to list all that he taught me. I would also like to thank prof. dr. Stefan Manegold for his herculean efforts to create a productive and secure scientific environment. I was also glad and proud to work with the MonetDB team. In particular, I would like to acknowledge Hannes and Holger, whom I now consider as friends. I was lucky to work alongside Mark and Benno, as well as Yagiz, Mrunal, Vamsi and Lefteris. And I wish to express my admiration and affection for CWI - admiration, because I hold its scientific activities in high regard, and affection because I spent a few of the happiest years of my life as an employee of this institution.

A number of people outside CWI directly contributed to the work described in this thesis. In particular, I thank dr. Omar Alonso, who gave me my chance as a Microsoft intern, coached me and introduced me to the life in the Silicon Valley. I am grateful to prof. dr. Emmanuel Müller, for taking me as a collaborator and for providing valuable feedback. I would also like to thank the members my PhD committee, prof. dr. Maarten de Rijke, prof. dr. Peter Adriaans, prof. dr. Marcel Worring, prof. dr. Gerard Weikum and prof. dr. Bart Goethals. Finally, I appreciated the kind support of prof. dr. Arjen de Vries and prof. dr. Arno Siebes, with whom I would be glad to collaborate in the future.

Between January and September 2015, I interrupted my research activities to join a music tour. I am thankful to Martin and Stefan, but also prof. dr, Lynda Hardman, Irma van Lunenburg and Karin van Gemert for supporting this decision and making this adventure possible without sacrificing my scientific career. And of course, I am deeply thankful to the FAUVE Corp. for taking me on board. In particular, I thank Q and Pitou, who have inspired me and given me self-confidence for many years. I also thank Siz, Noki and Prez, who have been amazing travel companions.

Finally, I would like to acknowledge my family and friends for their support. I thank my mother and father, who are awesome parents. I also thank my siblings, Cécile, Blandine and Léo, who will certainly be embarrassed to read about my affection towards them. Finally, I wish to acknowledge a few of the close friends who supported me: Steven, Halldora, Durex, Chloé, Marc and especially Sophie.

Contents

Chapter 1

Introduction

1 The Big Data Era: Hype and Reality

“We live in the Big Data era”. So could this thesis have started if we were to believe many recent press titles. Since 2008, Big Data appeared in *Nature* [34], *the Economist* [94], *the Harvard Business Review* [8], *the Wall Street Journal* [100], *Wired* [7], *The New York Times* [58] and truckloads of others. The expression Big Data describes the idea that businesses, scientists and public administrations can perform tasks with large amounts of data that would not have been possible otherwise. Journalists are often enthusiastic, sometimes hyperbolic. According to many, it is “a revolution that will transform how we live, work, and think” [62]. And this enthusiasm has spread beyond media. Figure 1.1 shows the number of Google searches containing the terms “Big Data” from all around the world. From 2012 on, the popularity of the term explodes: the number of searches quadruples in 4 years.

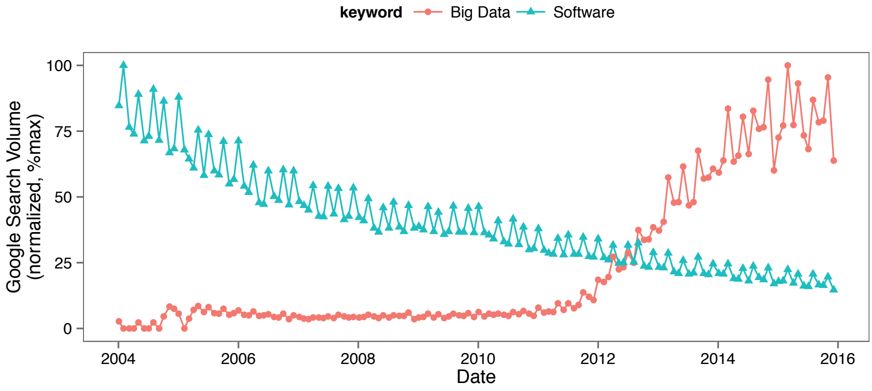


Figure 1.1: Number of Google searches for the terms “Big Data” and “Software”, from Google Trends.

And yet, the promises of Big Data do not convince everyone. In March 2014, the *Financial Times* titled one article “Big data: are we making a big mistake?” [37]. A month later, the *New York Times* was reporting “Eight (No, Nine!) Problems with Big Data” [61]. These articles argue that Big Data is a vacuous term, used by companies to appear innovative. And indeed, no one seems to agree about which datasets are “big” and which are not [40]. Furthermore, skeptics point out that data analysts are too naive. For instance, Chris Anderson of Wired attracted lots of attention when he predicted that Big Data would make the scientific method “obselete”, because “with enough data, the numbers speak for themselves” [7]. This claim was bold, but probably wrong. The size of a dataset does not eliminate its biases [54]. In fact, it increases the chance of making spurious findings [45].

Our view is that the need for efficient techniques to manage data goes well beyond the Big Data phenomenon. Businesses and scientists have had access to data for decades, probably centuries. The quest for better tools to store it, query it, process it and visualize it is still ongoing and relevant. To illustrate, let us get back to Figure 1.1. What interests us now is not the content of the chart, but how we made it. To create this visualization, we queried the website Google Trends, exported the results in comma-separated format, cleaned the file with command line tools, loaded the result in the statistical package R. We then applied grouping operations to reduce the granularity of the data, plotted the result with a third party charting library and cropped the file with an image editor. These operations involved four different interfaces, and took about half an hour. For our sake and that of all future researchers, we hope that advances in data management will not stop here.

2 Queries, Results and Exploration

In this thesis, we focus on *structured data*, that is, tables stored in a Database Management Systems (DBMS). To manipulate those, engineers have come up with the *query-result paradigm.* Users interrogate the system, specifying which subset of the data they are interested in, and what to do with it. The system replies with the results, as accurately and quickly as possible [69]. The most popular language to carry out this task has long been SQL, but alternatives exist. For instance, several Microsoft products have supported Query-By-Example [112], a language for non-programmers. With Tableau [88], users interrogate their data warehouses with drag-and-drops and inspect their results with sophisticated visualizations.

In its current implementations, the query-result paradigm relies on a strong hypothesis: it assumes that users know exactly what they want, and how to get it. Query languages expect precise instructions, such as "give me all the rows of table `Country` for which the value of `CountryID` is `NL`", or "give me the last 20 rows of my file". But there exists an important use case for which this assumption does not hold: **data exploration**.

> *Data exploration* is the task of querying a dataset to increase the knowledge that we have of it.

Typically, users explore a database when they first encounter it, in order to develop an intuition of what it contains. This task is a challenge because it is subject to a reciprocal dependency: on one hand, users need to pose queries to know the data. On the other hand, they need to know the data to pose queries. Therefore, explorers operate by trial and error. They start with simple, naive queries, to get an overview of the dataset. Then, as their knowledge increases, their queries become more specific.

The exploration process is rarely structured. Most users "play" with their DBMS, or they "tweak" it, to build a mental representation of its content [1]. This approach is problematic because it is completely manual: it depends entirely on patience, intuition and good luck. If the data is small, trial and error may be sufficient. But how to deal with hundreds of columns and hundreds of thousands of tuples? Manual effort is tedious, time consuming, and subject to errors. Users need more systematic tools. This leads us to our research problem:

> How can we provide automatic support for data exploration?

Our aim is to develop techniques and software tools to help users discover their data in a quick, easy and thorough fashion. Ultimately, we envision a system to answer *the* question behind data analysis:

> *"Computer, what is interesting in my data?"*

Data exploration is important because it is ubiquitous. Tools to support this task would find applications in virtually all fields which involve a database. They would help employees from large companies, whose databases are often complex and undocumented. They would uncover new investigation material for journalists. And they would serve scientists seeking inspiration. After all, few researchers know what they are after before they actually have found it.

3 How Hard is Our Problem?

Unfortunately, automating data exploration is arduous, maybe even unfeasible. Indeed, this task is subject to two fundamental contradictions:

Contradiction 1. Exploration is a subjective process: what is interesting for a user may be boring for the other. But then, how can we automate such a subjective activity?

Contradiction 2. Exploration is an ad hoc, *open ended* process. The portion of the database involved in a discovery can take any form: a tuple, a set of columns, a correlation or even a column name. How can we engineer a closed, systematic solution for such an open task?

These contradictions are fundamental because they rule out *full automation.* At this point, we simply cannot design a "magic" autonomous system, which would make discoveries while the user waits. To solve the first paradox, we would need to read our users' mind. To solve the second, we would need to develop a universal query language, which would be flexible enough to adapt to any human requirement. Those are still distant scientific prospects.

4 Our Contribution: Exploration Assistants

Our solution is to provide a middle ground between full automation and manual effort. We present **virtual assistants**, to explore data in a semi-automatic fashion. Each assistant relies on a *user model*, that is, a formalized set of assumptions about the explorer's interests for a predefined scenario. From these models, our systems make recommendations, collect feedback and react accordingly. Thus, they invite users to a "discussion", involving database queries, pattern analysis and visualization. Let us introduce each system:

- Our first assistant, **Claude**, excels at detecting the relationships between the columns in a data warehouse. To interrogate Claude, users specify a variable in which they are interested (e.g., `Profit` in a marketing context, or `Salary` for a census). In response, Claude suggests database views, highlighting the columns and tuples which strongly influence this variable.

- Our second assistant, **Blaeu**, is a cartography expert: it creates *maps*, to summarize the content of the database. These maps are interactive: users can zoom in, project, or highlight properties of interest. Through these actions, explorers can *browse* their data, and discover potentially interesting Select-Project-Join queries.

- **Ziggy** completes Claude and Blaeu's suggestions. Its aim is to help users who already have a query, but do not know what is interesting about it. Our system can pinpoint what makes a set of tuples "special", by highlighting its differences with the rest of the database.
- Finally, **Raimond** generalizes our approaches to non-structured data. Raimond targets *microblogs* (e.g., tweets), written in natural language. From those, it can extract quantitative data and organize its findings with thematic timelines. To demonstrate Raimond, we focus on news events. We show that generalizing semi-automatic data exploration beyond tables is feasible.

Our work relies heavily on *machine learning*. We will show that many methods from this field are applicable to data exploration, but they are too heavy in terms of user involvement. Our explorers are not statisticians, and they may have neither the patience nor the skills to tune statistical inference algorithms. Hence, an important contribution of this thesis is to provide new statistical techniques, focusing on interpretability, convenience and speed rather than strict accuracy.

5 Structure and Covered Publications

We present the background material necessary to understand this thesis in the next three chapters. Chapter 2 focuses on data warehouses and visual analytics. Chapter 3 deals with data mining. Chapter 4 presents elementary notions of information theory. We then dedicate one section to each assistant. In Chapter 5, we describe Claude, based on the following paper:

- **Semi-Automated Exploration of Data Warehouses**
  Thibault Sellam, Emmanuel Müller, Martin Kersten
  ACM Conference on Information and Knowledge Management (CIKM), Knowledge Management track, 2015

In Chapter 6, we present Blaeu. This chapter is based on the following three papers:

- **Meet Charles, Big Data Query Advisor**
  Thibault Sellam, Martin Kersten
  Conference on Innovative Data Research (CIDR), 2013

- **Cluster-Driven Navigation of the Query Space**
  Thibault Sellam, Martin Kersten
  IEEE Transactions on Knowledge and Data Engineering (TKDE), Submitted and accepted in 2015

- **Blaeu: Mapping and Navigating Large Tables with Cluster Analysis**
  Thibault Sellam, Robin Cijvat, Richard Koopmanschap and Martin Kersten
  Very Large Databases (VLDB), demonstration track, 2016

Ziggy, presented in chapter 7, is based on the following papers:

- **"Hey Ziggy, What Am I Looking At?" - Describing Tuples for Data Explorers**
  Thibault Sellam, Martin Kersten
  Scientific and Statistical Database Management (SSDBM), 2016

- **Ziggy: Characterizing Query Results for Data Explorers**
  Thibault Sellam, Martin Kersten
  Very Large Databases (VLDB), demonstration track, 2016

- **Have a Chat with Clustine, Conversational Engine to Query Large Tables**
  Thibault Sellam, Martin Kersten
  Workshop on Human in the Loop Data Analytics (co-located with SIGMOD), 2016

In Chapter 8, we describe Raimond. Raimond was developed during an internship at Microsft Research (Mountain View, CA) and was patented. We describe Raimond in this paper:

- **Raimond: Quantitative Data Extraction from Twitter to Describe Events**
  Thibault Sellam, Omar Alonso
  International Conference on Web Engineering (ICWE), 2015

In Chapter 9, we confront our findings with related work, conclude, and present future research directions.

Chapter 2

Background 1: Data Warehouses and Visualization

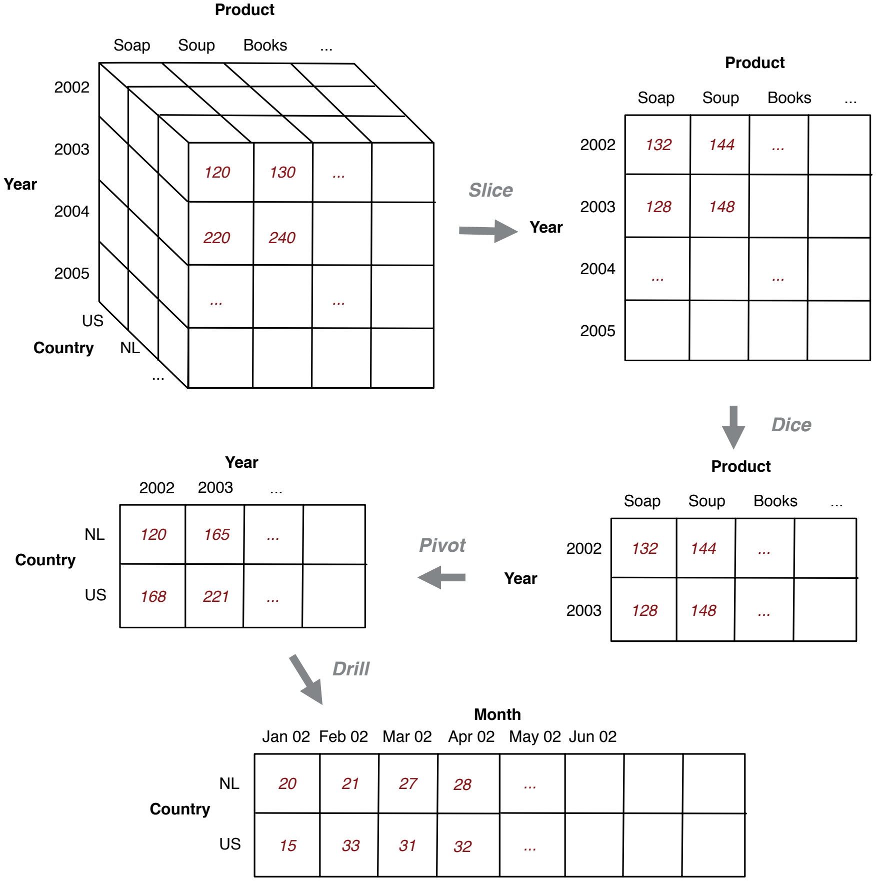
Relational database systems appeared in the 70's, to support operational business tasks. Typical use cases included storing client information or processing bank operations. The ability to update tables concurrently and consistently was critical and transaction throughput was the key measure of success. This model is still widely in used today, and it is referred to as *OLTP*. In the 80's and 90's, businesses began to aggregate their databases across departments, to get a strategic view of their resources and results. They set up *data warehouses*, that is, large data repositories to contain those aggregates [21]. For software engineers, data warehouses posed a whole world of new challenges. For a start, the queries involved more data than in the OLTP world. For instance, managers would be interested in the sales of a whole semester, rather than that of a unique transaction. Furthermore, these queries were more difficult to optimize, as they involved complex aggregations and arithmetic operations. Lastly, data warehouses were meant to be queried by humans, not by automated processes. In this context, the couple formed by SQL queries and tabular results was not optimal anymore: SQL was too rigid, and raw tuples were overwhelming. Software editors and researchers had to invent new interaction methods. In this chapter, we review two of those: data cubes and visual analytics. Both of these approaches aim at *summarizing* the data. The first uses aggregates, while the second uses multivariate visualizations.

1 The OLAP model, data cubes and pivot tables

The OLAP model first appeared in the database system Arbor Essbase, in the early 90's. Its aim is to provide a generic mechanism to manipulate and summarize multidimensional data. Roughly, this model relies on two concepts: the *data cube*, which is an abstract model of the database, and

Country	Year	Product	*Sales*
US	2002	Soap	*212*
US	2003	Soap	*312*
NL	2002	Soup	*205*
NL	2003	Soup	*303*
...	...	...	...

(a) Tabular view.



(b) Data cube view.

Figure 2.1: Two abstractions for a multidimensional dataset.

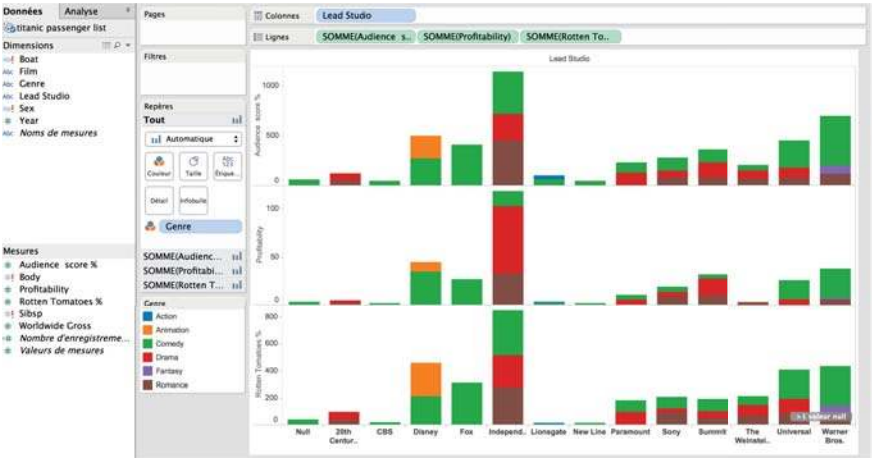


Figure 2.2: A screenshot of Tableau 9.2 (with permission from Tableau Software, 2015).

a set of primitives to manipulate this cube. Let us present those concepts through an example. Figure 2.1a shows fictive marketing data, in tabular form. According to the OLAP model, this dataset contains two types of columns: the *measure*, Sales, and three *dimensions*, Country, Year and Product. The measure is the variable that we want to analyze. The dimensions describe the context in which it varies. The top-left part of Figure 2.1b presents the data cube corresponding to this set. Observe that the three dimensions define a grid, in which each cell contains a value of the measure. The OLAP model provides us with (at least) four primitives to inspect this cube: we can *slice*, *dice*, *pivot*, or *drill*. Slicing lets us project the cube on a subset of its dimensions. For instance, we can select a two dimension view of the cube, to visualize it (as we did in the figure). By dicing, we restrict the range of the dimensions. With a pivot, we replace one dimension by another. Finally, drilling lets us inspect the data at a thinner level of granularity. The data cube is a pure abstraction. It reflects in no way how the data is stored physically. Neither does it reflect the logical schema of the database (i.e., at the SQL level). It is merely an additional layer, meant to facilitate the navigation of data warehouses.

In principle, we could manipulate data cubes directly through SQL. Indeed, most OLAP primitives have a direct equivalent in this language. But in practice, many users prefer graphical front-ends. Historically, the tools of choice were spreadsheet systems; first with Lotus 1-2-3, then with

Microsoft Excel. In fact, these systems had already introduced *pivot tables* in the late 80's, which essentially are light data cubes. But alternatives have emerged. Nowadays, a dozen software editors offer so-called Business Intelligence dashboards, which exploit modern graphical methods. Popular products include QlikView, IBM Cognos or Microsoft BI. Figure 2.2 presents a screenshot of Tableau [88], which has gained a large success in both academia and industry.

2 Visual Analytics

In parallel to the development of the OLAP model, experts in computer graphics have introduced *visual analytics* software, such as XmdvTool [101], GGobi [90], and ScatterDice [30]. These tools have the same objective as data cubes: they help users query and visualize large datasets. However, their logic is different. In the OLAP world, users interact with the data through aggregates and statistics. They manipulate computed summaries, they never access their data in its raw form. Visual analytics provide a more direct access to the database. They present the original values, not aggregates. To do so, they map the tuples to *visual metaphors*, which exploit the user's visual bandwidth to its maximum. Also, they provide commands to interact with the resulting displays: typically, users can zoom, pan, or select. We now present four families of visualization techniques, on which much of visual analytics rely: *pixel-oriented* methods, *icon-based* displays, *geometric projections*, and *hierarchical* techniques [47].

Pixel oriented visualizations. Pixel oriented visualizations map each value from the database to a colored area. As an illustration, Figure 2.3 represents the Iris dataset, a database of flowers from the UCI repository [10]. Each panel represents a variable. Within the panels, the rectangles represent tuples, organized in a round robin fashion. Observe the flag-like structure of the two rightmost displays: we perceive three clusters, which correspond the three types of flowers present in the database.

Icon-based visualizations. Figure 2.4 presents an example of icon-based visualization. This method maps each tuple to an icon, which shape represents the data values. In the example, each variable is mapped to a star, and each column is represented by the length of one axis. This techniques is efficient to compare items and detect micro-variations, but it gives a poor view of the overall structure of the data.

Projection-based methods. One drawback of pixel-oriented and icon-based methods is that they do not actually show the multidimensional space where the data lives. Projection-based methods address this issue.

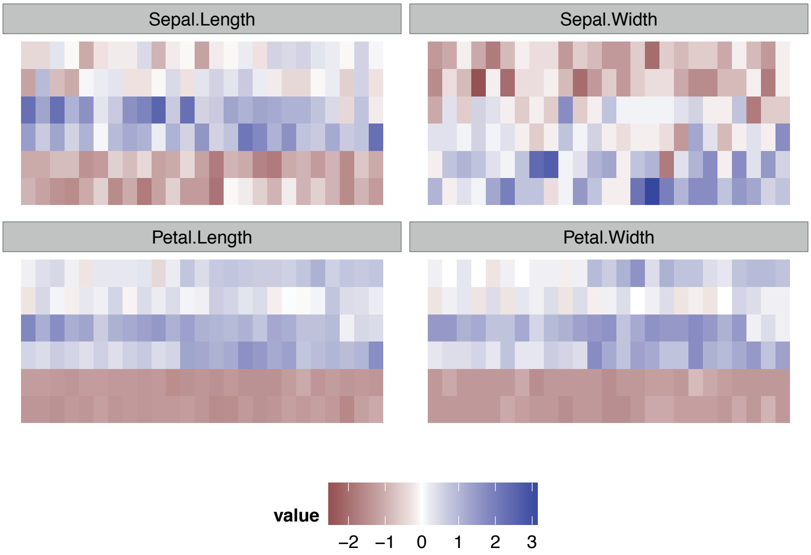


Figure 2.3: Pixel Oriented view of the Iris dataset.

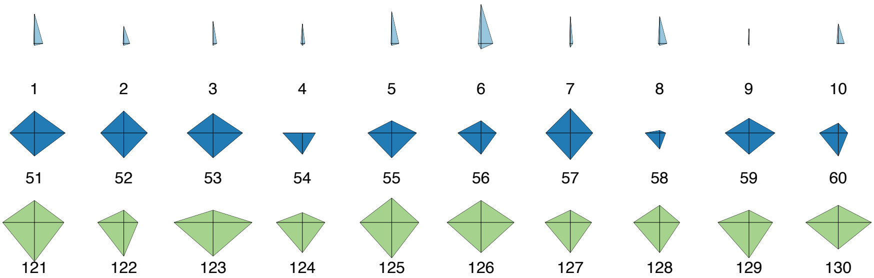


Figure 2.4: Star plots representing 30 rows of the Iris dataset. In clockwise order, the axises represent the variables Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width.

Figure 2.5 presents a *scatter-plot matrix.* It depicts a matrix of projections, where each panel represents a two-dimensional view of the database. This method excels at describing the density of the data, that is, empty zones and clusters. It can also reveal pairwise correlations. But it is limited to a few dimensions: for M variables, it requires M^2 panels. Figure 2.6 presents *parallel coordinates*, another popular approach based on projections. This methods represents each tuple by a series of connected segments. It supports more variables than scatter-plot matrices, but less tuples.

Hierarchical displays. Hierarchical displays present the data with nested sets of categories. Consider for instance the *treemap* in Figure 2.7. The data is divided in three sections: Setosa, Versicolor and Virginica. Within these sections, the map represents two categories: "Long Sepal" and "Short Sepal". Those are then divided in sub-categories, "Long Petal" and "Short Petal". Generally, hierarchical displays constitute a poor choice to describe continuous multidimensional data, but they are efficient for categorical variables, or datasets that are inherently hierarchical (e.g., file systems). In the following chapters, we will use them to describe the results of cluster analysis.

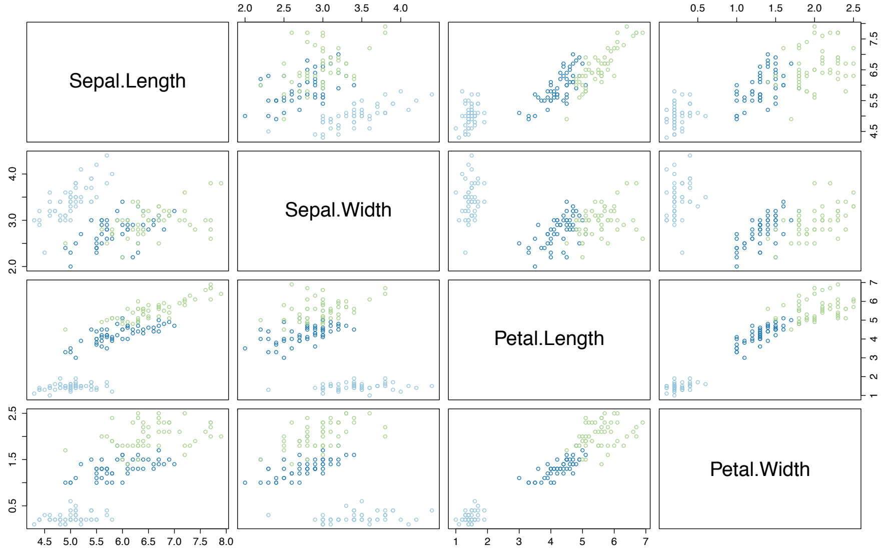


Figure 2.5: Scatter-plot matrixof the Iris dataset.

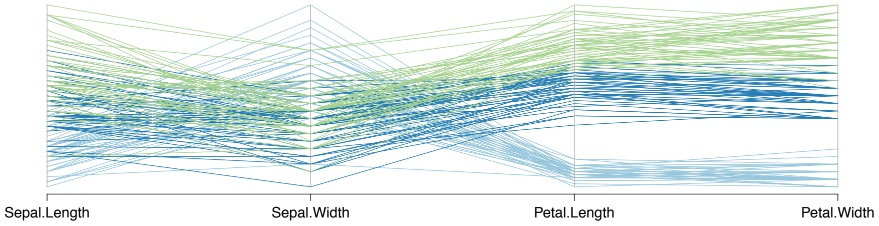


Figure 2.6: Parallel coordinates view of the Iris dataset.

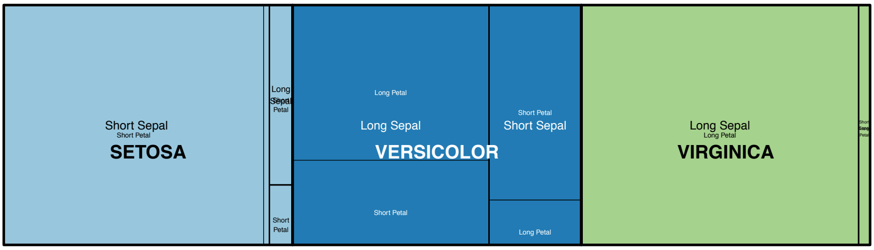


Figure 2.7: Tree Map of the Iris dataset.

Chapter 3

Background 2: Data Mining

Data cubes and visualizations involve a great deal of manual effort. Typically, analysts must start with a hypothesis ("there is something strange is my October sales"), specify a view which will confirm or infirm it ("give me all the sales of October"), then interpret the results ("aha, indeed I see an unusual deviation"). Data mining is an attempt to automate this process. Instead of writing queries, users specify *patterns* (e.g., "give me all the outliers in my sales"), or statistical models (e.g., "give me a synthetic formula to describe my sales"). Then, the system is responsible for identifying the portion of the database that fits the requirements. With data mining, users operate at a higher level of abstraction than bits and tuples. For this reason, this discipline is also referred to as *knowledge discovery*. Data mining is also very close to *machine learning*, a subfield of AI. In fact, all the techniques and algorithms discussed in this thesis directly come from this domain. Therefore, we will often use the terms *data mining* and *machine learning* interchangeably.

Because our work relies heavily on data mining, our presentation of this field will be substantially more detailed than that of OLAP cubes and visual analytics. This section is self-contained: readers who are already familiar with the concepts covered may freely skip it. In Section 1, we will present the basics of machine learning: we will introduce supervised learning, unsupervised learning, and describe a few algorithms for each task. In Section 2, we will discuss datasets with high dimensionality. We will present the "curse of dimensionality", discuss how it affects different data mining, and discuss practical solutions for both supervised and unsupervised learning.

Salary	Age	Job	Credit
80k	59	Executive	Safe
12k	22	Student	Risky
75k	24	Actor	Risky
55k	30	Architect	Safe
...	...	...	...

(a) Training set.

Salary	Age	Job
70k	59	Lawyer
32k	29	Engineer
25k	23	Student
75k	50	Marketing
...	...	...

(b) Testing set.

Figure 3.1: Dummy data sets for a classification task. The aim is to predict the credit rating of loan applicants.

1 A Data Mining Primer

This thesis focuses on the two most common tasks from data mining: *supervised learning* and *unsupervised learning.*

Supervised learning. The aim of supervised learning is to make predictions from examples. This method operates in two steps, the *learning phase* and the *prediction phase.*

1. During the learning phase, our system infers a *statistical model* from a *training set.* The training set is a bag of couples $(\mathbf{x}, t)$ where $\mathbf{x} = (x_1, \ldots, x_M)$ is a vector and t is a label. The statistical model is a function f which maps each $\mathbf{x}$ to its label t. Figure 3.1a presents a training set for a credit rating task. In this example, we wish to produce a model f to map applicants to ratings.

2. During the *prediction phase*, we consider a set of tuples for which the labels t are missing, as shown in Figure 3.1b. Our aim is to guess the labels, using the model f found previously.

If the labels t come from a continuous domain, our task is called *regression.* If they are categorical, we call it *classification.* For regression, linear models and Gaussian processes are popular methods. For classification, common algorithms include Naive Bayes, decision trees, SVMs and neural networks. We will describe the first two techniques in the following sections.

Unsupervised learning. Unsupervised learning involves no training phase. It assigns the classes t to the training items $\mathbf{x}$ directly, without any preliminary example. The most common approach for this task is *cluster analysis*, or *clustering.* Clustering consists in partitioning the data such that similar items are grouped and different items are separated. Figure 3.2 shows an example of such partitions.

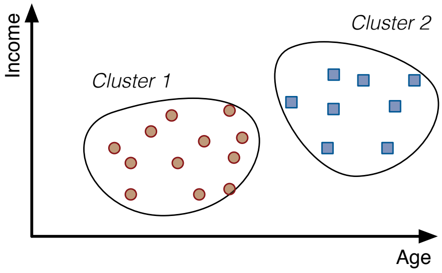


Figure 3.2: Clustering credit applicants.

All the algorithms that we will discuss in the remainder of this thesis rely on the assumption that the tuples are *independent and identically distributed* (or *i.i.d.*). This means that all the tuples come from the same probability distribution, and they were all drawn independently of each other. This assumption is disputable in many cases, but it greatly simplifies the computations. Methods to tackle it (e.g., drift detection or Markov sequences) are beyond the scope of this thesis.

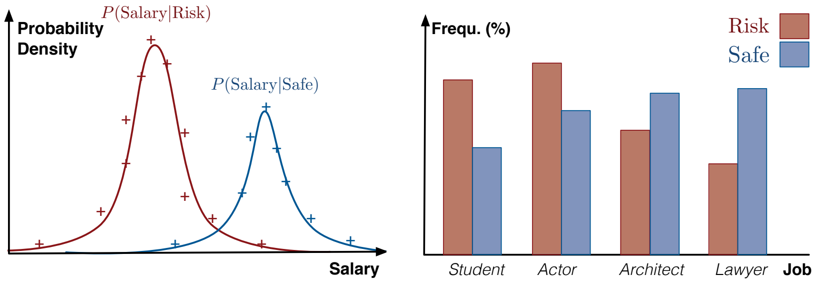
1.1 Classification

We now focus on supervised learning, and more specifically on classification. We can divide classification methods in two families: *generative* methods and *discriminative* methods. The first family specifies a full probability distribution for the couples $(\mathbf{x}, t)$, from which we can draw samples. The second predicts t directly, without describing the distribution of the data. The following sections present one algorithm for each approach: *Naive Bayes* for the generative method and *Decision Trees* for the discriminative method. We will then discuss how to tune these algorithms, and how to evaluate their accuracy.

1.1.1 Naive Bayes

Let us return to our example of Figure 3.1. For a given tuple $\mathbf{x} = (x_1, \ldots, x_M)$, our aim is to predict if the variable t is more likely to have the value “Safe” or “Risky” . One way to do so is to compute the probabilities $P(\text{Safe} \mid \mathbf{x})$ and $P(\text{Risky} \mid \mathbf{x})$, which respectively represent the likelihood that a given user $\mathbf{x}$ is safe or risky. We then compute the ratio:

$$R = \frac{P(\text{Safe} \mid \mathbf{x})}{P(\text{Risky} \mid \mathbf{x})} \tag{3.1}$$



(a) Attribute Salary (continuous).

(b) Attribute Job (discrete).

Figure 3.3: Modeling the conditional attribute densities $P(x_j \mid t)$.

If R is greater than 1, we keep the value "Safe". Otherwise, we set t to be "Risky".

Naive Bayes helps us compute the probabilities $P(t \mid \mathbf{x})$. Suppose that $P(x_j|t)$ represents the probability function of the variable x_j for the class t, for instance, the age of all safe applicants. The key observation is that under certain assumptions (which we discuss next), the following relationship holds:

$$\begin{aligned} R &= \frac{P(\text{Safe} \mid x_1, \ldots x_m)}{P(\text{Risky} \mid x_1, \ldots x_m)} \\ &= \frac{P(x_1|\text{Safe}) \cdot \ldots \cdot P(x_M \mid \text{Safe})}{P(x_1|\text{Risky}) \cdot \ldots \cdot P(x_M \mid \text{Risky})} \cdot \frac{P(\text{Safe})}{P(\text{Risky})} \\ &= \prod_{j=1}^{M} \frac{P(x_j \mid \text{Safe})}{P(x_j \mid \text{Risky})} \cdot \frac{P(\text{Safe})}{P(\text{Risky})} \end{aligned} \tag{3.2}$$

Fortunately, we can obtain all the elements of this equation from the training set. To compute the terms $P(x_j \mid t)$, we model the distribution of each variable x_j for safe and risky applicants separately. If x_j is continuous, we fit Gaussians, as in Figure 3.3a. If it is categorical, we use histograms, as in Figure 3.3b. To estimate the ratio $\frac{P(\text{Safe})}{P(\text{Risky})}$, we simply count and divide the number of observations from each class in the data.

Equation 3.2 relies on a very strong assumption: it assumes that within the tuples of each class, all the attributes are independent from each other. Formally, we have $P(x_i, x_j|t) = P(x_i|t) \cdot P(x_j|t)$. In reality, this assumption almost never holds, which is why Naive Bayes carries its name. Yet, this algorithm is surprisingly accurate in many practical situations. Its precision, combined with its speed and its ability to cope with mixed data types make it one of the most widely used classification algorithms [106].

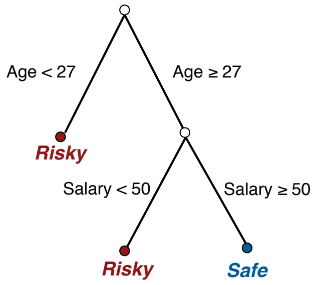


Figure 3.4: Example of decision tree for the Credit data set.

1.1.2 Decision Trees

Naive Bayes is a "black box" method: its decision are hard to interpret. We now present a popular alternative: decision trees. Decision trees are legible, which partly explains their success, but they are also fast and flexible [106].

A decision tree is a nested sequence of tests. Consider the example pictured in Figure 3.4. This structure can be interpreted as an algorithm to classify credit applicants. For any new item $\mathbf{x}$, we explore the tree from top to bottom, choosing the branches for which the condition holds. Eventually, we end up at a terminal node, which indicates the final decision.

Authors have developed several algorithms to infer decision trees from training data, such as CART, ID3 and C4.5 [106], but the general principles are similar. All these algorithms build trees in a top-down fashion. At each step, they select a leaf from the tree, select the corresponding portion of the data, and split it in two (sometimes more for C4.5). To create the split, they test each variable successively and keep the one that yields the "best" split. The quality of a split depends on how well it separates the class: we want the descendant nodes to be as *pure* as possible. In our example, a perfect split would separate safe and risky applicants, with no exceptions. To measure the impurity of a node, each algorithm uses its own function. By default, CART uses the Gini index:

$$\sum_{k \in [1,K]} p_k \cdot (1 - p_k) \tag{3.3}$$

where p_k denotes the proportion of individuals of class k. A popular alternative is the cross-entropy:

$$\sum_{k \in [1,K]} p_k \cdot \ln p_k \tag{3.4}$$

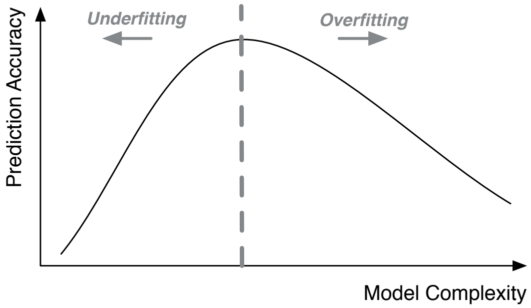


Figure 3.5: The non-monotonic relationship between mode complexity and prediction accuracy.

Both functions return 0 if all the data belongs to the same class, and they reach their maximum when the classes are present in equal proportions (e.g., in a two class case, $p_1 = 0.5$ and $p_2 = 0.5$).

Observe that deep trees are not always better. In practice, it is often necessary to *prune* our tree after we have built it, in order to obtain good prediction performance. We discuss this point in the following section.

1.1.3 Model Complexity, Overfitting and Underfitting

All supervised learnings algorithms provide ways to control the *complexity* of the model to be created. In the case of Naive Bayes, we can incorporate or ignore variables. In the case of decision trees, we can tune the height of the tree, either during the learning phase or during an optional pruning step. A model should not be too simple, otherwise it may miss important features of the training set. But it should not be too complex either. This is a consequence of the fact that training sets are limited, and therefore biased. They rarely show the full picture of the phenomenon that we are modeling. If our model fits the data too closely, it may reflect artifacts of the sample instead of properties of the real world. This situation is called *overfitting*, and it is major source of concern for data analysts. The smaller the data is, the higher are the chances that our model overfits. This effect is amplified by noisy observations (e.g., inconsistent labels), and it becomes critical in high dimensionalities, as we will later show. Figure 3.5 illustrates the balance between underfitting and overfitting.

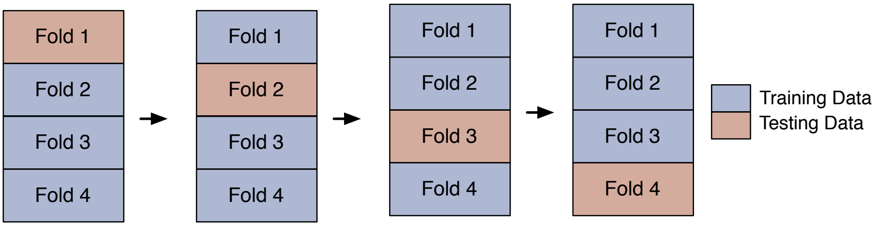


Figure 3.6: 4-fold cross validation.

1.1.4 Evaluation

To evaluate the quality of a statistical model, we need to answer two questions: which test data do we use? Then, what is our metric?

Start with the test data. Typically, we only have access to one labeled data set, which we can use for either training or testing. One option is to use it for both. But this mode of evaluation does not account for overfitting: it cannot detect whether our algorithm actually "understood" the target concepts, or if it simply learned the labels "by heart". A better option is to split the training data in two: one set for learning, and another one for validation. We can generalize this approach with *cross-validation*. Cross-validation partitions the training set in k equally large subsets. We use the first subset for validation, and the remainder for training. We then rotate: we use the second subset for validation, and the remainder for training. We repeat this operation for each of the k folds and average the scores. Figure 3.6 illustrates this method.

For a given testing set, the simplest metric to evaluate a classifier is the *misclassification rate*, that is, the proportion of correctly classified items. Nevertheless, dozens of alternatives exist. The most popular ones are the *precision*, the *recall* and the *F1 score*. Consider an arbitrary class to be predicted. The precision measures the purity of the set of items mapped to this class. The recall measures the completeness of this set. The F1 is the harmonic mean of these values. Formally, if TP, TN, FP and FN respectively represent the number of true positives, true negatives, false positives and false negatives, we have:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.6}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.7}$$

The advantage of the F1 is that it can deal with classes of different sizes. In contrast, we can "cheat" the misclassification rate by systematically predicting the most frequent class.

1.2 Cluster Analysis

Previously, we presented supervised learning. We now discuss clustering. The aim of clustering is to split data into *clusters*, that is, groups of similar objects. We can classify the algorithms along two axes: flat/hierarchical, and feature-based/dissimilarity-based [46].

- *Flat* algorithms (also called *divisive* algorithms) partition the objects into disjoint sets. In most cases, users must specify the number of partitions k a priori. Oppositely, *hierarchical* algorithms return a tree of nested partitions.

- *Feature-based* algorithms operate on the tuples as they are stored in the database. Oppositely, *dissimilarity-based* algorithms take a *dissimilarity matrix* as input. A dissimilarity matrix contains the pairwise dissimilarity between all the objects in the database. This matrix is square, and usually symmetric, as shown below:

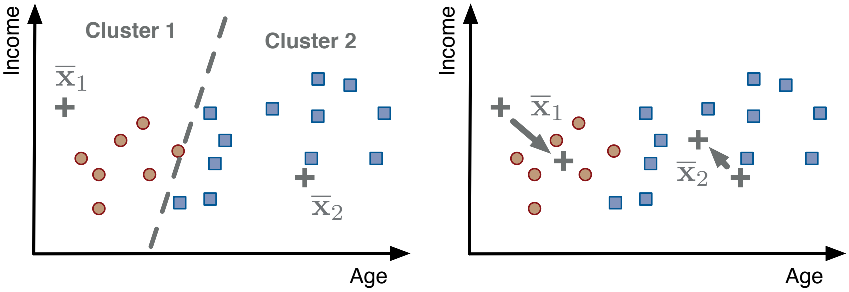
$$\mathbf{D} = \begin{bmatrix} d(\mathbf{x}_1, \mathbf{x}_1) & d(\mathbf{x}_1, \mathbf{x}_2) & \dots & d(\mathbf{x}_1, \mathbf{x}_M) \\ d(\mathbf{x}_2, \mathbf{x}_1) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ d(\mathbf{x}_N, \mathbf{x}_1) & \dots & \dots & d(\mathbf{x}_N, \mathbf{x}_M) \end{bmatrix}$$

  The advantage of dissimilarity-based clustering approach is that it can deal with non-numeric data. For instance, we can easily obtain a dissimilarity matrix from a set of strings, using e.g., the edit distance. Applying a feature-based algorithm to the same is much harder.

We will now present one algorithm for each combination, as shown in table 3.1. We will then discuss methods to detect the best number of clusters in a dataset.

	Flat	Hierarchical
Feature-based	k-means	Divisive k-means
Dissimilarity-based	PAM	Agglomerative clust.

Table 3.1: Algorithms covered.



(a) Cluster assignment step.

(b) Centroid update step.

Figure 3.7: One iteration of the k-means algorithm.

1.2.1 The k-means Algorithm

The k-means algorithm is probably the most widely used procedure in all data mining. It splits data in such a way that the distance between the points inside each partition is minimal. In other words, it seeks *tight* clusters. To evaluate the tightness of a cluster, it relies on the *within-cluster sum of squares* (WSS). The WSS aggregates the distance between the points and the centers of the clusters to which they belong (also called *centroids*). Formally, consider a set of k clusters C_i with centers $\overline{\mathbf{x}}_i$. The aim of k-means is to minimize the following quantity:

$$WSS = \sum_{C_i \in \{C_1,\ldots,C_k\}} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \overline{\mathbf{x}}_i||^2 \tag{3.8}$$

The most popular heuristic to solve this optimization problem is Lloyd's algorithm. The algorithm starts by picking k random centers. Then, each iteration is based on two steps. During the first step, it creates the clusters by assigning each point to its closest center, as shown in Figure 3.7a. During the second, it recomputes the centers using the new cluster assignments, as in Figure 3.7b. It repeats those two steps until convergence.

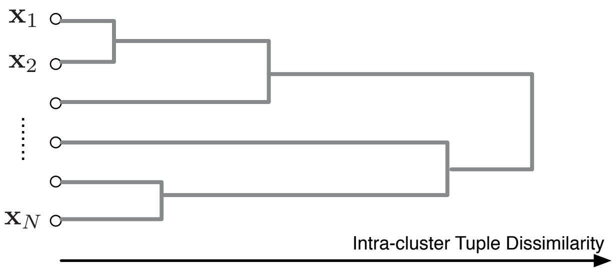


Figure 3.8: A dendrogram shows the hierarchy of the partitions.

1.2.2 Partitioning Around Medoids

Partioning Around Medoids (PAM) [46] generalizes k-means to arbitrary data types, such as strings, texts or videos. The main difference is that it works with a dissimilarity matrix rather than a table of tuples. Also, it relies on *medoids* instead of centroids. A medoid is a database object that lies at the center of a cluster (medoids come from the data, while centroids are artificial). If $\mathbf{x}_i^*$ describes the medoid of a cluster C_i, and d describes a distance function, the PAM algorithm seeks to minimize:

$$WSS = \sum_{C_i \in \{C_1,\dots,C_k\}} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{x}_i^*) \tag{3.9}$$

The body of the algorithm is itself very similar to k-means. Each iteration relies on two steps. During the first step, PAM assigns each point to its closest medoid. During the second step, it recalculates the medoids.

In its original form, PAM's time complexity is quadratic with the number of tuples in the database. In fact, the computation of the dissimilarity matrix itself runs in $\mathcal{O}(N^2)$. A more scalable alternative is CLARA [46]. At each step, CLARA takes a small sample from the database and extracts a set of medoids with PAM. It then assigns the whole database to those medoids, and computes the total score. It repeats this operation a predefined number of times, and keeps the best configuration.

1.2.3 Divisive Clustering

The k-means and PAM algorithms rely on a critical parameter: the number of clusters k to generate. In practice, we rarely have the background knowledge to set this number. Divisive clustering lets us bypass this problem. The algorithm operates as follows. First, we partition the database in two sets using k-means with $k = 2$. We then identify the largest partition, and split it in two. We obtain three partitions. We detect the largest one, and repeat the process. We stop when each partition contains one item, or when we reach some arbitrary threshold. The result of this procedure is a tree of nested partitions, which we can visualize with a *dendrogram*, as shown in Figure 3.8.

1.2.4 Agglomerative Clustering

Divisive k-means is a *top-down* algorithm: it starts with one large partition and finishes with small groups. Agglomerative clustering operates the other way around. To initialize the algorithm, we assign each object to its own cluster. Then, at each iteration, we detect the two closest clusters, and merge them. We stop when one cluster contains all the data. To define how close two clusters are, we have several possibilities. One option is to use the distance between their closest points. In this case, we have:

$$D(C_i, C_j) = \min\{d(\mathbf{x}, \mathbf{x}') : \mathbf{x} \in C_i, \mathbf{x}' \in C_j\} \tag{3.10}$$

An alternative is to use the distance between the two furthest points:

$$D(C_i, C_j) = \max\{d(\mathbf{x}, \mathbf{x}') : \mathbf{x} \in C_i, \mathbf{x}' \in C_j\} \tag{3.11}$$

The former function leads to loose partitions, while the latter leads to tight clusters. We can obtain a compromise with the mean:

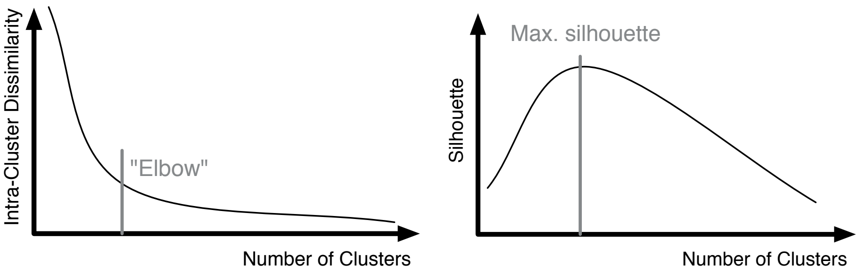
$$D(C_i, C_j) = \frac{1}{|C_1| \cdot |C_2|} \cdot \sum_{\mathbf{x} \in C_i, \mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}') \tag{3.12}$$

The algorithms corresponding to the three distance functions are respectively called *single link*, *complete link* and *average link clustering*.

1.2.5 Choosing the Number of Clusters k

Hierarchical methods provide a convenient way to avoid setting a number of clusters k explicitly. However, in many cases we need flat partitions, and therefore we need to set this k. The literature provides dozens of rules to detect the "best" number of clusters from the data, and we will now present three of those. However, we must warn the reader that these are heuristics. Eventually, the right decision depends on the data, the users and their use case.

The most simple method to detect the number of clusters in a dataset is the "elbow rule", illustrated in Figure 3.9a. The idea is to run our clustering



(a) The "elbow" method.

(b) Silhouette score maximization.

Figure 3.9: Methods to detect the number of clusters k.

algorithm for different values of k, and plot the quality of the results (e.g., the WSS) against the number of clusters. If the data is strongly clustered, a shift in the distribution will appear, which looks like an elbow. This point corresponds to the natural number of clusters in the data set.

Unfortunately, the elbow is rarely clear or visible in practice. An alternative approach is to exploit the *silhouette coefficient* [80]. As the WSS, the silhouette coefficient measures the quality of a cluster assignment. However, it is not monotonous. If we plot it against different values of k, a peak appears, as shown in Figure 3.9b. This peak corresponds to the best number of clusters. Technically, the silhouette coefficient describes how well each object "fits" inside its cluster. Consider an object $\mathbf{x}_i$. If $a(\mathbf{x}_i)$ represents the average dissimilarity between $\mathbf{x}_i$ and the other tuples from its cluster, and $b(\mathbf{x}_i)$ is the lowest average dissimilarity between $\mathbf{x}_i$ and the points of another cluster, we have:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}} \tag{3.13}$$

If $\mathbf{x}_i$ is at the center of its cluster, then we will have $s(\mathbf{x}_i) = 1$. If it lies on the border of the cluster, we obtain $s(\mathbf{x}_i) = 0$. Finally, if $\mathbf{x}_i$ is closer to another cluster than its own, the score is negative (-1 in the worst case). To obtain a global score, we average the silhouette scores of each object the database.

Finally, a more robust alternative is the Gap statistic [93]. This index uses two datasets: the original database, and a synthetic set which serves as a baseline. To generate the baseline, we sample tuples uniformly from the bounding box of the original dataset. Thus, we obtain a database which values have the same domain as that of the real data, but which contains no cluster. For a given k the value of the Gap statistic is the difference

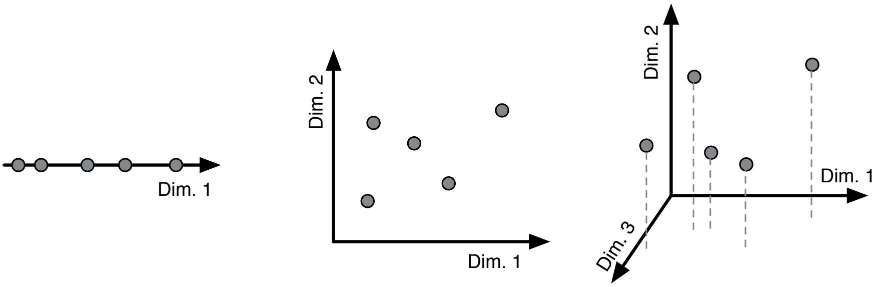


Figure 3.10: Five data points in 1, 2 and 3 dimensions. Observe how the data gets sparser as the number of dimensions increase.

in clustering quality between those two sets. Formally, if $WSS(k)$ is the score obtained with the real data and $E_S\left[WSS(k)\right]$ is the expected score of clustering S uniformly distributed samples, we have:

$$Gap_S(k) = E_S\left[\log WSS(k)\right] - \log WSS(k) \tag{3.14}$$

The notation $E_S\left[WSS(k)\right]$ expresses the fact that we run the experiment with several samples and then average results, i.e., we perform Monte-Carlo simulation.

2 Mining High Dimension Datasets

Previously, we presented the basics of classification and clustering. We now discuss the problem which arise with these methods when datasets contain many variables.

The *curse of dimensionality* tells us that the more variables a database contains, the *sparser* it gets. Consider for instance the 5 points pictured in Figure 3.10. As we increase the number of dimensions in our dataset, the distance between the points grows, and the space in which they live get emptier. And this effect increases exponentially with the number of dimensions. This observation has important consequences for machine learning. In the following section we will present those that concern supervised learning. In the next, we will discuss unsupervised learning.

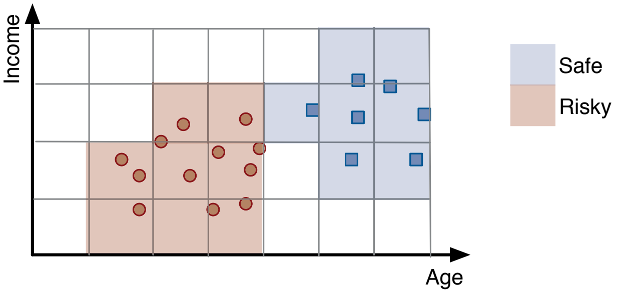


Figure 3.11: A grid-based classifier.

2.1 Supervised Learning with Many Variables

2.1.1 The Explosive Need For Data

We present the effect of dimensionality on regression and classification through an example, largely inspired by Bishop [14]. Consider the classification task introduced in Figure 3.1. We want to predict the credit rating of a set of applicants. We already presented two methods to do so, namely Naive Bayes and decision trees. We now introduce a third one: *grid-based classification*. The idea behind this method is to divide the data space into cells, as shown in Fig. 3.12. Then, we associate each cell to the most frequent credit rating, either Risky of Safe. When a test tuple comes in, we identify the cell to which it belongs, and retrieve the corresponding value of the target.

To produce an accurate classifier, we need as many training points as possible. Ideally, we should have at least one per cell. Suppose that we bin each dimension into S cells. If our data contains one dimension, then we need S training points. If contains two dimension, we need S^2 examples. More generally, if our dataset contains M dimensions, we need S^M examples. Therefore, the number of training examples we need grows exponentially with the number of dimensions of the data. This is a typical manifestation of the curse of dimensionality: we end up like the mythological king who had to place ever more wheat on a chessboard to catch up with a geometric progression. And this problem does not only plague our grid-based classifier: without proper tuning, all classification algorithms suffer from this effect.

2.1.2 Feature Selection

To deal with the curse of dimensionality, data analysts have two options: either they provide more training data, or they reduce the number of columns in their database. Let us discuss the second option. The most common method to reduce the dimensionality of a dataset is *feature selection.* Feature selection aims at identifying the variables which are useful for a given inference task, and eliminate all the others. The hope is that the resulting set will be small enough to be processed efficiently. We cannot overstate how important this step is in practice. Following the classification of Guyon and Elisseff [36], we present three types of feature selection methods: *wrappers*, *filters* and *embedded methods.*

Wrappers. Consider a training set and a supervised learning algorithm. Wrappers use the algorithm as a "black box" to check the predictive power of different subsets of variables. They start by building a classifier for each dimension separately. They check which one leads to the best results, and attempt combinations with two variables. They keep the best candidate, and reiterate with higher number of columns. They repeat the process until the performance starts to decrease. Wrappers can also go the opposite direction: they start with all the variables in the database, and suppress them one by one. This method is simple, flexible and accurate. However, it is often slow, because it builds a classifier for each combination of variables to be tested.

Filters. Filters separate the variable selection from the actual classification. In a first step, they identify potentially interesting variables. In the second step, they run the actual learning algorithm. To detect interesting variables, they usually check if the dimensions are statistically dependent to the variable to be predicted, using for instance the correlation coefficient. They then rank the variables and keep those that satisfy an arbitrary threshold. We will describe these approaches in detail and generalize them in Chapter 5.

Embedded methods. Embedded methods combine classification and feature selection in one procedure. In fact we have already reviewed the most popular of those: decision trees, which have a built-in mechanisms to filter variables. An alternative is to chose an existing full-space algorithm, such as linear regression or SVM, and force them to "mute" some variables by modifying their objective function. This process is known as *regularization.*

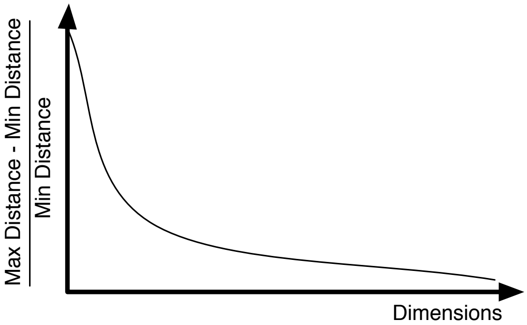


Figure 3.12: Effect of the curse of dimensionality on distances. The Y axis represents the normalized difference between the largest and the smallest pairwise distance between the objects in the data.

2.2 The Curse of Dimensionality in Clustering

We now discuss the effects of high dimensionality on unsupervised learning.

2.2.1 The Leveling of Distances

We have seen that when we introduce new variables in a database, the pairwise distance between its objects grows (cf. Figure 3.10). If the new variables are strongly correlated to the original ones, then this effect is relatively harmless: the pairwise distances are scaled, but their relative proportions remain identical. However, if the new variables are independent from the old ones, then the distances are distorted: objects that were close will become distant, and objects that were far apart will become close. Therefore, increasing the dimension of the data has a *leveling* effect on the distances between the data points. Eventually, all the objects become equidistant. Figure 3.12 illustrates this effect. Formally, if M represents the number of independent variables in the data, and if $MAX_d(M)$ and $MIN_d(M)$ respectively represent the largest and smallest pairwise distance between the objects in the database, we have [13]:

$$\lim_{M \to +\infty} \frac{MAX_d(M) - MIN_d(M)}{MIN_d(M)} = 0 \tag{3.15}$$

Yet, recall that the aim of cluster analysis is to group similar objects and separate different ones. In high dimensionality, this task loses all its meaning: the objects become equidistant, and therefore the clusters disappear. A partitioning based on clustering is as good as random.

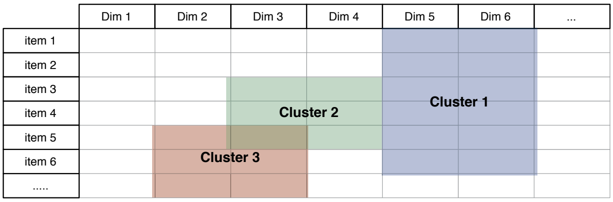


Figure 3.13: Subspace clustering.

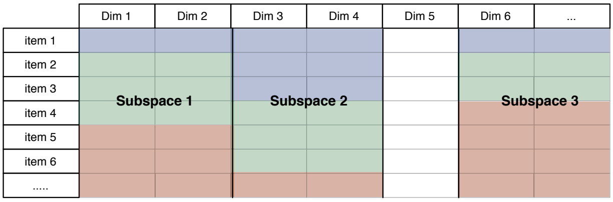


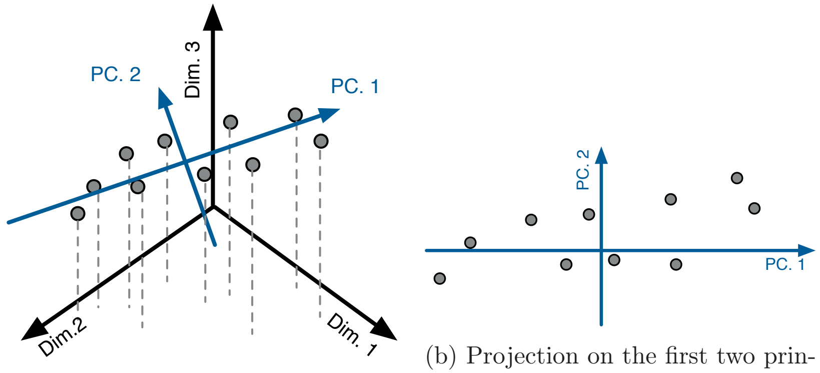
Figure 3.14: Multiview clustering.

2.2.2 Subspace Clustering, Multiview Clustering

This last decade, several researchers have generalized cluster analysis to high dimensionality spaces. The main idea is to seek clusters into *subspaces* of the data. Instead of returning simple partitions, the algorithms seek couples (S_i, C_i) where S_i is a subspace and C_i is a cluster. Thus, they return both tuples and the sets of dimensions on which they are clustered. *Subspace clustering* returns one subspace per cluster, as shown in Figure 3.13. Alternatively, *multiview clustering* decouples subspace search and the cluster analysis, as shown in Figure 3.14. We will discuss these methods in more detail in Chapter 6, and introduce our own.

2.3 Principal Component Analysis

To finish, observe that we can distinguish the data's *physical dimensionality* from its *intrinsic dimensionality*. The first property refers the number of columns in the database. The second one describes the number of independent variables necessary to represent the dataset without loss of information [17]. Consider a database with M columns. The physical dimensionality of this set is M. If all the columns are completely independent, then its intrinsic dimensionality is also M. Oppositely, if all columns contain exactly the same data, then the intrinsic dimensionality falls to 1. Indeed, we need just one variable to represent the whole dataset. Many



(a) Original data space.

(b) Projection on the first two principal components.

Figure 3.15: Principal Component Analysis.

real-life datasets lie between these two extremes: the columns present some statistical dependency, but these are not full correlations. Accordingly, their intrinsic dimensionality varies between 1 and M.

The *Principal Component Analysis* (PCA) algorithm lets us transform a wide dataset into a dense, compressed version. The physical dimensionality of its output is smaller than that the original data, and hence it is easier to process. However, the intrinsic dimensionality is preserved. Thus, PCA is a common preprocessing step for cluster analysis, classification, or for visualization. Technically, PCA seeks to project the data onto low dimension hyperplanes, in such a way that the subsequent loss of information is minimized. Figure 3.15 illustrates this idea: PCA projects a 3D cloud of points onto a 2D plane, but it manages to preserve much of the point's distribution. The axes on which the data is projected are called *principal components*. We obtain them by analyzing the spectrum of the database's covariance matrix [14].

Chapter 4

Background 3: Information Theory

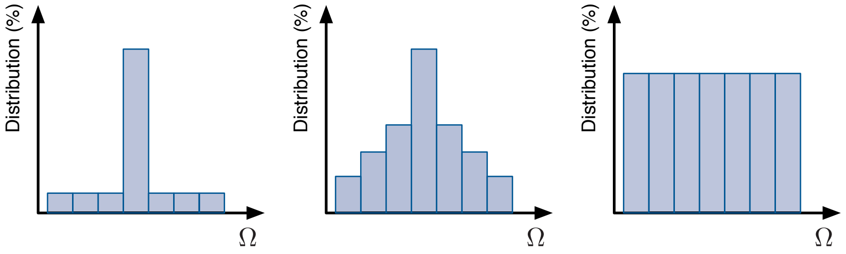
We now turn to *information theory*, a branch of applied mathematics concerned with the compression and transmission of data. Information theory provides us with a powerful framework to understand how variables relate to each other, and we will heavily rely on it in the coming chapters.

1 Introducing the Entropy

The *entropy* $H(\mathcal{X})$ of a variable $\mathcal{X}$ describes its variability, that is, how "unpredictable" it is [23]. If $\mathcal{X}$ is a constant, then $H(\mathcal{X}) = 0$. In contrast, if $\mathcal{X}$ is highly unpredictable (e.g., $\mathcal{X}$ is the outcome of flipping a perfectly balanced coin) then $H(\mathcal{X})$ is maximal. Formally, if $\mathcal{X}$ is a discrete variable with sample space Ω, then we have:

$$H(\mathcal{X}) = -\sum_{x \in \Omega} P(\mathcal{X} = x) \cdot \log P(\mathcal{X} = x) \tag{4.1}$$

Figure 4.1 gives several examples of distributions along with their respective entropies.



(a) Low entropy: $H(\mathcal{X})$=1.65.

(b) Medium entropy: $H(\mathcal{X})$=2.44.

(c) High entropy: $H(\mathcal{X})$=2.80.

Figure 4.1: Histograms and entropy for three discrete distributions.

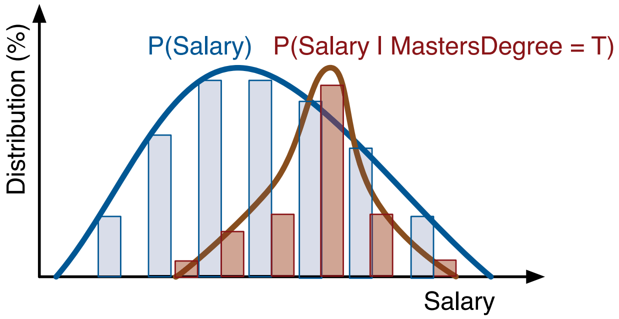


Figure 4.2: Distributions $P(\text{Salary})$ and $P(\text{Salary} \mid \text{Masters.Degree} = \text{T})$.

It is often said that the entropy contains the "quantity of information" contained in a message. To understand why, let us briefly turn to compression theory. Consider a file in which each character $\mathcal{X}$ is drawn randomly from the distribution $p(\mathcal{X})$. Furthermore, suppose that we wish to compress this file, without loss of information. According to Shannon's *source coding theorem* [23], the average number of bits necessary to store each symbol cannot fall below the entropy. In other words, the entropy is a theoretical lower bound on the average code length in a lossless compression scheme. Accordingly, we measure it in *bits*.

2 The Mutual Information

Apart from its compression-related properties, the entropy gives us a mean to quantify how variables interact. Indeed, if two variables are *statistically dependent*, then conditioning one variable (that is, restricting the range of its values) will affect the entropy of the second. Suppose that we wish to test the relationship between two variables from a census, Masters.Degree and Education. We can do so as follows. First, we compute the distribution of Salary and compute its entropy $H(\text{Salary})$. Second, we compute the distribution of Salary for all the individuals with a Master's, and compute the new entropy $H(\text{Salary} \mid \text{Masters.Degree} = \text{T})$. Figure 4.2 illustrates both distributions. Finally, we compute the difference between the two entropies:

$$\Delta H = H(\text{Salary}) - H(\text{Salary} \mid \text{Masters.Degree} = \text{T}) \tag{4.2}$$

If ΔH is positive, then the variables Masters.Degree and Salary are dependent: knowing the value of one reduces the uncertainty of the second.

Oppositely, if ΔH is null, these variables are independent. Therefore, ΔH quantifies the degree of relationship between two variables.

The *mututal information* is a generalization of ΔH. If $\mathcal{X}$ and $\mathcal{Y}$ are two random variables, the expression $H(Y|\mathcal{X} = k)$ describes the entropy of Y given $\mathcal{X} = k$. If we average this expression over all possible values of k, we obtain the *conditional entropy*:

$$H(\mathcal{Y}|\mathcal{X}) = \mathbb{E}_x[H(\mathcal{Y}|\mathcal{X} = x)] \tag{4.3}$$

We define the mutual information I as follows:

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) \tag{4.4}$$

The mutual information describes the loss in entropy between Y and $Y|\mathcal{X}$, and therefore it quantifies the dependence between $\mathcal{X}$ and Y. It is symmetric, and it is always positive or null.

The mutual information is not the only method to quantify the dependence between variables. The statistics literature contains several other candidates, including the correlation coefficient. The advantage of the mutual information is its generality: it can cope with both categorical and continuous data (after adjustments, cf. next section). It supports univariate and multivariate distributions indifferently. More importantly, it can detect *non-linear* relationships between variables.

A few authors have presented extensions to the mutual information. A notable variant is the *variation of information* [63]:

$$VI(\mathcal{X},\mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}) - 2 \cdot I(\mathcal{X};\mathcal{Y}) \tag{4.5}$$

As opposed to the regular mutual information, the VI is a true metric. Hence, it obeys the triangle inequality $VI(\mathcal{X},\mathcal{Y}) + VI(\mathcal{Y},\mathcal{Z}) \leq VI(\mathcal{X},\mathcal{Z})$. This property will come in handy when we will apply cluster analysis to variables in Chapter 6.

3 Chain Rule and Conditional Mutual Information

The entropy and the mutual information can handle more than one variable. The expression $H(\mathcal{X},\mathcal{X}_2)$ describes the entropy of the *random vector* $(\mathcal{X}_1,\mathcal{X}_2)$. Also the expression $I(\mathcal{X}_1,\mathcal{X}_2;\mathcal{Y})$ describes the dependency between this vector and the variable $\mathcal{Y}$ (observe the difference between the comma and the semicolon).

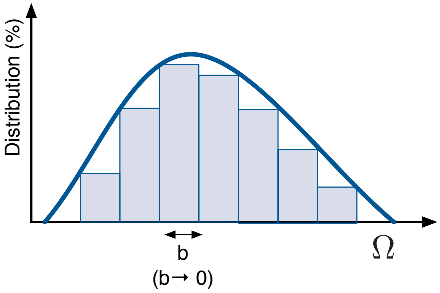


Figure 4.3: Discretized probability function.

To compute the mutual information $I(\mathcal{X}_1, \mathcal{X}_2; \mathcal{Y})$, we can generalize Equation 4.4 as follows:

$$I(\mathcal{X}_1, \mathcal{X}_2; \mathcal{Y}) = H(\mathcal{X}_1, \mathcal{X}_2) - H(\mathcal{X}_1, \mathcal{X}_2|\mathcal{Y}) \tag{4.6}$$

Alternatively, we can use the *mutual information chain rule*, which is better suited for iterative processing:

$$I(\mathcal{X}_1, \ldots, \mathcal{X}_D; \mathcal{Y}) = I(\mathcal{X}_1; \mathcal{Y}) + I(\mathcal{X}_2; \mathcal{Y}|\mathcal{X}_1) + \ldots + I(\mathcal{X}_M; \mathcal{Y}|\mathcal{X}_1, \ldots \mathcal{X}_{M-1}) \tag{4.7}$$

$$= \sum_{m \in [1,M]} I(\mathcal{X}_m; \mathcal{T}|\mathcal{X}_1, \ldots, \mathcal{X}_{m-1}) \tag{4.8}$$

In those equations, the notation $I(\mathcal{X}_j; \mathcal{T}|\mathcal{X}_i)$ expresses the *conditional mutual information*. The conditional mutual information is a conditioned version of the mutual information: it describes the dependency between $\mathcal{X}_j$ and $\mathcal{T}$ *given restrictions on* $\mathcal{X}_i$. To obtain it, we compute the mutual information between $\mathcal{X}_j$ and $\mathcal{T}$ given all the possible values of $\mathcal{X}_i$, and average the results. Formally:

$$I(\mathcal{X}_j; \mathcal{T}|\mathcal{X}_i) = \mathbb{E}_{x_i}\left[I(\mathcal{X}_j; \mathcal{T})|\mathcal{X}_i = x_i)\right] \tag{4.9}$$

The influence of $\mathcal{X}_i$ can go either way: it can weaken the dependency between $\mathcal{X}_j$ and $\mathcal{T}$, or it can strengthen it. The conditional mutual information is positive or null, and it is bounded by the entropy of $\mathcal{X}_j$ and T.

4 Continuous Entropy

The entropy as defined in Equation 4.1 only supports discrete data. The *differential* entropy generalizes it to continuous domains. If $p(\mathcal{X})$ is a continuous probability distribution, we define it as follows:

$$H(\mathcal{X}) = -\int_{\Omega} p(x) \cdot \log p(x) \mathrm{d}x \tag{4.10}$$

We replaced the sum in Equation 4.1 by an integral, but we preserved the functional form. We can substitute this version of the entropy in equations 4.3 and 4.4: we obtain the continuous form of the conditional entropy and the mutual information.

The continuous and discrete versions of the entropy are intimately linked. Suppose that we discretize the continuous variable $\mathcal{X}$ into a new variable $\mathcal{X}^b$, where b is the bin size as shown in Figure 4.3. Assuming that $p(\mathcal{X})$ is Riemann integrable, the following result holds:

$$H(\mathcal{X}) = \lim_{b \to 0} H(\mathcal{X}^b) + \log b \tag{4.11}$$

Observe that the term $\log b$ tends to $-\infty$. It compensates for the fact that $H(\mathcal{X}^b)$ diverges when b gets small. We refer the interested reader to Thomas and Cover [23] for more details.

5 Estimation

In practice, we almost never know the exact distribution of the random variable $\mathcal{X}$ that interests us. We only have access to samples. Hence, we must use *estimators*. If the variable is discrete, we can use simple histogram-based solutions. We set $\hat{P}(X) \approx P(\mathcal{X} = x)$ to be the proportion of tuples with $X = x$, and we plug this estimator in our information theoretic measurements. Dealing with continuous variables is more complex, as we need more expensive density estimators [19]. A pragmatic (but lossy) solution is to discretize those variables. We will use this method by default.

6 Summary

Let us summarize the main points in this section.

- The entropy $H(\mathcal{X})$ of a variable $\mathcal{X}$ describes its uncertainty. We generalize this notion to vectors with the notation $H(\mathcal{X}_1, \ldots \mathcal{X}_m)$.
- The mutual information $I(\mathcal{X}; \mathcal{Y})$ measures the statistical dependency between $\mathcal{X}$ and $\mathcal{Y}$. We generalize this notion to vectors with the notation $I(\mathcal{X}_1, \ldots \mathcal{X}_m; \mathcal{Y}_1, \ldots \mathcal{Y}_n)$. The mutual information is sensitive to both linear and non-linear dependencies.

- The mutual information is not trivially additive: in the vast majority of cases, we have $I(\mathcal{X}_1, \mathcal{X}_2; \mathcal{Y}) \neq I(\mathcal{X}_1; \mathcal{Y}) + I(\mathcal{X}_2; \mathcal{Y})$. The correct value depends on complex three-variable interactions, described by the conditional mutual information.

- In its original form, the entropy is undefined for continuous variables. A pragmatic (though lossy) solution is to bin the data and treat it as categorical. A more advanced (though typically expensive) approach is to estimate the differential entropy, a generalization of the entropy to continuous domains [11].

Chapter 5

Claude: a Data Warehouse Explanation System

1 Introduction

In Chapter 2, we presented data warehouses and described two methods to explore them, data cubes and visualizations. Usually, those tools are fast and intuitive. They excel at providing quick answers and clear displays. But they rely entirely on manual effort. Eventually, they are merely a layer on top of SQL. They rely on the query-result paradigm, and they can involve long cycles of trial and error. In Chapter 3, we discussed data mining. These techniques bring automation to data exploration. Yet, they require much more background knowledge. At the time of writing, experts in data mining (so-called "data scientists") were still a rare and expensive resource. Can we find a middle way? Can we design a method to analyze data warehouses that would be both automatic and accessible?

1.1 Contributions

We now introduce Claude, a semi-automatic system to explain the content of a data warehouse. Claude answers the question: "What is in my database?" To do so, it analyzes the statistical structure of the data and infers potentially informative queries. Three features make Claude unique compared to other machine learning algorithms. First, it returns SQL statements instead of abstract statistical models. Thus, users can directly visualize its findings with existing database front ends. Second, it explains its choices: it can illustrate its findings with examples. Finally, Claude enforces results diversity. Instead of seeking one optimal query, it returns several different possibilities to be validated by the user. Thus, our system combines the automation of data mining with the transparency and flexibility of database queries.

In this chapter, we will first present a mathematical model of what makes a database view informative. The main idea is to exploit the linear and non-linear statistical dependencies between the columns of the database. We will formalize this idea with information theory; more specifically we will exploit the mutual information and the Kullback-Leibler divergence. Then, we will present several algorithms to generate views from this model. We will show that naive solutions are too slow to be useful in practice. Therefore, we will present aggressive heuristics based on approximations and greedy search. In the last sections of this chapter, we will present our experiments with real-life datasets. First we will describe a use case, for which will we discuss our system's findings. Then, we will report on systematic experiments, during which we simulate users with statistical classifiers. Our results will reveal that Claude can effectively capture the statistical structure of the database, and that it is faster than existing solutions.

1.2 Outline

The rest of this chapter is organized as follows. Section 2 gives an overview of our model, which we refine in Section 3. Sections 4 describes how to compute the quality of a view, Section 5 presents our view detection algorithm, and Section 6 describes how to justify our results. We present our experiments in Section 7, related work in Section 8 and conclude in Section 9.

2 General Model

2.1 Objective

Let us present our model through an example. We want to understand which US cities are prone to crime. We have a database that provides several dozen socio-economic indicators for thousands of US cities (for instance, employment, age, or diplomas), as well as the number of crimes for each city. Following the OLAP terminology, we refer to the first group of columns as the *dimensions*, and the crime index as the *measure* (we presented these notions in Chapter 2). We are oblivious to the physical structure of the data, we assume that our database is stored in one large table. Our aim is to understand the "big picture": which variables correlate with unusually high or low levels of crime? Which cities are impacted? We want to describe how the measure varies across the dimensions.

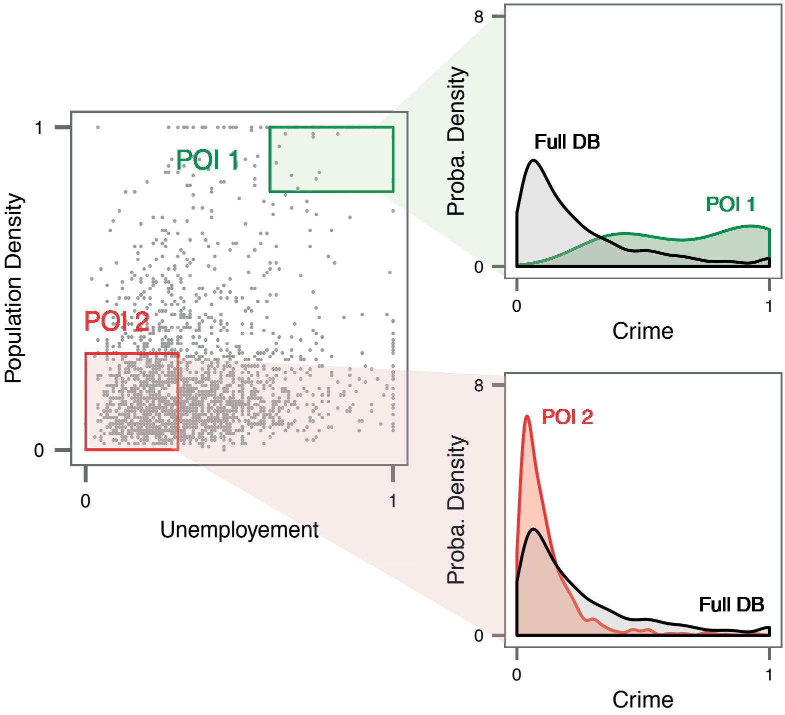


Figure 5.1: Example view, with two points of interest. All values are normalized.

To answer these questions, our system generates *views* and *points of interest* (POIs).

- A view is an "informative" selection of dimensions.
- A POI is a region in the view where the measure behaves "unusually".

We will formalize the terms "informative" and "unusual" in the following section. Let us first present them with an example. Figure 5.1 presents one view and two POIs. The view involves the dimensions `Unemployment` and `Population Density`. Why did Claude pick these two columns? The POIs provide explanations. Cities with high densities and high unemployment rates have more crimes (cf. POI_1). In contrast, cities with low densities and lower employment rates tend to be safer (cf. POI_2). We see that the columns of the view influence the measure. The POIs illustrate how this influence manifests itself.

2.2 Formalization

We introduced views and POIs. Let us now discuss how to measure their quality.

2.2.1 View Quality

In our example, we consider that the columns `Unemployment` and `Population Density` are informative because their combination influences the target `Crime`. One way to measure this influence is to use statistical dependency. If two columns are independent, then there is little chance that they are related in the real world. Oppositely, a correlation indicates a possible relationship. Therefore, a view in which the dimensions are strongly dependent to the measure contains potentially interesting information. Our whole model relies on this observation: we assume that a view is interesting if its dimensions are jointly dependent to the measure. To formalize this property, we introduce the *view strength*:

Definition 5.1. *Consider a view $V = \{X_1, \ldots, X_D\}$, the target T, and a measure of statistical dependence $\mathfrak{S}$. We define the* strength *of the view V as follows:*

$$\sigma(V) = \mathfrak{S}(\mathcal{X}_1, \ldots, \mathcal{X}_D; \mathcal{T}) \tag{5.1}$$

The notation X refers to database columns, while $\mathcal{X}$ refers to their underlying random variables. The variables X_m and T respectively represent a dimension and the measure. We will instantiate the function $\mathfrak{S}$ later, in section 3.

2.2.2 POI quality

Observe the two POIs in Figure 5.1. The measure behaves “unusually”: its distribution in these regions differs from that in the rest of the database. It is skewed to the right in POI_1, it is skewed to the left in POI_2. We name this property *POI divergence.* Consider a view based on the random variables $\mathcal{X}_1, \ldots, \mathcal{X}_D$, with respective sample spaces $\Omega_1, \ldots, \Omega_D$. The set $R \subset \Omega_1 \times \ldots \times \Omega_D$ represents a region in this view. The random variable $\mathcal{T}$ represents the target for the whole database. The random variable $\big[\mathcal{T} | (\mathcal{X}_1, \ldots, \mathcal{X}_D) \in R\big]$ represents the target for the tuples within R. We shorten this notation to $\mathcal{T}|R$. In our model, the region R is a good point of interest if $\mathcal{T}|R$ and $\mathcal{T}$ have large differences in distribution.

Definition 5.2. *Let $R \subset \Omega_1 \times \ldots \times \Omega_D$ represent a region in the view $\{X_1, \ldots, X_D\}$, and let $\mathcal{T}$ represent our target variable. The function $\mathfrak{D}$ measures the dissimilarity between two probability distributions. We define R's* divergence *as follows:*

$$\delta(R) = \mathfrak{D}\big(\mathcal{T}|R; \mathcal{T}\big) \tag{5.2}$$

Here again we will instantiate the function $\mathfrak{D}$ in Section 3.

2.2.3 Problem Statement

We presented views and POIs, and we defined how to measure their quality. We can now formulate the general problem statement behind Claude:

Problem 5.1. *Consider a dataset DB, a target column T, a measure of statistical dependence $\mathfrak{S}$ and a measure of distribution dissimilarity $\mathfrak{D}$.*

- *Find the top K strongest views with at most D columns.*
- *For each of these views, find the top P divergent POIs.*

We call the first sub-problem column search *and the second* POI detection*.*

In the rest of this chapter, we will illustrate Claude's views with mathematical notation or visualizations. In practice however, Claude expresses its recommendations with SQL queries. It describes the views with simple Select-Project statements:

```
SELECT X1, ... , Xd, T
FROM DB;
```

It returns each point of interest as follows:

```
SELECT X1, ... , Xd, T
FROM VIEW
WHERE X1 BETWEEN [L1, H1]
  AND ...
  AND Xd BETWEEN [Ld, Hd]
```

In these queries, `X1`, ..., `Xd` represent the variables of the view, the intervals `[L1, H1]` ... `[Ln, Hn]` represent the bounds of the POI, and `T` represents the target.

3 Model Instantiation

We presented views and POIs without specifying any measure of dependence $\mathfrak{S}$ or dissimilarity $\mathfrak{D}$. We now instantiate these quantities with fundamental information theory.

3.1 View Strength

According to our model, a set of dimensions is interesting if its columns are jointly dependent to the target. To quantify this dependence, we use the mutual information, presented in Chapter 4. Recall that this measure presents many advantages: it is sensible to non-linear effects, it can cope with any kind of variables, and it is practical to compute. Accordingly, we set:

$$\sigma(V) = I(\mathcal{X}_1, \ldots, \mathcal{X}_D; \mathcal{T}) \tag{5.3}$$

We consider, that a view is strong if the mutual information between the dimensions and the measure is high.

Recall from Chapter 4 that the mutual information deals natively with categorical data. Extensions for continuous data exist, but they are more computationally involved. In our implementation, we opted for a pragmatic solution: we bin the continuous variables and treat them as categorical.

3.2 Points of Interest and Divergence

Let us now refine our definition of divergence. We established that the divergence of a POI is the dissimilarity between the target's distribution within this POI, and the target's distribution in the whole database. To measure this dissimilarity, we use the Kullback-Leibler divergence (KL). The KL divergence measures the difference between two probability distributions. It is null if the two distributions are similar, and it grows when the distributions differ. Formally, if $\mathcal{X}$ and $\mathcal{Y}$ are two discrete random variables with the same sample space Ω, we have:

$$KL\big(\mathcal{X} \parallel \mathcal{Y}\big) = \sum_{x \in \Omega} P(\mathcal{X} = x) \cdot \log \frac{P(\mathcal{X} = x)}{P(\mathcal{Y} = x)} \tag{5.4}$$

In the continuous case, we have:

$$KL\big(\mathcal{X} \parallel \mathcal{Y}\big) = \int_{\Omega} P(\mathcal{X} = x) \cdot \log \frac{P(\mathcal{X} = x)}{P(\mathcal{Y} = x)} dx \tag{5.5}$$

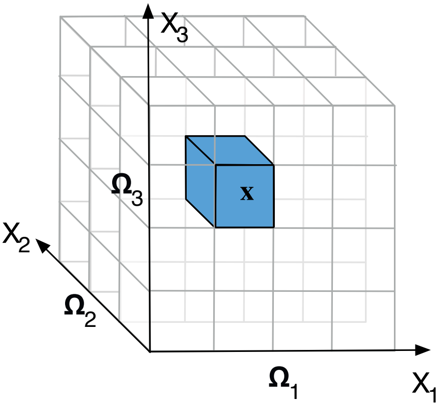


Figure 5.2: Example of a view with three discrete variables $V = \{X_1, X_2, X_3\}$. The average divergence of the cells $\delta(\mathbf{x})$ equals the strength of the view $\sigma(V)$.

Here is our final version of the POI divergence:

$$\delta(R) = KL\big(\mathcal{T}|R \parallel \mathcal{T}\big) \tag{5.6}$$

Our region R is a good point of interest if $KL(\mathcal{T}|R \parallel \mathcal{T})$ is as large as possible. Here again, we can speed up the computations by discretizing the continuous variables, to approximate the integral in Equation 5.5.

3.3 The Relationship Between Strength and Divergence

We now justify the choice of the Kullback-Leibler to instantiate the POI divergence. The main idea is that the KL divergence and the mutual information are “two sides of the same coin”: we obtain the latter by averaging the former over all possible POIs. Therefore, our instantiation of strength and divergence are tightly related.

Consider a view V, made of discrete variables. If we compute the divergence of each distinct tuple and average the results, we obtain V’s strength. We illustrate this property with Figure 5.2. We generalize this observation as follows:

Lemma 5.1. *If V is a view with d variables and $\mathbf{X} \in \Omega_1 \times \ldots \Omega_D$ is a tuple from this view, then:*

$$\sigma(V) = E_{\mathbf{X}}\big[\delta(\{\mathbf{X}\})\big] \tag{5.7}$$

Proof. Consider the random vector $\mathbf{X} = [\mathcal{X}^1, \dots, \mathcal{X}^M]^\top$ and the measure $\mathcal{T}$. From Cover and Thomas, Equation 2.35 [23] and Bayes' theorem, we make the following derivation:

$$\begin{aligned}
I(\mathbf{X}, \mathcal{T}) &= \int_{\mathbf{X}} \int_{\mathcal{T}} P(\mathbf{X}, \mathcal{T}) \log \frac{P(\mathbf{X}, \mathcal{T})}{P(\mathbf{X})P(\mathcal{T})} d\mathbf{X} d\mathcal{T} && (5.8)\\
&= \int_{\mathbf{X}} P(\mathbf{X}) \int_{\mathcal{T}} P(\mathcal{T} \mid \mathbf{X}) \log \frac{P(\mathcal{T}|\mathbf{X})}{P(\mathcal{T})} d\mathcal{T} d\mathbf{X} && (5.9)\\
&= \int_{\mathbf{X}} P(\mathbf{X}) \cdot KL\big(\mathcal{T}|\mathbf{X} \parallel \mathcal{T}\big) d\mathbf{X} && (5.10)\\
&= E_{\mathbf{X}}\big[KL\big(\mathcal{T}|\mathbf{X} \parallel \mathcal{T}\big)\big] && (5.11)
\end{aligned}$$

Substituting the left side with Equation 5.3, and the right side with Equation 5.6, we obtain the lemma. We skip the discrete case, which uses the exact same development with sums instead of integrals. □

Let us now study the effects of discretization, as discussed in Section 3.1 and 3.2. Suppose that we obtained a view V by binning a set of continuous variables V^*. The average divergence of V's bins equals the strength of V, but not that of V^*. Fortunately, these quantities converge as the bins get small.

Lemma 5.2. *The view V is a set of continuous variables, V^b is a discretized version of V in which each variable is binned with bin size b, and $\mathbf{X}^b$ is a tuple from V^b. We have $\mathbb{E}_{\mathbf{X^b}}\big[\delta(\{\mathbf{X^b}\})\big] \to \sigma(V)$ as $b \to 0$.*

Proof. Let the D-dimensional random vector $\mathbf{X}$ describe the (continuous) variables of V, and $\mathbf{X}^b$ describe the (discrete) variables of V^b. By generalizing Equation 4.11 to the multivariate case, we obtain:

$$\lim_{b \to 0} H(\mathbf{X}^b) + D \cdot \log b = H(\mathbf{X}) \qquad (5.12)$$

Using Equation 4.4, we have:

$$\begin{aligned}
\lim_{b \to 0} I(\mathbf{X}^b, \mathcal{T}) &= \lim_{b \to 0} H(\mathbf{X}^b) - H(\mathbf{X}^b \mid \mathcal{T}) && (5.13)\\
&= H(\mathbf{X}) - H(\mathbf{X} \mid \mathcal{T}) - D \log b + D \log b && (5.14)\\
&= H(\mathbf{X}) - H(\mathbf{X} \mid \mathcal{T}) && (5.15)
\end{aligned}$$

By substituting with Equation 5.3, we obtain

$$\lim_{b \to 0} \sigma(V^b) = \sigma(V) \qquad (5.16)$$

We apply Equation 5.7 to obtain the lemma. □

4 Approximate View Strength

We now have a functional definition of view strength. At this point, we could easily envision a greedy heuristic to detect the top K views in a database. We start with simple views, based on one dimension. We then add columns, one by one. To test if a column X is worth adding to a view V, we compute the strength $\sigma(V \cup \{X\})$. If the result is high enough, we keep the candidate. If not, we discard it. We will present such an algorithm in Section 5. However, we must first discuss how to compute $\sigma(V \cup \{X\})$.

Equations 5.3 and 5.7 describe several methods to compute the strength of a view. Nevertheless, none of these fit iterative algorithms. Suppose that we wish to compute the strength of a view V, then the strength of another view $V \cup \{X\}$. These equations give us no opportunity to share computations: we must obtain $\sigma(V)$ and $\sigma(V \cup \{X\})$ separately. Furthermore, both expressions are expensive, as they involve group-by queries over the whole database. Therefore, we need an alternative, iterative formulation for the strength of a view. Our solution is to exploit the Mutual Information chain rule, as follows:

Lemma 5.3. *Consider a view $V = \{X_1, \ldots, X_i\}$, and a target T. For any column X_{i+1}:*

$$\sigma(V \cup \{X_{i+1}\}) = \sigma(V) + I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_1, \ldots, \mathcal{X}_i) \tag{5.17}$$

Proof. This lemma is a direct consequence of Equation 4.7. □

Recall from Chapter 4 that $I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_1, \ldots, \mathcal{X}_i)$ expresses the conditional mutual information. It describes the dependency between $\mathcal{X}_{i+1}$ and $\mathcal{T}$ given the variables $\mathcal{X}_1, \ldots, \mathcal{X}_i$. Recall also that the variables on which we condition can both strengthen or weaken the dependency. The value of $I(\mathcal{X}_{i+1}; \mathcal{T})$ may be high, but $I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_1, \ldots, \mathcal{X}_i)$ may be low. Oppositely, $I(\mathcal{X}_{i+1}; \mathcal{T})$ may be low and $I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_1, \ldots, \mathcal{X}_i)$ high.

Unfortunately, computing the conditional mutual information is expensive - in fact it is as expensive as computing $\sigma(V \cup \{X_{i+1}\})$ directly. However, we can use an approximation. We introduce the following scheme:

$$I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_1, \ldots, \mathcal{X}_i) \approx I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_i) \tag{5.18}$$

We simply ignore the high order dependencies. This approximation is naive, but lets us compute the strength of our candidates much faster:

$$\begin{aligned} \sigma(V \cup \{X_{i+1}\}) &= \sigma(V) + I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_1, \ldots, \mathcal{X}_i) \\ &\approx \sigma(V) + I(\mathcal{X}_{i+1}; \mathcal{T} | \mathcal{X}_i) \end{aligned} \tag{5.19}$$

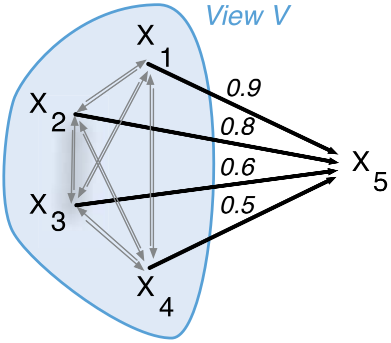


Figure 5.3: Example of co-dependency graph with 5 dimensions. To approximate the strength of $V \cup \{X_5\}$, we add the weight of edge (X_4, X_5) to V's strength - in this case 0.5.

Thanks to this approximation, we can *stage* the view strength computations. We operate in two steps: one offline step and one online step. Offline, we compute the conditional mutual information $I(\mathcal{X}_j;\mathcal{T}|\mathcal{X}_i)$ between every pair of variable $(\mathcal{X}_i,\mathcal{X}_j)$. We call the resulting structure *co-dependency graph*. In this directed graph, the vertices represent the dimensions, and the edges represent the conditional mutual information. Online, we obtain the view strength iteratively, exploiting Equation 5.19. To compute the strength of a view $V \cup \{X_{i+1}\}$ with $V = \{X_1,\dots,X_i\}$, we fetch the value of $I(\mathcal{X}_{i+1};\mathcal{T}|\mathcal{X}_i)$ in the co-dependency graph and add it to V's strength. We illustrate this method in Figure 5.3. Previously, computing $\sigma(V \cup \{X_{i+1}\})$ involved heavy groupings and aggregations on the whole dataset. Now, we simply perform a lookup in a graph with N edges, where N is the number of columns in the database.

Note that our approximation has a drawback: it depends on the order in which we include the variables in the view. If we enrich a view by successively adding variables X_1, X_2 and X_3, then we obtain a different strength than if we incorporate X_3, X_2 then X_1. Similarly, in Equation 5.19, we obtain different approximations if we change the indexing of the dimensions $\mathcal{X}_1,\dots,\mathcal{X}_i$. For more robustness, we introduce a "pessimistic" variant:

$$\sigma(V \cup \{X_{i+1}\}) \approx \min_{n\in[1,i]} \sigma(V) + I(\mathcal{X}_{i+1};\mathcal{T}|\mathcal{X}_n) \tag{5.20}$$

Instead of adding the strength $I(\mathcal{X}_{i+1};\mathcal{T}|\mathcal{X}_i)$, where $\mathcal{X}_i$ is the last variable inserted, we add $I(\mathcal{X}_{i+1};T|\mathcal{X}_n)$, where $\mathcal{X}_n$ is the variable which weakens $\mathcal{X}_{i+1}$ the most.

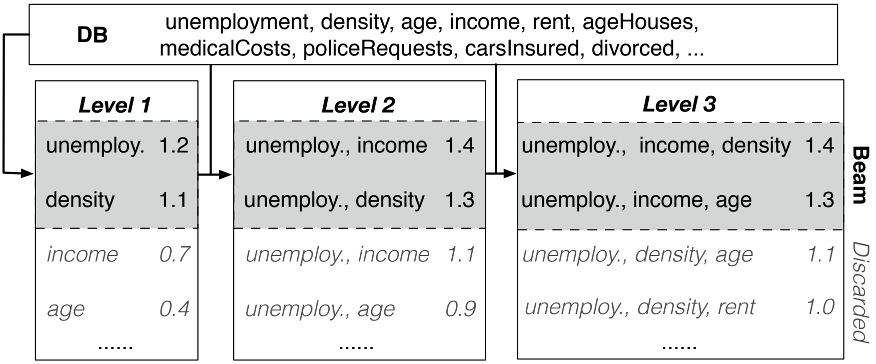

DB
unemployment, density, age, income, rent, ageHouses, medicalCosts, policeRequests, carsInsured, divorced, ...
Level 1
unemploy. 1.2
density 1.1
income 0.7
age 0.4
......
Level 2
unemploy., income 1.4
unemploy., density 1.3
unemploy., income 1.1
unemploy., age 0.9
......
Level 3
unemploy., income, density 1.4
unemploy., income, age 1.3
unemploy., density, age 1.1
unemploy., density, rent 1.0
......
Beam
Discarded

Figure 5.4: Example of Beam Search, with $D = 3$ and beam size $B = 2$

5 Practical Column selection

This section presents our view search strategy. Our aim is to find the top K views with at most D columns. If our database includes N dimensions, our search space contains $\sum_{n \leq N} \binom{N}{n} = 2^N$ combinations, which is clearly impractical. Therefore, we resort to a greedy, level-wise heuristic.

5.1 Base algorithm

Our algorithm is based on *beam search*, illustrated in Figure 5.4. To initialize the algorithm, we compute the strength of each variable separately. We sort the candidates, and keep the top B elements. We call this set the *beam*, greyed in the figure. Then, we generate new candidates, by appending each variable of the database to each variable in the beam. We obtain views with two columns. We compute the strength of these views, keep the top B strongest and discard the others. This gives us a new beam. We repeat the procedure until the views in the beam contain D variables, or the views stop improving. Algorithm 1 presents the full procedure.

Thanks to our strategy, we avoid exploring an exponentially large search space. Instead, we compute the strengths of at most $N.B$ candidates at each level. The size of the beam lets us control the trade-off between accuracy and runtime. With a small beam, we evaluate less candidates, and thus terminate earlier. Oppositely, a large beam lets us explore more candidates. Let us explain why this is necessary. At each level of the algorithm, we discard the views which are too weak to reach the top B candidates. We assume that if a combination of columns is weak at level i, then it will be weak at all subsequent levels. Unfortunately, this assumption rarely

Algorithm 1 Beam Search for view selection

```
function TopViews($K$, $D$, $B$, $DB$)
    Beam $\leftarrow \{\}$
    for $i \in [1, D]$ do
        Cand $\leftarrow \{\}$, Scores $\leftarrow \{\}$
        for $V \in$ Beam do
            for $X \in$ columns($DB$) do
                Cand $\leftarrow$ Cand $\cup \{V \cup \{X\}\}$
                Scores $\leftarrow$ Scores $\cup\, \sigma(V \cup \{X\})$
            end for
        end for
        Beam $\leftarrow$ findTopK(Cand, Scores, $B$)
    end for
    return findTopK(Cand, Scores, $K$)
end function
```

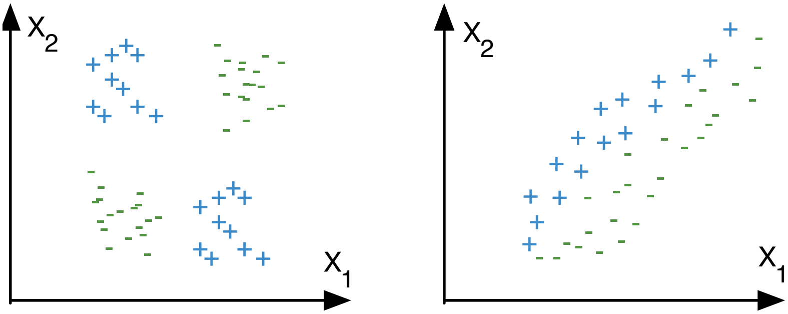


Figure 5.5: Limit cases of the beam search strategy. The variables X_1 and X_2 represent two dimensions. The symbol and color of the plots represent the value of the target.

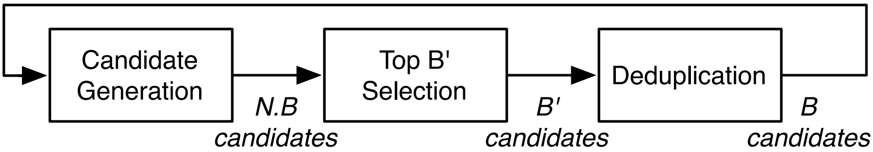


Figure 5.6: Beam search augmented with a deduplication step. We display in italic the size of the intermediate results. Note that $N \cdot B \geq B' \geq B$.

holds: we can form strong views by combining weak columns; there are “jumps” in the search space. Consider for instance the two classic scenarios pictured in Figure 5.5. The dimensions X_1 and X_2 taken in isolation are weak: we can infer no useful information about the target from either of them. However, their combination is very interesting. Equivalently, the views $\{X_1\}$ and $\{X_2\}$ have a very poor strength, but $\{X_1, X_2\}$ is an excellent candidate. If the beam is too tight, we may discard $\{X_1\}$ and $\{X_2\}$ early because of low scores. We lose the opportunity to discover $\{X_1, X_2\}$. Therefore, we recommend to set $B > K$. During our experiments, we obtained excellent results with $B \geq 2 \cdot K$ (cf. Section 7.3).

5.2 Approximations and Refinements

In total, we evaluate the strength of $B \cdot N$ candidates for the D levels of the beam search. To carry out this computation, we can either use the exact formulation of strength, as shown in Equation 5.3, or use the approximation scheme presented in Section 4. In Claude's implementation, we opted for a hybrid approach. We perform the first two levels of search with the exact strength (which is equivalent to building the co-dependency graph). Then, for all subsequent steps, we use the approximations. Finally, we revert to the exact strength for the top k ranking, at the very end of the procedure. Thanks to this method, we obtain significant speed gains at little accuracy cost.

5.3 Deduplication

Our algorithm seeks strong views. In some cases however, it may be preferable to have weaker but more diverse views. To deal with those cases, we introduce an optional *deduplication* step, during which we reduce the number of views with an algorithm from the literature. As pictured in Figure 5.6, we run this procedure at the end of each beam search iteration. By definition, deduplication reduces the number of candidates. Therefore, to obtain B views at the end of the algorithm, we must generate $B' > B$ views beforehand. A low B' yields more variety, while a high B' may lead to stronger views.

Authors have proposed a quantity of methods to deduplicate itemsets in the pattern mining literature [108, 96]. We opted for a simple compression-based approach. First, we compute the dissimilarity between every pair of views with the Jaccard dissimilarity. Given two views V_i and V_j, it is defined as follows: $d_J(V_i, V_j) = |V_i \cap V_j| / |V_i \cup V_j|$. We then cluster the resulting matrix with Partitioning Around Medoids, an algorithm of type k-medoids. We refer the interested reader to Section 1.2.2 of Chapter 3 for more details.

6 Detecting Points Of Interest

We previously described how to find strong views. We now explain how to identify P points of interests for each of these views.

We instantiate POIs by a well-known analysis called *subgroup discovery* [49, 105], which can be formulated as follows: given a set of tuples, a target column and a measure of exceptionality, detect sets of tuples for which the target behaves exceptionally. In our case, we instantiate the exceptionality measure with divergence. As pointed our by van Leeuwen and Knobbe [95], we can also solve the subgroup discovery problem with beam search. Let V represent the view to analyze. As the variables are binned, we can form a grid over V, as shown in Figure 5.2. We denote by b the number of bins for each variable. To initialize the algorithm, we compute the divergence of each cell and keep the top B_{POI} most divergent ones. We obtain our beam. We then "drill" into each of these cells: we decompose them into smaller cells by splitting the edges into b bins. We evaluate the new candidates and keep the top B_{POI} most divergent. We reiterate until the algorithm converges. As shown in the subgroup discovery literature [105, 95], we can generalize this method to binary and nominal data. For each distinct level x_i of a variable X, we create two groups: tuples for which $X = x_i$, and tuples for which $X \neq x_i$.

In practice, KL-based approaches tends to favor smaller regions. Therefore, Beam Search may converge late, or not at all. A practical solution is to set a minimum count threshold. Alternatively, we can alter our model to take the size into account [95]. Let R represents a region with count $|R|$, and $|DB|$ represent the number of tuples in the database. We introduce the *weighted* deviation $\delta_w(R) = |R|/|DB| \times \delta(R)$. This new score introduces a penalty for small POIs.

Dataset	Columns	Rows	#Views	#Variables
MuskMolecules	167	6,600	22	18
Crime	128	1,996	20	17
BreastCancer	34	234	10	13
PenDigits	17	7,496	9	10
BankMarketing	17	45,213	11	8
LetterRecog	16	20,000	10	12
USCensus	14	32,578	10	7
MAGICTelescope	11	19,022	1	10

Table 5.1: Characteristics of the datasets. The last two columns are used for comparison with 4S, cf. Section 7.3.

View	Score (normalized)
Police.Overtime, Pct.Vacant.Boarded, Pct.Race.White	0.51
Pct.Families.2.Parents, Pct.Race.White, Police.Requests.Per.Officer	0.49
Pct.Police.White, Pct.Police.Minority, Pct.Vacant.House.Boarded	0.37
Pct.Empl.Profes.Services, Pct.Empl.Manual, Pct.Police.On.Patrol	0.37
Pct.Retired, Pct.Use.Public.Transports, Pct.Police.On.Patrol	0.35
Pct.Recently.Moved, Population.Density, Police.Cars	0.34

Table 5.2: Example of views generated by Claude for the US Crime dataset.

7 Experiments

We now present our experimental results. All our experiments are based on 8 datasets from the UCI Repository, described in Table 7.6. The files are available online[1]. In several experiments, we report the *normalized* view strength instead of the usual strength. If V is a view with entropy $H(V)$, we obtain it as follows: $\sigma_{norm}(V) = \sigma(V)/H(V)$. The advantage of this this measurement is that varies between 0 and 1.

7.1 Detailed Example: Crimes in the US

In this section, we showcase Claude with a real-life example: we analyze the Communities and Crime dataset from the UCI repository[2]. Our aim is to understand which US cities are subject to violent crimes. Our database

[1] archive.ics.uci.edu/ml/

[2] archive.ics.uci.edu/ml/datasets/Communities+and+Crime

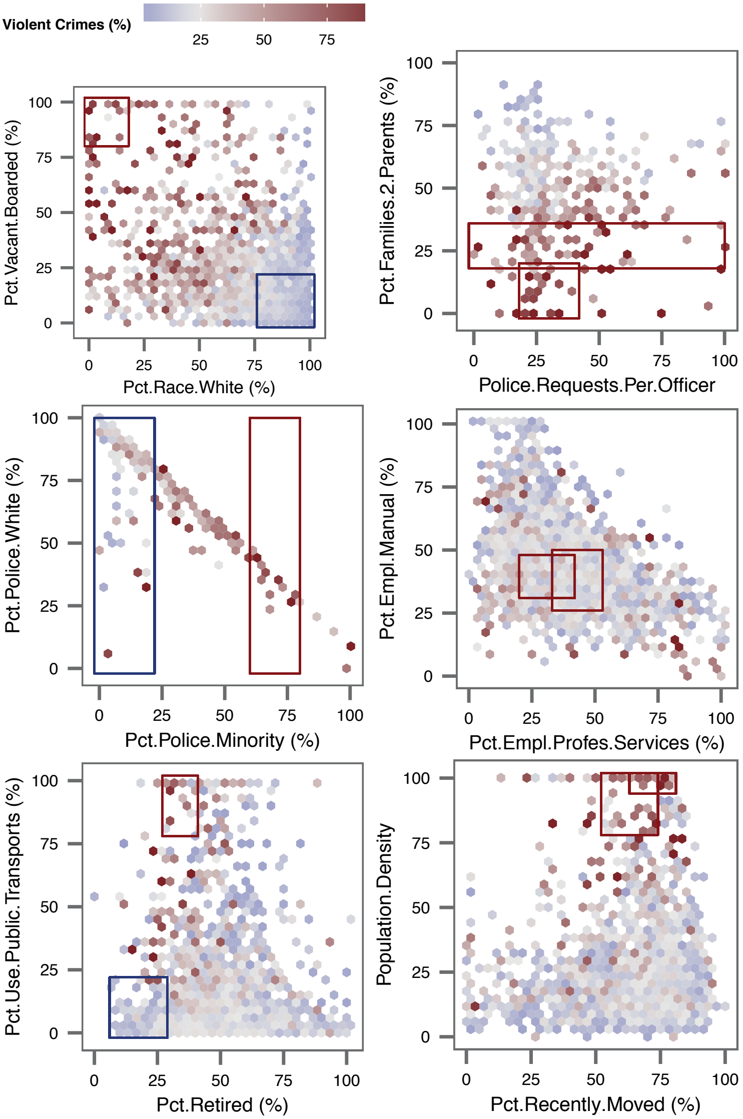


Figure 5.7: Heatmaps of the US Crime Dataset, based on Claude's output. Each box represents a Point of Interest.

compiles crime data and socio-economic indicators about 1994 communities, with a total of 128 variables. The data comes mostly from the 90's, and it was provided by official US sources - among others, the 1990 US census and the 1995 FBI Uniform Crime Report. All the variables are normalized to have a minimum of 0 and a maximum of 100.

We generated $K = 100$ views with up to $D = 3$ dimensions, both with and without deduplication. We present a selection of views in Table 5.2, along with 2-dimension heat maps in Figure 5.7. Observe that strong views have a visual signature: in the top two maps, the blue and red areas are neatly separated. In the bottom two views, the distinction is less clear.

The first view of Table 5.2 is the best one we found: `Police.Overtime, Pct.Race.White, Pct.Vacant.Boarded`. It has a score of 0.51. which means that these three variables contain 51% of the target's information. The columns `Police.Overtime` and `Pct.White.Race` respectively describe the average time overworked by the police and the percentage of caucasian population. The third variable, `Pct.Vacant.Boarded` was surprising to us: it describes the percentage of vacant houses which are boarded up. How does this relate to crime? We could assume that boarded houses are associated with long term abandon, and thus, poverty. The top-left plot of Figure 5.7 shows the relation between race, boarded houses and crime. Observe that the variables complement each other: a high proportion of caucasians may or may not lead to low crime. However, a high proportion of caucasians combined with a low rate of boarded house correspond to safe areas, while few caucasians and many boarded houses correspond to more violent communities.

Our second view shows that cities with more monoparental families tend to be more violent: the correlation is clearly visible, and both POIs point to the bottom of the chart. However, close inspection also reveals surprises: a few communities have a relatively high number of two-parents families, but also high indicators of police requests and crime (in the top right corner of the chart). Manual queries reveal that many of these cities are located in the suburbs of Los Angeles, and contain a majority of Hispanics. Does this explain the peculiarity? We leave this question open for future investigations. We see that some findings come from the recommendations directly while others are serendipitous. But in both cases, Claude lets us discover "nuggets" with little prior knowledge and few assumptions.

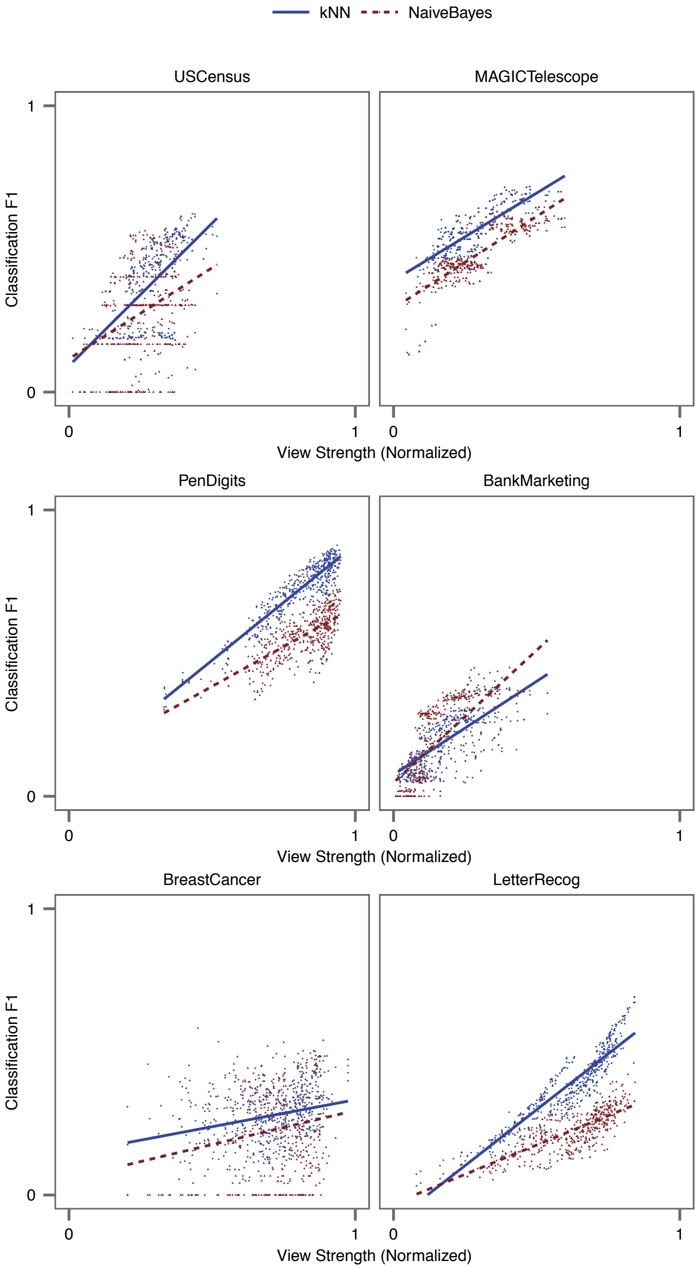


Figure 5.8: Strength vs. Classification accuracy for 500 random views. We obtained the blue and red lines with linear regression.

7.2 View Strength and Prediction

In this section, we show experimentally that our notion of view strength "works", e.g. that strong views effectively provide information about the target column. To verify this assumption, we simulate users with statistical classifiers. Consider a view V over a database. If a classifier can predict the value of the target from V's columns, then V is informative. Oppositely, if the classifier fails, then V is potentially uninteresting. In a nutshell, we should observe a positive correlation between view strength and classification accuracy.

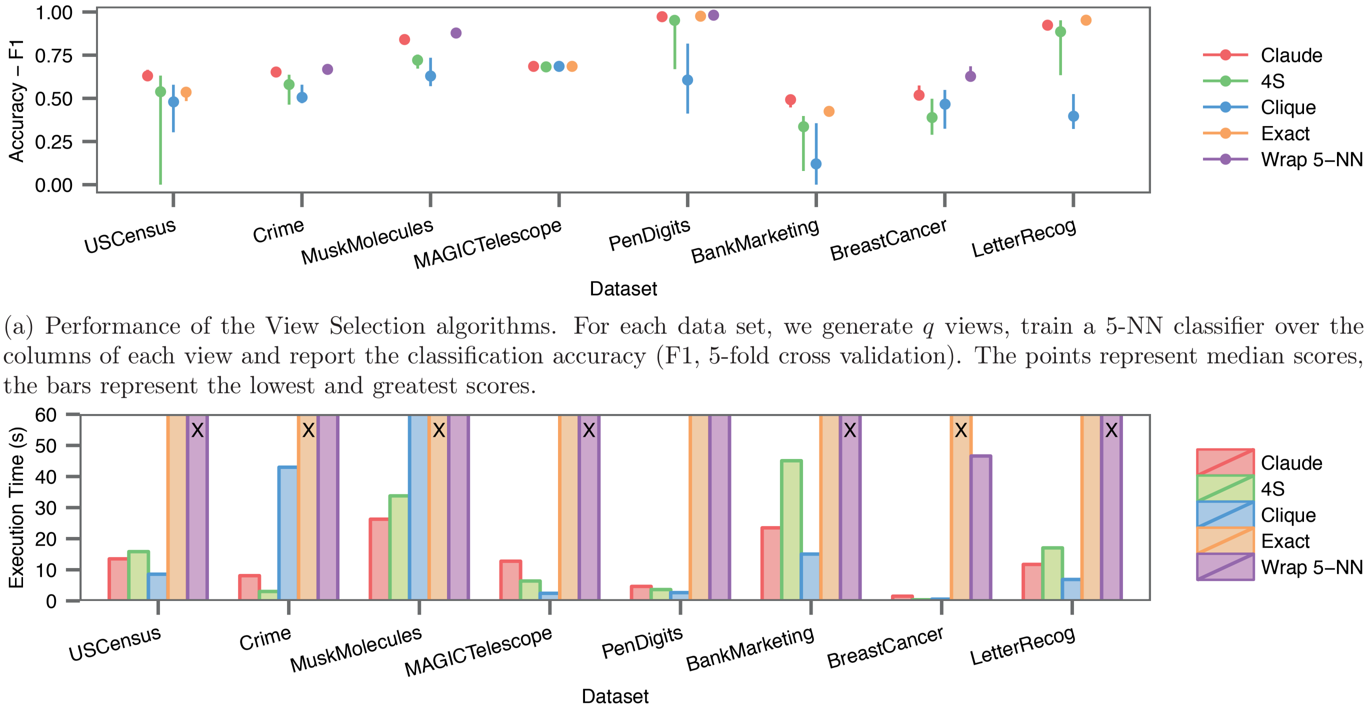
We now detail our experiment. We chose three datasets from the UCI repository. For each dataset, we generated 500 random views and measured their strengths. We then trained classifiers on each of these views, and measured their performance. We report the results in Figure 5.8. We chose two classification algorithms: Naive Bayes, and 5-Nearest Neighbors. We chose those because they contain no built-in mechanism to filter out irrelevant variables (as opposed to, e.g., decision trees). We measure classification performance with 5-fold validation, to avoid the effects of overfitting.

In all three cases, we observe a positive correlation between the strengths of the views and the accuracy of the predictions. We confirm these observations with statistical tests: the coefficients of determination (R^2) vary between 0.11 and 0.84, which indicates the presence of a trend (despite some variability). Furthermore, the p-values associated to the coefficients are all under 10^{-3}, this gives us excellent confidence that the strength influences positively the prediction accuracy. In conclusion, strong views are indeed more instructive.

7.3 View Selection

We now evaluate Claude's output and runtime in detail. In this section, we verify if Claude's algorithm produces good views in a short amount of time. To do so, we compare it to four methods, three of which come from the machine learning literature. Our first baseline, `Exact`, is similar to Claude, but we removed the approximation scheme presented in 4 - instead we compute the exact the mutual information, as in Equation 5.3 The method should be slower, but more accurate.

The second algorithm, `Clique`, is a top-down approach inspired by recent work on pattern mining [107]. We build a graph where each vertex i represents a column D_i, and each edge (i, j) represents the view $\{D_i, D_j\}$. We then eliminate all the edges except those which represent the top B



(a) Performance of the View Selection algorithms. For each data set, we generate q views, train a 5-NN classifier over the columns of each view and report the classification accuracy (F1, 5-fold cross validation). The points represent median scores, the bars represent the lowest and greatest scores.

(b) Execution time of the View Selection algorithms. A X symbol indicates that the experiment did not finish within 3,600 seconds.

Figure 5.9: Experiments with the view selection algorithm.

views. To detect views with $D > 2$ columns, we seek cliques in this degenerated graph. We used the `igraph` package from R. We expect this algorithm to be very fast, but less accurate.

The third method, `Wrap 5-NN`, is a classic feature selection algorithm [36]. The idea is to train a 5-Nearest Neighbor classifier with increasingly large sets of variables. We first test each variable separately, and keep the column which led to the best prediction. Then we keep adding variables in a breadth-first manner, until the quality of the predictions stops increasing or we reach n variables. Our implementation is based on the `class` package from R. We modified the original algorithm to maintain and update q distinct sets of variables instead of just one. We chose the nearest neighbor algorithm because it is fast, and it gave us good performance, as shown in 7.2. We expect this algorithm to be very slow, but close to optimal.

Finally, the last method, `4S` is a state-of-the-art subspace search method from the unsupervised learning literature [70]. The aim of the algorithm is to detect "interesting" subspaces in large databases, independently of a target variable. To do so, it seeks groups of variables which are mutually correlated, with sketches and graph-based techniques. We used the author's implementation, written in Java. We expect the algorithm to be very fast and reasonably accurate.

We use 8 public datasets, presented in Section 7. For a fair comparison, we must ensure that each algorithm generates the same number of views (K) with the same number of variables (D). However, we have no way to specify these parameters a priori with 4S, because the algorithm has a built-in mechanism to pick optimal values. Therefore, we run 4S first on each dataset, we let it chose K and D, and we use these values for the remaining algorithms. We report the obtained parameters in Table 7.6.

We implemented Claude in R, except for some information theory primitives written in C. For practical reasons, we interrupted all the experiments which lasted more than 1 hour. Our test system is based on a 3.40 GHz Intel(R) Core(TM) i7-2600 processor. It is equipped with 16 GB RAM, but the Java heap space is limited to 8 GB. The operating system is Fedora 16.

7.3.1 Accuracy

In Figure 5.9a, we compare the quality of the views returned by each algorithm. For each competitor, we generate K views with D variables, train a classifier on each view and measure the quality of the predictions. For the classification, we use both Naive Bayes and 5-Nearest Neighbors, and report the highest score. We measure accuracy with the F1 score on

5-fold cross validation; higher is better.

The method `Wrap 5-NN` comes first for all the datasets on which it completed. This is not surprising since the algorithm optimizes exactly what we measure: `Wrap 5-NN` is our "gold standard". Our two algorithms, `Claude` and `Exhaustive`, come very close. This indicates that both algorithms find good views, and that our approximation scheme works correctly. The algorithms `4S` and `Clique` come much lower. As `4S` is completely unsupervised, we cannot expect it to perform as well as the other approaches. The assumptions behind `Clique` are apparently too naive.

7.3.2 Runtime

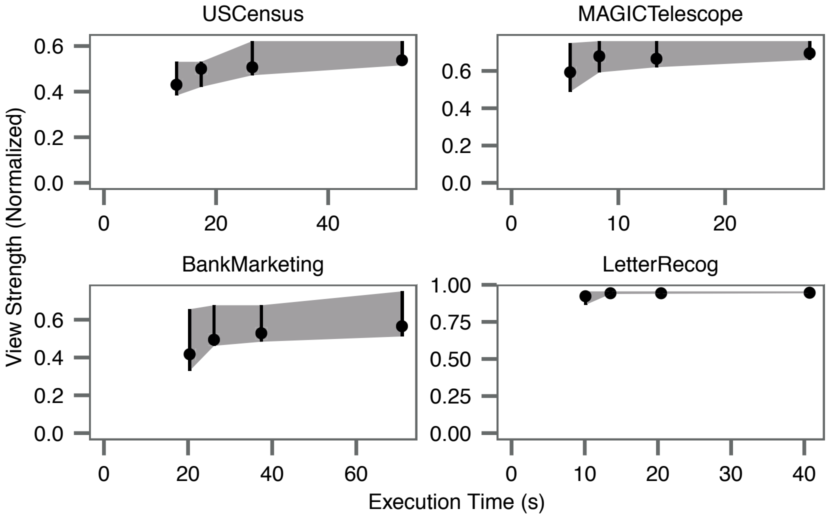
Figure 5.9b shows the runtime of our experiments. The algorithms `Exact` and `Wrap 5-NN` are orders of magnitude slower than the other approaches. The remaining three approaches are comparable: depending on the datasets, either `Clique` or `4S` come first. Claude comes first for `MuskMolecules`, and close second for all the other datasets. In conclusion, Claude is comparable to its competitors in terms of runtime, but it generates better views.

7.4 Impact of the beam size

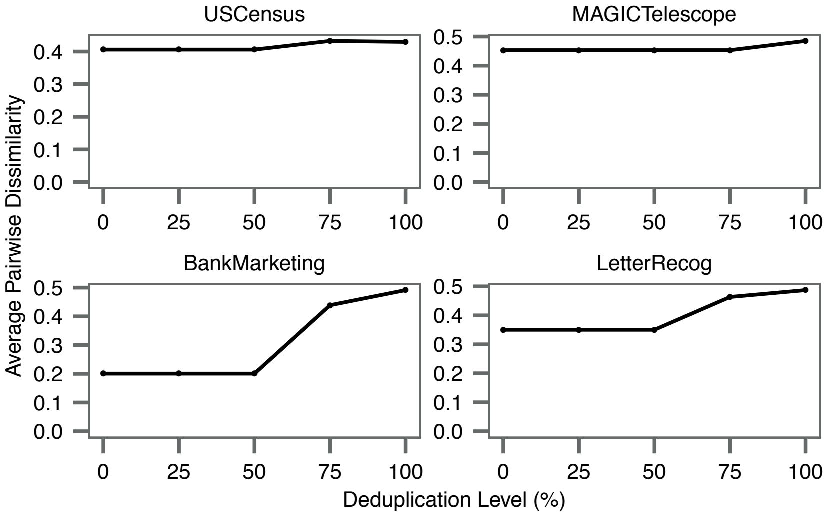
Figure 5.10a shows the impact of the beam size B on Claude's performance, for 4 databases. To obtain these plots, we ran Claude with $K = 25$ and $D = 5$, and varied B between 25 and 250. We observe that smaller beam sizes lead to lower execution times, while larger beam sizes lead to stronger views. However, the heuristic converges fast: we observe little to no improvement for B greater than 50.

7.5 Impact of the deduplication

We show the impact of our deduplication strategy in Figure 5.10b. We ran Claude with $K = 25$ and $D = 5$ and increased the level of deduplication, i.e., varied the value of B' between B and $N.B$ (cf. Section 5.3). A level of 0% means that $B' = B$. A level of 100% means that $B' = N.B$. To measure the diversity of the views, we measured the Jaccard dissimilarity between every pair of views and averaged the results. We observe that the strategy works in all four cases, but with different levels of efficiency. In the `BankMarketing` case, our strategy almost doubles the pairwise



(a) Impact of the beam size on the execution time and view strength. For each dataset, we generated 25 views with beam size 25, 50, 100 and 250. The points represent the medium scores, the bars represent the lowest and greatest scores.



(b) Impact of the deduplication. We generated 25 views for each dataset. The y-axis presents the average Jaccard dissimilarity between every pair of views.

Figure 5.10: Experiments with beam size and deduplication.

dissimilarity of the views. The effect is much lighter on datasets with few columns, such as USCensus and MAGICTelescope.

7.6 POI Detection

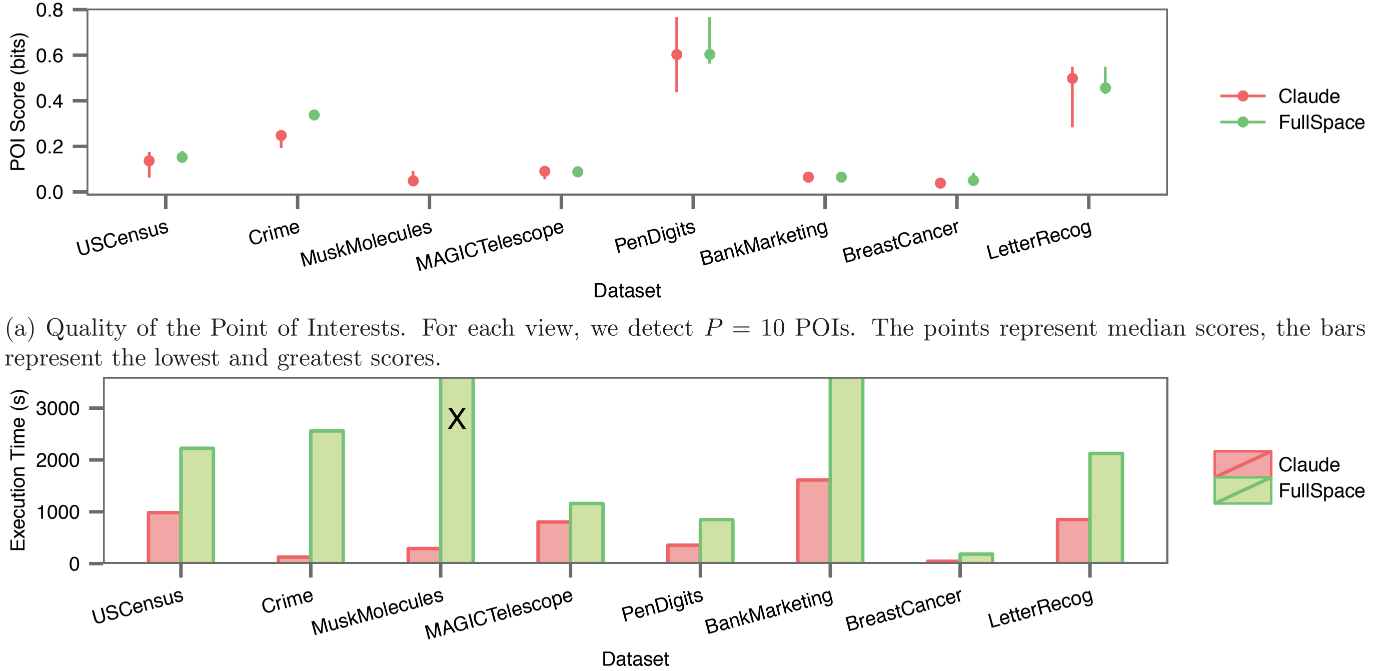
In this section, we evaluate Claude's POI detection strategy. We compare two approaches. The first approach is the algorithm presented in the chapter: first we search K views, then we return P POIs per view. The second approach, FullSpace, is the method used in much of the recent Subgroup Discovery literature [95, 29]. The idea is to apply Beam Search on the whole database directly. Instead of seeking P POIs in K projections, we seek $K.P$ selections from the full column space; we skip the view selection step. We use the same datasets as previously. Our default parameters are $K = 25$, $D = 5$, $B = 50$ and $P = 10$. To set the beam size, we use a rule of thumb: $B_{POI} = 2.k$ (B_{POI} is the beam used for POI detection, not for view search). To gather sufficient data, we raise our time limit to 2 hours.

Figure 5.11a compares the quality of the POIs found by both algorithms. The strategy FullSpace gives slightly better results on Crime and PenDigits, but the difference is close to null. The scores are similar on all the other datasets. We conclude that Claude's POIs are very close to those found by a state-of-the-art Subgroup Discovery approach. Figure 5.11b compares the runtimes of both approaches. We observe that Claude is much faster than FullSpace. The difference grows with the number of columns: the runtimes are almost similar for datasets with few columns (MAGICTelescope), but Claude is considerably faster for larger databases (more than an order of magnitude difference for Musk-Molecules). This is a positive side-effect of our approach: decoupling view search and POI extraction allows us to find subgroups faster in high dimensional datasets.

8 Related Work

8.1 SQL Query Recommendation

We identify two types of approaches: *human-driven* systems and *data-driven* systems. Human-driven systems learn from user feedback. For instance, Chatzopoulou et al. make recommendations from query logs, similarly to search engines [20]. In Explore-by-Example, the system infers queries from examples provided by the user [26]. Sarawagi's method builds a maximum entropy model over the database from the user's history [83].



(a) Quality of the Point of Interests. For each view, we detect $P = 10$ POIs. The points represent median scores, the bars represent the lowest and greatest scores.

(b) Execution time of the POI detection. A X symbol indicates that the experiment did not finish within 7,200 seconds.

Figure 5.11: Experiments with POIs.

Bonifati et al. propose a similar method to recommend joins [15]. Claude competes with neither of these approaches, since it uses the content of the database only.

Our work is closer to data-driven data recommendation. The general idea is to build a statistical model of the database, and find regions which behave unexpectedly. Sarawagi et al. have published seminal work on this topic [82]. But their system target a different use case. They assume that the database contains only a dozen well-known dimensions, and that the challenge is to drill-in correctly. Therefore, they make no recommendations concerning columns, and they focus on micro-patterns. In our model, databases can contain more than a hundred dimensions. The whole challenge is to present how they relate to each other. We are interested in broad dependencies, not local exceptions. Gradient analysis [42, 27] is close to our work, but it also focuses on micro-deviations. Finally, Dash et al. have proposed a method to reveal surprising subsets in a faceted search context [24]. This method is related to Claude, but it targets document search, not OLAP views.

8.2 Projection Search

Authors from the data visualization literature have proposed methods to detect the "best" projections of multidimensional data sets, such as Projection Pursuit [32], Scagnostics [104], or Tatu et al.'s relevance measures [91]. Such methods would form excellent complements for Claude's recommendations. Nevertheless, most of them focus on 2-dimensional scatterplots, are limited to continuous variables, and involve materializing and analyzing every possible 2D projection of the data.

8.3 Feature Selection, Subspace Search

Choosing which variables to use for classification or regression is a crucial problem, for which dozens of methods were proposed [36]. Similarly to Claude, some of these methods rely on mutual information [74]. Nevertheless, the objective is different. A feature selection algorithm seeks *one* set of variables, on which a statistical predictor will perform optimally. Claude seeks several, small sets of variables, simple enough to be interpreted by a humans. In fact, Claude is halfway between inference and exploration. On the unsupervised learning side, our work is close to subspace search. The idea is detect subspaces where the data is clustered distinctly [48, 70]. We compare Claude to state-of-the-art methods in our Experiments section.

9 Summary

We formalized what makes a data warehouse query “informative”, using the mutual information and the Kullback-Leibler divergence. We presented practical methods to detect these queries, using carefully designed approximations. Finally, we presented and evaluated Claude, a system based on these ideas. The methods we developed for this study have broader applications than the strict realm of query recommendation. Our column selection scheme competes with state-of-the-art feature selection methods. Also, the idea to decouple column selection from subgroup search could benefit a wide range of subgroup discovery algorithms.

Chapter 6

Blaeu: Maps to Navigate the Query Space

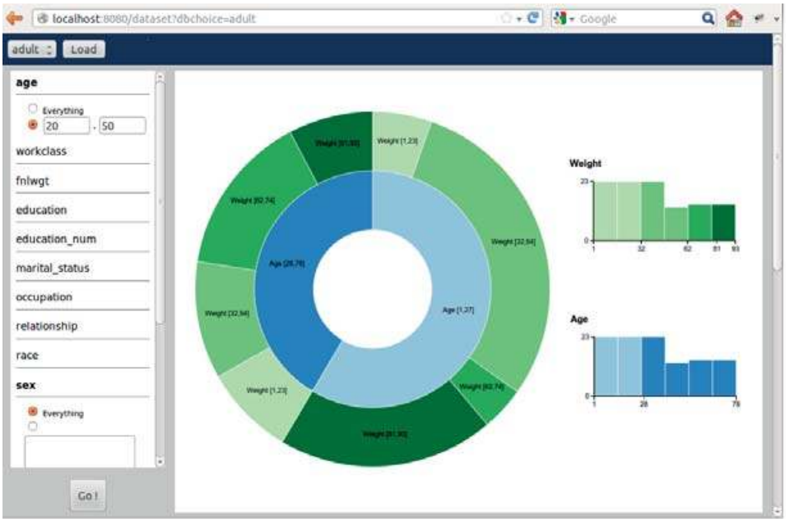
1 Introduction

Previously, we introduced Claude, a query recommendation system for data warehouses. The main idea is to exploit the specific nature of the OLAP model: we know that our data contains dimensions and measures, and we know that data warehouse users seek to understand how the measure varies across the dimensions. Consequently, we can exploit the statistical dependencies between the two types of columns.

We now generalize query recommendation beyond the OLAP use case. Here again, we model the database with one large table. However, we make no assumption about the content of this table: a column could be a dimension, a measure, or simply noise. We only assume that somewhere in the database lies an interesting set of tuples.

1.1 Contributions

In this chapter, we introduce Blaeu, a system to write queries through cluster analysis. With Blaeu, users proceed in a step-by-step, interactive manner. At each step, the system generates presents a *map* of the data, obtained through clustering. This map gives a summary of the database. Furthermore, it gives options for query refinements: if the users are interested in one of the clusters, they can click on it. The system then zooms in, and present a new map. If they are not satisfied, they can zoom out, or request an alternative map, based on other columns. Thanks to these actions, our explorers can effectively *browse* their databases. They discover their content in an assisted, semi-automatic fashion. Figure 6.1 presents a screen capture of Blaeu's interface.



localhost:8080/dataset?dbchoice=adult

adult

age: ○ Everything ● 20 - 50

sex: ● Everything ○

Figure 6.1: Screenshot of Blaeu's interface

Our first contribution is the data mapping model: we expose the link between query recommendation and cluster analysis, and study the mathematical properties of the subsequent query space. Our second contribution is Blaeu's mapping engine. To create data maps, Blaeu must find clusters in the database. Yet, it must be fast enough to support interaction, and it must rely on few parameters. We present three clustering algorithms which address these requirements. The first algorithm, *SimpleMap*, targets small to medium data sets. The second algorithm, *MultiMap*, supports high dimensional datasets with a multi-view approach. The last algorithm, *LightMap*, focuses on speed and interaction. Additionally, we present an aggressive sampling strategy to accelerate mapping. Experiments will reveal that Blaeu can cluster millions of tuples in dozens of subspaces within seconds, on a commodity laptop.

1.2 Outline

The rest of this Chapter is organized as follows. Sections 3, 4 and 5 present our three algorithms. Section 6 discusses how sampling and recycling accelerate Blaeu's mapping engine. We present use cases and experiments in Sections 7 and 8. Finally we compare Blaeu to related work in Section 9 and conclude in Section 10.

2 Data Cartography

In this section, we present the most important concept behind Blaeu: the data map. Through this simple abstraction, users can visualize and query their data. We introduce our terminology and our system.

2.1 Overview

Our users are confronted with an unknown database. This database contains a few special, interesting tuples. If the users saw these tuples, they could recognize then. Yet, they are hidden among thousands of others. How can we help?

With Blaeu, users write queries in a top-down fashion. They start with a wide Select-Project-Join query. Then, they refine it, step by step. At each step, they visualize the current selection, identify the interesting tuples, and narrow their query. Thus, they drill in iteratively, as they discover the database. In principle, users could carry out top-down exploration through any database front-end. However, raw tables are often difficult to read and the number of potential refinements can be huge. Our idea is to guide the users with cluster analysis. At each step, Blaeu decomposes the current selection into clusters. If the users suspects that one of the partitions contains the tuples of interest, they can click on it. Then, Blaeu updates the selection and creates new clusters. The process is repeated until the users are satisfied.

Cluster analysis is a classic data mining technique. Its aim is to partition a data set, such that similar items are grouped, and distinct objects are separated [46]. The literature contains dozens of different ways to define a cluster, we will instantiate the definition later in the chapter. In our case, this method is helpful in two ways. First, it lets Blaeu *summarize* the users' selection: instead of showing long list of tuples, it displays a few clusters. Second, our system uses clustering to *suggest refinements*. If a tuple is interesting, then its neighborhood probably contains other interesting tuples. Oppositely, if a tuple is irrelevant, its neighbors are likely to be irrelevant too. Therefore, grouping similar tuples helps users separate the interesting tuples from the noise. We will refine these arguments in the following section. First, let's show how Blaeu works in practice.

With Blaeu, users explore their data through *data maps*. A data map is an interactive representation of the clusters in a data set. Figure 6.2 illustrates how to work with data maps. Suppose that we explore a database which describes the alumni of a fictional university. To seed the process, we provide a wide query. We select the whole database, projected on the

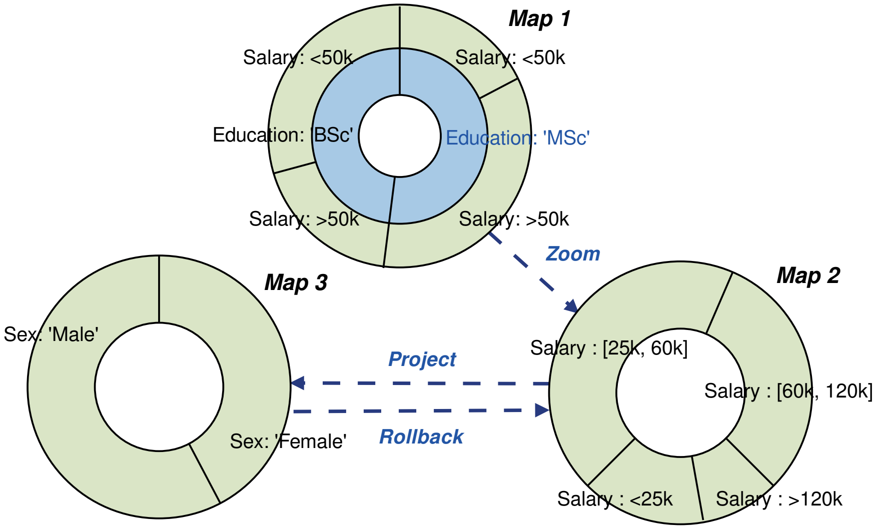


Figure 6.2: "Surfing" the data maps

columns `Salary` and `Education`. Blaeu clusters the data, and it returns a data map, titled Map 1. The map describes four types of individuals: alumni with a Bachelor's and a low salary, alumni with a Bachelor's and high salary, alumni with a Master's and a low salary, alumni with a Master's and a high salary. The size of the slices represent the count of the clusters. We see that the map summarizes our selection. Furthermore, it gives us options: to refine our selection, we can simply click on one of the regions. Then, Blaeu returns a new map. For instance, if we select the region `MSc`, we obtain Map 2. We call this operation a *zoom*.

Blaeu offers two additional primitives: the *projection* and the *rollback*. The projection does not affect the selection of tuples, but it changes the columns used by the map. If we project Map 2 on `Sex`, we obtain Map 3. With a rollback, we go back to a previous state of the system. Thanks to zooms, projections and rollbacks, we can "surf" from one map to another.

Figure 6.3 gives an overview of the system's components. Users interact with a graphical interface. They specify the seed query with an input menu. They zoom with clicks, and project with drags. Internally, all the user actions are translated into a proprietary language, MapQL.

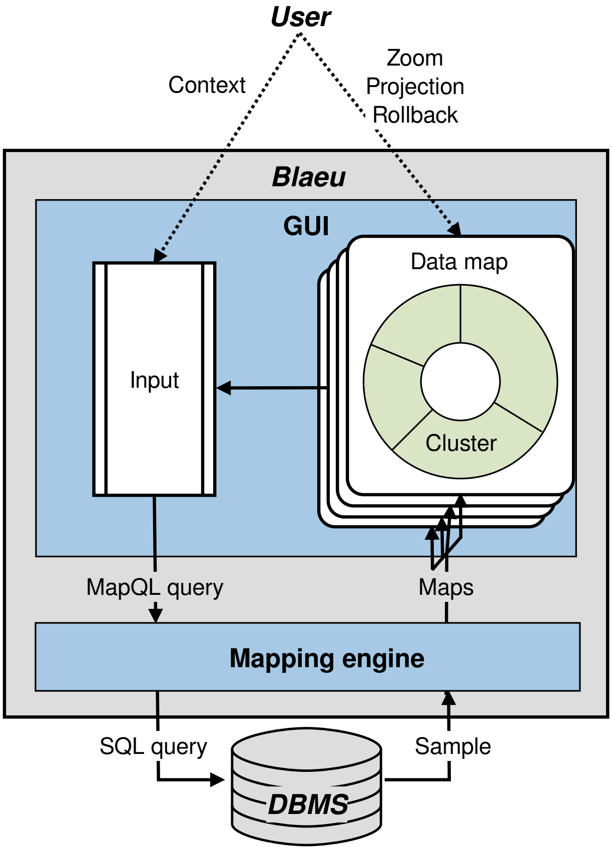


Figure 6.3: High level view of Blaeu's architecture.

2.2 Formalization

We introduced the data map, and presented how to interact with it. We now define these concepts formally.

Definition 6.1. *Let $Q \subset DB$ represent a set of tuples in a database DB. We obtain a partitioning $\mathcal{C} = \{C_1, \ldots, C_K\}$ over Q with cluster analysis. A* data map *is a representation of $\mathcal{C}$ which supports zooms, projections and rollbacks.*

Our definition contains two degrees of freedom: it is oblivious to our choice of clustering method, and it does not specify how to represent the clusters. These choices depend on practical considerations, discussed in further sections.

Each data map is based on a set of tuples Q. We call this set the *context* of the map. To start an exploration session, the user provides a first context Q_0 explicitly. We name it the *seed context.* The subsequent contexts are obtained iteratively, with zooms, rollbacks and projections. At each step i, these operations transform a context Q_i into a new, refined context Q_{i+1}. Let us define the possible transformations with relational algebra:

Definition 6.2. *Let Q_i describe a set of tuples with primary key k, $i \geq 0$. A data map over Q_i supports three types of operations:*

- *Zoom in cluster C_j: $Q_{i+1} = \sigma_{t \in C_j}(Q_i)$*
- *Projection on a set of variables V_p: $Q_{i+1} = \pi_{V_p \cup \{k\}}(Q_i \bowtie Q_0)$*
- *Rollback: $Q_{i+1} = Q_{i-1}$, undefined if $i < 1$*

As we defined the operators in a declarative way, implementation details may vary. For instance, we described the projection with an inner join $Q_i \bowtie Q_0$, to fetch the columns missing from Q_i. In practice, we can bypass this join: we maintain *all* the columns of Q_0 during the whole session, and perform the projections lazily, when we build the maps. This method speeds up the exploration and lets Blaeu operate without any primary key.

Thanks to these definitions, we can formalize Blaeu's expressivity.

Lemma 6.1. *Let $\mathbb{Q}_{SPJ}$ describe the set of all possible Select-Project-Join queries over a database DB. If clus describes all the clusters in the database and col describes all the columns in the database, the set of* *queries that users can express with zooms, projections and rollbacks is the following:*

$$\mathbb{Q}_{Blaeu} = \{\pi_P(\sigma_R(Q)) \mid P \subset col, R \in clus, Q \in \mathbb{Q}_{SPJ}\}$$

Proof. The first query of the session Q_0 is a SPJ query. The property can be derived recursively with definition 6.2. □

This lemma leads to $\mathbb{Q}_{Blaeu} \subset \mathbb{Q}_{SPJ}$: Blaeu's primitives yield SPJ queries. Yet, not all SPJ queries may be expressed. Blaeu *quantizes* $\mathbb{Q}_{SPJ}$, transforming this large continuous space into a small and finite set of options.

Blaeu trades control for accessibility. This approach is useful in two cases. First, it helps users with fuzzy, high level queries. Suppose that our users are seeking "young, well paid individuals" in the alumni database. A classic query language requires that they set precise thresholds on the age and the salary; for instance, less than 35 years old and more than \$100,000 per year. Thus, they need some preliminary knowledge, or some investigation time. In contrast, Blaeu automatically partitions the columns age and salary into semantic groups, such that the users can reach the target tuples with a few zooms. Second, Blaeu supports users with *no* query. These users are browsing; they seek interesting tuples but they ignore where to find them. With a classic SQL-based system, they must make guesses. These guesses can lead to empty or overwhelmingly large result sets. With Blaeu, the users obtain a few query suggestions. Thus, they can assess each option in turn, and pick one according to their preferences.

2.3 Properties

According to Definition 6.1, a data map serves both as output and input. It serves as output because it summarizes the user's current selection. It serves as input because users can refine their queries by clicking on the clusters. The idea to summarize data with clusters has been studied for decades [46]. However, it is less clear how clustering can generate interesting query refinements. In this section, we create a user model, and show under which conditions this property holds.

We model a user by an *utility function* $u : DB \to \{-1, 1\}$. The function takes a tuple t as input, and returns $u(t) = +1$ if t is interesting, $u(t) = -1$ otherwise. To deal with sets, we sum the utility of each tuple. Formally, for a set of tuples $Q \subset DB$, the aggregated utility is $U(Q) = \sum_{t \in Q} u(t)$. Our aim is to suggest interesting subsets of the data. For a given partitioning,

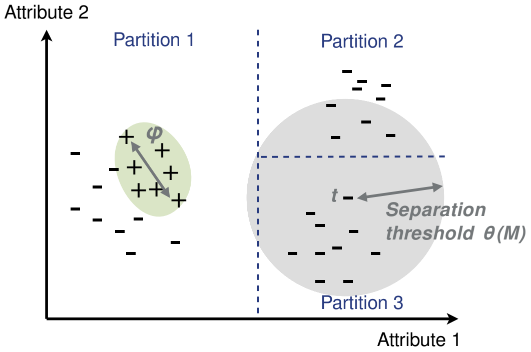


Figure 6.4: Example of informative map. Each point represents a tuple in a two dimensional space, the symbols + and - represent the utility, the blue dashed lines represent clusters. The grey area around tuple t represents the separation threshold $\theta(\mathcal{C})$. Most of Partition 3's tuples are within $\theta(\mathcal{C})$ of tuple t. Therefore, t belongs to Partition 3. Conversely, as t does not belong to Partition 2, the majority Partition 2's tuples are beyond $\theta(\mathcal{C})$.

if at least one of the partitions is interesting, then we reached our goal. We formalize this objective as follows:

Definition 6.3. *Let set $Q \subset DB$ describe a set of tuples, and let $\mathcal{C}$ represent a partitioning $\{C_1, \ldots, C_K\}$ of Q. We say that $\mathcal{C}$ is* informative *if and only if there exists a $C_j \in \mathcal{C}$ such that $U(C_j) > U(Q)$.*

Generating informative partitions would be trivial if we knew our user's utility function. This is not the case, we know almost nothing. However, we can prove the following property: if the interesting tuples are sufficiently close to each other, then recommending *clusters* is a safe bet.

Consider a partitioning of $\mathcal{C}$ over Q, obtained by clustering. From $\mathcal{C}$ we infer a *separation threshold* $\theta(\mathcal{C})$. This function measures how well separated the clusters are. It returns a high value if $\mathcal{C}$'s clusters are tight and far apart. It returns a low value if $\mathcal{C}$'s clusters have a large overlap. We define this threshold as follows:

Definition 6.4. *The* separation threshold *$\theta(\mathcal{C})$ of a partitioning $\mathcal{C}$ is the largest distance such that for any tuple t, for any partition $C_j \in \mathcal{C}$: if at least half of C_j's tuples are within a distance $\theta(\mathcal{C})$ of t, then $t \in C_j$.*

We illustrate this property with Fig. 6.4. Thanks to this notion, we can identify scenarios in which clusters always form interesting recommendations.

Lemma 6.2. *Consider a set of tuples Q with at least one interesting tuple. Let ϕ represent the largest possible distance between two interesting tuples of Q ($\phi = 0$ if there is only one interesting tuple). Any partitioning $\mathcal{C}$ of Q such that $\theta(\mathcal{C}) > \phi$ is informative.*

Proof. Let T represent the set of all the interesting tuples. There is at least one tuple t^* in the database such that $u(t^*) = +1$. We note C^* its cluster in $\mathcal{C}$. Now consider a cluster $C \neq C^*$. For the majority of C's tuples t we have $d(t, t^*) > \theta(\mathcal{C})$. If $\theta(\mathcal{C}) > \phi$, then the majority of C's tuples are outside T. Therefore, $U(C) < 0$. Note that $U(Q) = U(C^*) + \sum_{C \in \mathcal{C} \setminus \{C^*\}} U(C)$. We derive the following: $U(C^*) = U(Q) - \sum_{C \in \mathcal{C} \setminus \{C^*\}} U(C) > U(Q)$. In conclusion, the map $\mathcal{C}$ is informative. □

Intuitively, this lemma states that data maps work when the interesting tuples are similar to each other and the data is clustered. If these conditions are met, then the interesting tuples end up in the same cluster, as in Figure 6.4. Therefore, at least one partition in the map is interesting. Oppositely, if the interesting tuples are far from each other, they may spread over several clusters. In this case, we have no guarantee about the map's interestingness. This property highlights a limitation of our model: if the interesting subset of tuples is large and heterogeneous, then users must decompose their exploration in several stages, discovering at each stage a small, homogeneous selection of tuples.

To avoid confusion, we insist that Blaeu also works when the users are seeking outliers. According to Blaeu, a subset is interesting if its tuples are close *to each other*. This does not mean that the tuples must be close *to the rest of the data*. Therefore, outliers can very well appear in data maps. For Blaeu, an outlier is simply a small, isolated cluster.

2.4 Representation

We now discuss how to represent clusters. Which information about the clusters should we convey? Which visualization method should we use?

A data map should at least provide one identifier for each cluster. Additionally, it should reflect the structure of the partitions: if we use hierarchical clustering, the map should show the inclusions; if we use flat partitions, it should present the clusters side-by-side. Our implementation describes the clusters with bounding boxes (e.g., `Education: 'MSc', Age < 30`), and it organizes them in a tree (we justify these choices in Section 3). Thus, we instantiate the maps with sunburst charts, known to be efficient in this case [109]. Note that several other visualization methods could qualify, based on treemaps (cf. Chapter 2), trees or even raw text.

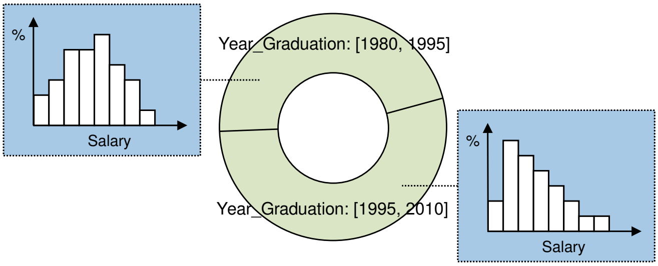


Figure 6.5: Example of map, augmented with additional statistics. For each region, we show a histogram of the variable `Salary`.

In practice, identifiers such as bounding boxes may not provide enough information to assess the content of the clusters. For each partition, Blaeu's front-end provides additional statistics and a few sample tuples. In this chapter, we annotate each cluster with a count, conveyed by the size of the slices. Our prototype provides more options: users can request an aggregate or a histogram over any variable in the database. As an illustration, Figure 6.5 shows how Blaeu displays the distribution of `Salary` for different generations of alumni. In effect, this feature enriches Blaeu's expressivity: it allows grouping and aggregating on top of the queries in $\mathbb{Q}_{Blaeu}$. Thus, users can quickly perform side-by-side comparisons. More generally, this function lets Blaeu emulate traditional OLAP front-ends: the highlighted variable is equivalent to a measure, and the context variables are equivalent to dimensions. In the interface, users request aggregates with mouse-overs.

3 Algorithm 1: Building Maps

In the previous sections, we discussed cluster analysis as an abstract task, we were indifferent to how it was implemented. In this section, we discuss how to make it work in practice. We present a first algorithm, *SimpleMap*, based on existing machine learning methods. We will use this procedure as building block for the following sections.

Blaeu is subject to two contradictory requirements. On one hand, the mapping engine should be accurate, and it should be flexible enough to handle any kind of data, including missing values and categorical attributes. On the other hand, the output should be simple. The descriptions of the clusters must be interpretable by non technicians, and they must be translatable into SQL to express zooms. To avoid setting a fixed number

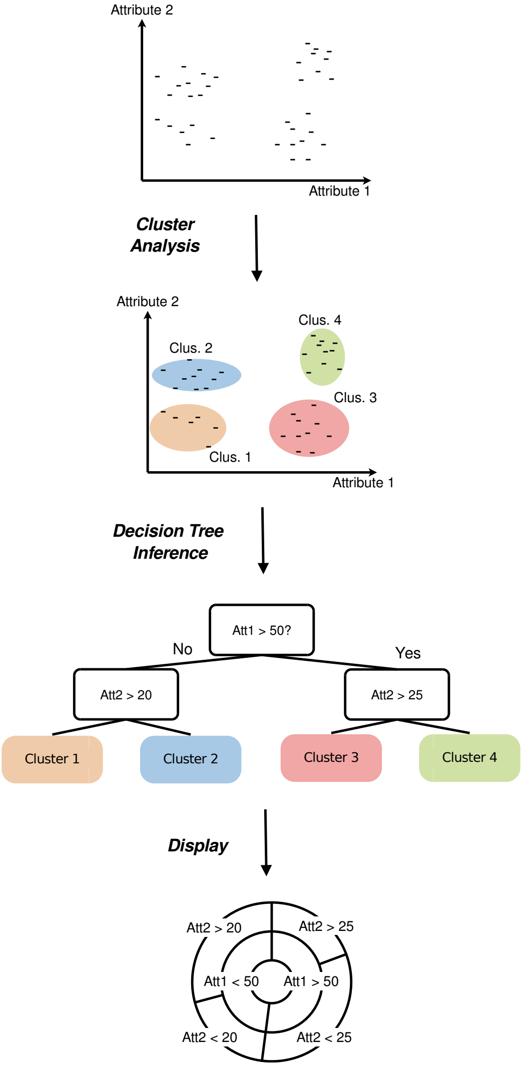


Figure 6.6: The SimpleMap algorithm combines cluster analysis and decision tree inference.

```
function SimpleMap($DB$, $MAX_c$, $MAX_t$)
    $DB' \leftarrow$ preprocess($DB$)
    $labels \leftarrow$ PAM($DB', MAX_c$)
    $tree \leftarrow$ CART($DB, labels, MAX_t$)
    return $tree$
end function
```

Figure 6.7: Detail of SimpleMap for mixed datasets. SimpleMap relies on a preprocessor (preprocess), an unsupervised learning algorithm (PAM), and a tree inference algorithm (CART).

of clusters, Blaeu should return a hierarchy of clusters rather than a flat partitioning. When the clusters are organized in nested sets, users can see several levels of resolution simultaneously. They can experiment with different settings without running the algorithm several times.

The idea behind SimpleMap is to separate *cluster detection* from *cluster description*. We pipeline two algorithms: one to detect the clusters, and another one to generate simpler, hierarchical descriptions. For both steps, we exploit existing work from the data mining literature. First, we cluster the data with a well established algorithm, such as k-means or PAM (Partitioning Around Medoids) [46], described in Chapter 3. Then we build a classification tree, using the cluster assignments as class labels. We illustrate the procedure with Figure 6.6.

Thanks to SimpleMap, we can use sophisticated clustering methods without sacrificing interpretability of the results. Another advantage is that we can run the clustering step and the decision tree step on two different versions of the data. This is useful when the data contains categorical variables. During the cluster analysis, we transform these variables into dummy binary variables, such that each binary variable represents one category. Then, we build the decision tree on the original dataset. The cost of SimpleMap is that of chaining two heuristic algorithms: each procedure induces a latency and some inaccuracies.

We present the detail of Simple Map in Figure 6.7. In our implementation, we use k-means if the data contains only numerical data, and PAM otherwise. PAM is a k-medoid algorithm: for each cluster, it seeks to minimize the distance between all the data points and a central, representative point called medoid. PAM can use any metric, therefore it can cope with a wide range of input data [46]. For the decision tree, we used CART (Classification and Regression Trees). CART operates by splitting the database recursively, minimizing an impurity criterion. This method is well established, it copes with both numerical and categorical data [16]. We present it in more details in Chapter 3.

Our algorithm relies on two parameters: the initial number of clusters to generate MAX_c, and the maximum number of leaves in the decision tree MAX_t. The first parameter can take any value, as long as we generate more clusters than leaves in the tree ($MAX_c > MAX_t$). A higher value leads to more precision, but also longer runtimes. By default, we set $MAX_c = 2MAX_t$. For the second parameter MAX_t, we generate as many leaves as the users' screen can display (by default, 8). If the users wish to work with less partitions, they can use intermediate nodes from the tree.

The total time complexity of SimpleMap depends on the choice of underlying algorithms. Let N represent the number of tuples in the database, and M the number of columns. The complexity of k-means is $\mathcal{O}(MNMAX_c)$. The complexity of the original PAM algorithm is quadratic in N, but we used CLARA [46], a randomized variant which also runs in $\mathcal{O}(NMMAX_c)$ (described in Chapter 3). The CART algorithm runs in $\mathcal{O}(MNMAX_t)$. Therefore, the total time complexity of SimpleMap is $\mathcal{O}\big(MN(MAX_c + MAX_t)\big)$.

4 Algorithm 2: Mapping High-Dimension Data

Previously, we presented a first mapping algorithm, SimpleMap. In this section, we extend our model to support datasets with many columns. We show that such large tables cannot be summarized with one map. We introduce *Multimap*, an algorithm to generate *sets* of maps, in which each map uses a different subset of columns.

4.1 Problem Formulation

Mapping large tables is challenging for two reasons. First, we must deal with the curse of dimensionality. In high dimensional spaces, all items tend to be equidistant. Therefore, clustering does not make sense. Experimental studies showed that problems start to appear with as little as 10-15 columns [13]. Second, we must consider the diversity of the data. Nothing stops a user from using completely unrelated variables. Clustering a survey on `eye color`, `job` and `tupleID` is irrelevant. Yet, this could very well happen in an exploration scenario. Our solution is to use a multi-view approach: we generate several maps instead of a large one. We operate in two steps. First, we partition the data vertically. Then, we create one map for each partition.

The aim of the vertical partitioning is to detect groups of columns which describe the same aspect of the data. To form the groups, we use statistical dependency. If two variables are perfectly independent, there is little chance that they are related in real life. Oppositely, a perfect correlation means that they measure the same property. To partition the data, we identify sets of columns which are mutually dependent.

A flexible and robust way to quantify the dependency between two variables is the Variation of Information (VI) [63]. As explained in Chapter 4, the VI measures the distance between two variables. It has a low value if the variables are similar, and a high value if the variables are independent. We compute it as follows: if $\mathcal{X}$ and $\mathcal{Y}$ are two variables, H denotes the entropy and I the mutual information, $VI(\mathcal{X}, \mathcal{Y}) = H(\mathcal{X}) + H(\mathcal{Y}) - 2I(\mathcal{X}, \mathcal{Y})$. As the VI can only cope with two variables, we introduce the *diameter*. The diameter of a set of variables is the largest VI observed among every pair:

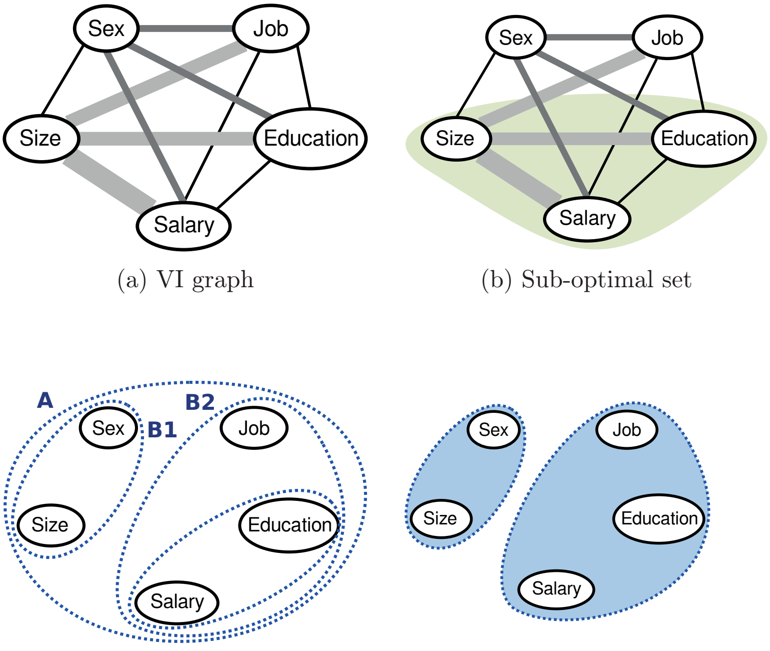
Definition 6.5. *For a set of variables $V = \{X_1, \ldots, X_D\}$, we define the* diameter *as follows:*

$$diameter(V) = \begin{cases} \max\limits_{X_i, X_j \in V} \hat{VI}(X_i, X_j) \text{ if } D \geq 2 \\ 0 \text{ otherwise} \end{cases}$$

Observe that $\hat{\mathrm{VI}}(X_i, X_j)$ is an estimator of $\mathrm{VI}(\mathcal{X}_i, \mathcal{X}_j)$. We discuss the difference between those two quantities in Chapter 4.

To make coherent maps, we need to identify sets of variables with low diameters. We represent this problem with a graph in Figure 6.8a. In this weighted, undirected graph, the vertices represent the variables, and the edges represent the distances. We want to find partitions inside which the heaviest edge is as light as possible.

Unfortunately, the diameter alone cannot tell us which partitions are the best. If we create one partition per variable, we obtain low diameters but the maps are not satisfying. We want a few sets with many columns but minimal diameters. *We must strike a balance between dependency and cardinality.* Formally, we can express this statement as a multi-objective optimization problem. Let the set V_{DB} contain all the variables of the database, and the set of sets $\mathcal{V}$ describe a partitioning of V_{DB}. We want to solve the following system:



(a) VI graph

(b) Sub-optimal set

(c) Comprehensive sets

(d) Max-simple sets

Figure 6.8: The VI graph describes the dependencies between the variables. The vertices represent the variables, the thickness of the edges depicts their weight, i.e., VI.

$$
\begin{aligned}
&\text{minimize} && \sum_{V \in \mathcal{V}} \mathit{diameter}(V) \\
&\text{minimize} && |\mathcal{V}| \\
&\text{subject to} && \bigcup_{V \in \mathcal{V}} V = V_{DB} \\
& && V \cap V' = \emptyset \text{ for every distinct } V,\, V' \text{ in } \mathcal{V}
\end{aligned}
$$

These two objectives are conflicting: less partitions lead to higher diameters, and lower diameters lead to more partitions.

4.2 Enumerating Candidates

We defined variable grouping as a multi-objective optimization problem: we want a large groups of strongly related variables. As our objectives

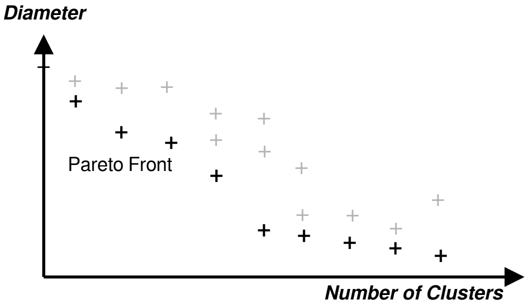


Figure 6.9: The Pareto front contains the set of acceptable partitionings.

are conflicting, there is no unique partitioning which satisfies them both. However, there exists a set of solutions for which we cannot improve one objective without degrading the other, the *Pareto front* [33]. The solutions in this set are called Pareto-efficient, we depict them in Figure 6.9. In this section, we discuss how to compute this Pareto front. We show that the problem can be solved by finding cliques in the VI graph.

First, we introduce a useful property. A set of variables is *comprehensive* if we cannot add any variable without increasing its diameter. To illustrate this notion, we use a counter-example. Consider the partition highlighted in Figure 6.8b. We could add the vertex `Sex` without degrading its diameter, because no edge from `Sex` is heavier than `Size-Salary`. Therefore, the set is not comprehensive.

Definition 6.6. *Let V_{DB} describe the columns of the database and $V_C \subset V_{DB}$ a set of columns. The set V_C is* comprehensive *iff.*

$$\forall X_i \in V_{DB} \setminus V_C, diameter(V_C \cup \{X_i\}) > diameter(V_C)$$

The relation between comprehensive sets and our optimization problem is straightforward.

Lemma 6.3. *The set of partitions $\mathcal{V} = \{V_1, \dots, V_K\}$ is Pareto-efficient if and only if its elements $V_1, \dots, V_K$ are comprehensive.*

Proof. Suppose that we wish to decrease the total diameter. To achieve this, we must split a partition, which increases the count. Suppose we wish to decrease the count. To achieve this, we must merge partitions. If the sets are comprehensive, this will increase the sum of diameters. Thus, we cannot ameliorate one objective without degrading the other, the set $\mathcal{V}$ is Pareto-efficient. □

Figure 6.8c shows the comprehensive sets in the example graph, and Figure 6.8d gives an example of Pareto-efficient partitioning based on this set. Note that any other partitioning based on the comprehensive sets would be valid. We postpone the discussion about the final choice to the end of this section.

How do we detect comprehensive sets? In fact, there is a tight connection between this problem and *maximal cliques*. Recall that a clique is a tight subset of vertices. In a clique, every pair of vertices is connected with an edge. The clique is maximal if there is no larger clique which contains it. To understand the relationship between comprehensive sets and cliques, we introduce the threshold function t_σ. This function takes a weighted graph as input, and eliminates all the edges with a weight larger than σ.

Lemma 6.4. *Consider a weighted, undirected graph $G = (V, E)$. A set of vertices $U \subset V$ is comprehensive iff. there is a σ such that U is a maximal clique in $t_\sigma(G)$.*

Proof. Let U describe a comprehensive set of vertices with diameter p. Consider the graph $t_p(G)$. The edges between the vertices of U weigh at most p. Therefore U is a clique in $t_p(G)$. Also, one cannot add a vertex of $t_p(G) \setminus U$ without degrading the diameter. Thus, U is a maximal clique. Oppositely, consider a graph $t_\sigma(G)$, and the maximal clique W. Every edge in $t_\sigma(G)$ weighs at most σ. Also, for every vertex in $t_\sigma(G) \setminus W$, there is an edge to W which weighs more than σ. W's diameter is at most σ, and it is comprehensive. □

Thanks to this connection, we can obtain the complexity of our problem.

Lemma 6.5. *Enumerating all the comprehensive sets of a weighted undirected graph is NP-hard.*

Proof. Enumerating maximal cliques in a non-weighted, undirected graph is NP-hard [46]. We can reduce this problem to enumerating comprehensive sets. Therefore, enumerating comprehensive sets is NP-hard. To perform the reduction, we map a non weighted graph H to a weighted graph G. For any pair of vertices, we create an edge of weight 0 in G if there is an edge in H. Otherwise, we create an edge with weight 1. If we know the comprehensive sets of G, we can infer the maximal cliques of H in polynomial time. □

```
function MultiMap($DB$, $MAX_c$, $MAX_t$, $MAX_p$)
    $columns \leftarrow$ extractColumns($DB$)
    $colGroups \leftarrow$ comprehensiveSets($columns, MAX_p$)
    $atlas \leftarrow \{\}$
    for $colGroup \in colGroups$ do
        $db \leftarrow pasteColumns(colGroup)$
        $map \leftarrow$ SimpleMap($db, MAX_c, MAX_t$)
        $atlas \leftarrow atlas \cup \{map\}$
    end for
    return $atlas$
end function
```

Figure 6.10: Detail of MultiMap. The function extractColumns takes a table as input and decomposes it into a set of arrays, where each array represents a column. The function pasteColumn does the opposite: it forms a table from a set of arrays.

Enumerating comprehensive sets is hard. Our solution is to relax our requirements. Consider a set of vertices U in a graph G. If we can find a threshold graph $t_\sigma(G)$ in which U is a clique (not necessarily maximal), then U is "good enough". Finding cliques in threshold graphs is precisely what the Complete Link algorithm does [46]. As detailed in Chapter 3, the Complete Link algorithm is a hierarchical clustering algorithm. It operates as follows. First, it creates one partition for each vertex. Then, it finds the two "closest" partitions, and it merges them. The distance between two partitions is the diameter of their union. The procedure is repeated until all the partitions are merged.

The output of the Complete Link algorithm is a tree of nested partitions called dendrogram. Figure 6.11 displays the dendrogram of our example. A node is drawn at height σ if the partitions it represents forms a clique in the graph $t_\sigma(G)$. To obtain near-Pareto efficient partitionings, we "cut" the tree: we let the user choose a maximum number of partitions MAX_p, and keep the solution which is just under this threshold. The threshold encodes the user's preference for one objective function over the other: a low MAX_p will lead to large partitions, a high value will yield many tight sets. In practice, setting this parameter can be difficult for novice users. We help them with two mechanisms. First, we present the dendrogram explicitly in the interface. This provides visual feedback. Second, we propose default values. Several dendrogram-cutting heuristics were introduced in the clustering literature, such as the Silhouette method [46] or Dynamic-

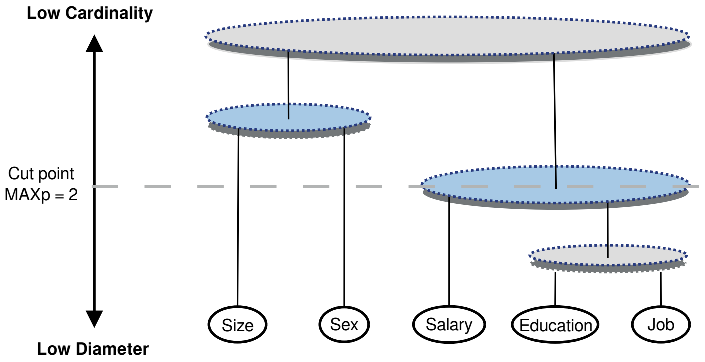


Figure 6.11: Dendrogram produced by the complete link algorithm. A node at height σ represents a clique in the graph $t_\sigma(G)$.

TreeCut [53]. Although they provide no accuracy guarantee, they form excellent starting points for exploration. We used DynamicTreeCut in our implementation.

4.3 Summary and Complexity

Previously, we presented the algorithm SimpleMap. We can now enrich it to deal with high dimensional spaces. Here is our new framework, MultiMap:

1. Partition the columns of the context into comprehensive sets
2. Create one map for each partition with SimpleMap

In essence we process the columns first, then the rows: we decouple subspace search and clustering [67]. Figure 6.10 details the algorithm. Recall that N and M describe the number of rows and columns respectively. The complexity of the column partitioning is $\mathcal{O}(NM^2)$, because we need to compute the dependency between every couple of columns and run the Complete Link algorithm. If the procedure generates P partitions, the second step will run in $\mathcal{O}\big(PNM(MAX_c + MAX_t)/P\big) = \mathcal{O}\big(NM(MAX_c + MAX_t)\big)$, using the results from Section 3. Thus the total time complexity of MultiMap is $\mathcal{O}\big(NM^2 + NM(MAX_c + MAX_t)\big) \approx \mathcal{O}(NM^2)$. The quadratic term in M indicates a potential bottleneck. To bypass it, we compute the VI graph offline, and reuse the vertical partitions across zooms (cf. recycling, discussed in Section 6). Also, our experiments nuance the

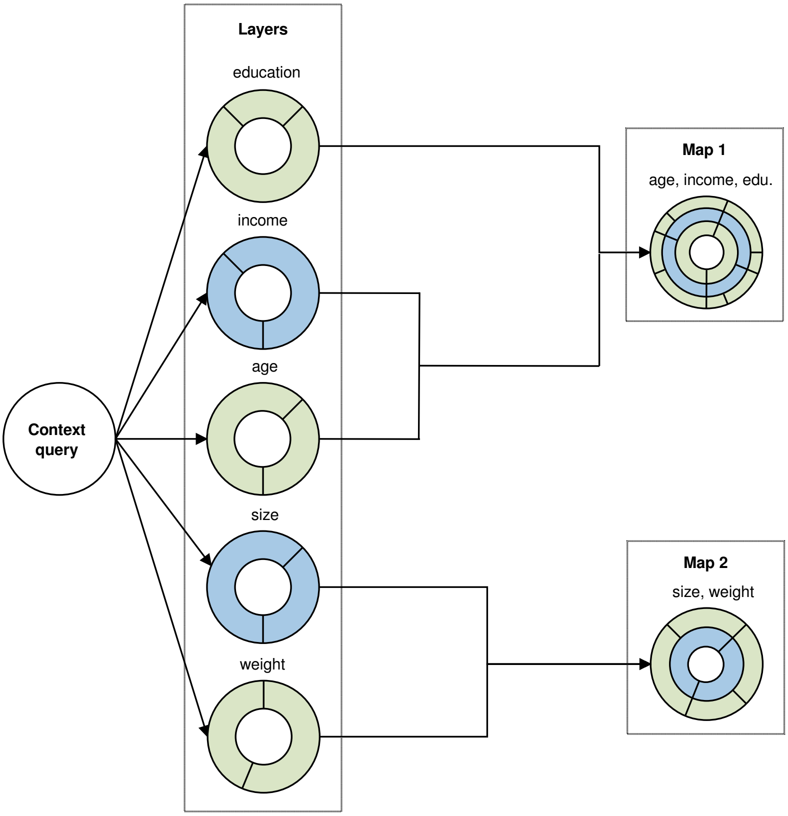
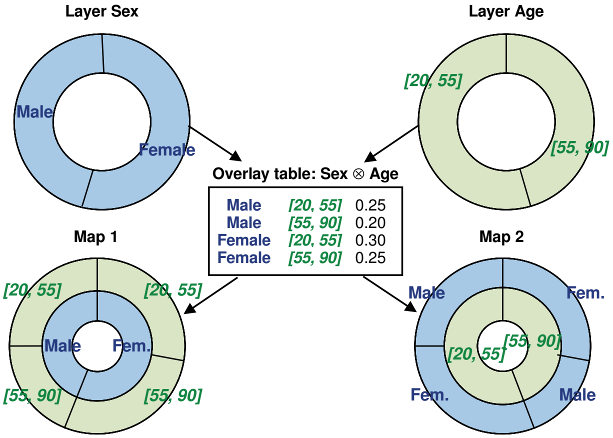


Figure 6.12: Overview of LightMap

complexity analysis: as long the the data does not contain more than several hundred columns, the execution time is completely dominated by the map creation step, linear in N and M (as shown in Section 10).

5 Algorithm 3: Lightweight Data Mapping

We now present our third algorithm *LightMap*. LightMap is more flexible than its predecessors: the maps it generates can be modified by the users at little cost. Also, it is very fast. We will see that this comes at a price: the algorithm is less accurate than MultiMap. LightMap operates in three phases. First it creates *layers*. A layer is a map, based on one column only. LightMap creates one layer for each column of the database. Then, it forms groups of similar layers, e.g., layers which describe the same aspect



Overlay table: Sex ⊗ Age

Male	*[20, 55]*	0.25
Male	*[55, 90]*	0.20
Female	*[20, 55]*	0.30
Female	*[55, 90]*	0.25

Figure 6.13: Two layers, their combination

of the data. Finally, it combines the layers of each group into larger maps. This process is illustrated in Figure 6.12.

The first two steps of LightMap use the primitives we presented earlier. We create the layers with SimpleMap. To group similar layers, we exploit the notion of comprehensive sets discussed in Section 4. However, this time we cluster maps, not columns - in fact, we form clusters of clusterings. During the last step, LightMap forms complex maps by combining layers. To combine two layers, LightMap intersects every partition of one map with the partitions of the other. We call the resulting structure *overlay table*. From this table, LightMap can generate hierarchical partitionings for any permutation of the variables. Consider for instance Figure 6.13. We present two layers, one based on `Sex`, the other on `Age`. The figure presents the overlay table in the center. This table lets us compute two maps: one based on `Sex` then `Age`, the other on `Age` then `Sex`. If we keep the overlay table in memory, the users can switch from one representation to the other at little cost, in order to chose the variable ordering which suits them most. By default, we order the layers by clustering quality (obtained during the layer creation phase).

We now summarize the full LightMap procedure:

1. Create one layer for each column in the context

2. Create comprehensive groups of *layers*

3. Overlay the layers in each group

```
function LightMap($DB$, $MAX_c$, $MAX_t$, $MAX_p$)
    /* Generate Layers */
    $columns \leftarrow$ extractColumns($DB$)
    $layers \leftarrow \{\}$
    for $column \in columns$ do
        $lay \leftarrow$ SimpleMap($column, MAX_c, MAX_t$)
        $layers \leftarrow layers \cup \{lay\}$
    end for
    /* Form groups of layers */
    $layerGroups \leftarrow$ comprehensiveSets($layers, MAX_p$)
    /* Aggregate the layers of each group */
    $atlas \leftarrow \{\}$
    for $layerGroup \in layerGroups$ do
        $oTable \leftarrow$ overlayTable($layerGroup$)
        $map \leftarrow$ makeMapFromTable($oTable, MAX_t$)
        $atlas \leftarrow atlas \cup \{map\}$
    end for
    return $atlas$
end function
```

Figure 6.14: Detail of LightMap. The function extractColumns splits a table into a set of arrays, one for each column. The function pasteColumn forms a table from a set of arrays. The function makeMapFromTable generates a map from an overlay table, with at most MAX_t leaves (if necessary, it truncates the overlay table).

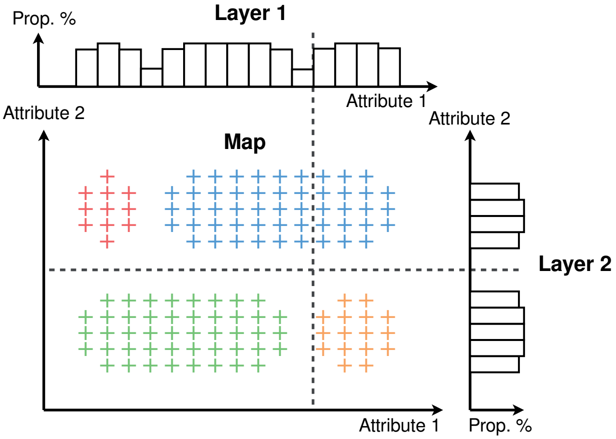


Figure 6.15: Limit case of the LightMap algorithm. The data contains four clusters, which full two-dimension structure is lost in the projections.

The detail of the algorithm is given in Figure 6.14. Note that LightMap's partitions are not real clusters. They approximate what the full MultiMap algorithm would have found. We suppose that this approximation is good enough for exploration. Formally, we can justify our scheme with the downward closure property of density. If an area has a high density in D dimensions, its projections on any $D - 1$ dimensional subspace is dense. In other words, the clusters do not “disappear” when they are projected to single dimensional spaces, and therefore they appear on the layers [51]. However, the projection incurs a loss of information, because the clusters may be stacked onto each other. Therefore, LightMap can incur mispredictions, as shown in Figure 6.15. This accuracy penalty is the counterpart for significant speedups.

The respective time complexities of map creation, layer grouping and layer combination are $\mathcal{O}\big(NM(MAX_c + MAX_t)\big)$, $\mathcal{O}(NM^2)$ and $\mathcal{O}(NM)$, using the results from Sections 3 and 4. Here again, we bypass the layer grouping step by recycling (presented in Section 6). Furthermore, our experiments nuance the complexity analysis: in our implementation, the first phase totally dominates the total runtime, even with several hundred columns (cf. Section 8.2).

6 Optimization

We improve the latency of Blaeu with two optimizations. First, *we recycle*. We compute the VI graph offline. During the exploration, we do not recluster the variables after each zoom. We run the full procedure for the first query, and reuse the comprehensive sets until the user changes the variables in the context.

Second, *we sample*. For each user action, we only take a few tuples from the context. As we have no preliminary knowledge about the data and we wish to estimate the relative proportions of each group, we use random sampling with replacement. To pick an optimal sample size, we run a *calibration phase* when Blaeu starts up. During this phase, we create synthetic data sets with different sizes, run Blaeu and measure the time spent. From these observations, we generate a cost model. Thanks to this model, we can pick a maximum sample size given a time objective and a number of columns.

To build our cost model, we fit a multiplicative model on the observed runtime. If *#rows* represents the number of tuples, and *#columns* the number of columns, our model can be expressed as follows:

$$runtime \approx \alpha \times \#rows^{\beta} \times \#columns^{\gamma} \tag{6.1}$$

If we compute the logarithm of each operand, the model becomes linear: $log(runtime) \approx log(\alpha) + \beta.log(\#rows) + \gamma.log(\#columns)$. Thanks to this transformation, we can obtain the coefficients with linear regression. Figure 6.16 pictures an example of calibration session. We used the same hardware as in our Experiments section. We observe that the model fits the observed values almost perfectly.

7 Sample Sessions

We now illustrate the system with two "real-life" scenarios. All the datasets are available on our website[1].

First, we use a sample of the On-Time database, provided by the US Bureau of Transportation Statistics. The dataset describes delays of US internal flights during January 2010. It contains about 521,000 rows and 91 columns.

[1] http://homepages.cwi.nl/~sellam/blaeu.html

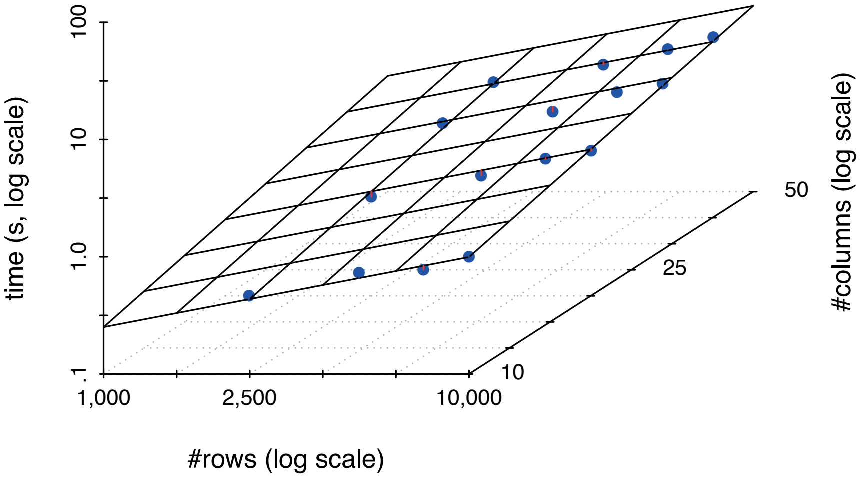


Figure 6.16: To calibrate our system, we run Blaeu on synthetic data, measure the runtime and fit a multiplicative model. The points represent the observations, the plane our fitted model. We obtain a correlation coefficient of 0.98, which indicates an almost perfect correlation.

Our users want to understand the causes of flight delays. For now, we assume that they know where to begin: they seed the exploration with the columns `Distance` and `ArrDelay`. The first variable `Distance` refers to the distance covered by the flight (in Miles). `ArrDelay` is the delay of the flight (in minutes). We reproduce Blaeu's map in the top-left corner of the figure. Four categories of flights appear: short flights (up to 1167 miles), or longer ones, with or without long delays. Our users decide to zoom in the short delays on short flights, and project to highlight the causes. About a quarter of delays come from the carrier. Another quarter is caused by a late aircraft. They rollback to the first map and zoom into the long delays on short flights. A slightly higher proportion of delays come from weather conditions. Also, the carriers are responsible for more than a quarter of the late flights. They zoom in the longest carrier delays, and project on the name of the companies. They observe that roughly half the delays come from only five carriers. Do they have more delays because they operate more flights? Or should they be suspicious next time we book a flight?

Our second example describes a somewhat more glamorous domain: Hollywood films. Our database describes a few economic indicators for 785 movies released between 2007 and 2012. Our users' focus is abstract and subjective: they are looking for disappointing movies, also called "flops". This time, they do not even know where to begin. Therefore, they run Blaeu on the whole database. Blaeu replies with a set of maps, listed in the top part of Figure 6.18. Two maps seem like good candidates: the first one, which shows the number of theater for each movie, and the third one, which describes the public and critic's response. Our users pick the first map, and zoom on the movies shown in many cinemas. Then, they switch map, and select those for which the critiques were bad. After three clicks, they already have a list of candidates. With a few more, they could dig further: did these movies generate any income? Did they fail similarly everywhere? Were they expensive? We see Blaeu lets our users "dive" in the data with only a few clicks.

8 Validation and Evaluation

Blaeu's suggestions only make sense if they reflect the structure of the data, and it they are computed at interaction time. In this section we evaluate the runtime and accuracy of our algorithms. In what follows, we will abbreviate SimpleMap, MultiMap and LightMap as S-Map, M-Map and L-Map respectively.

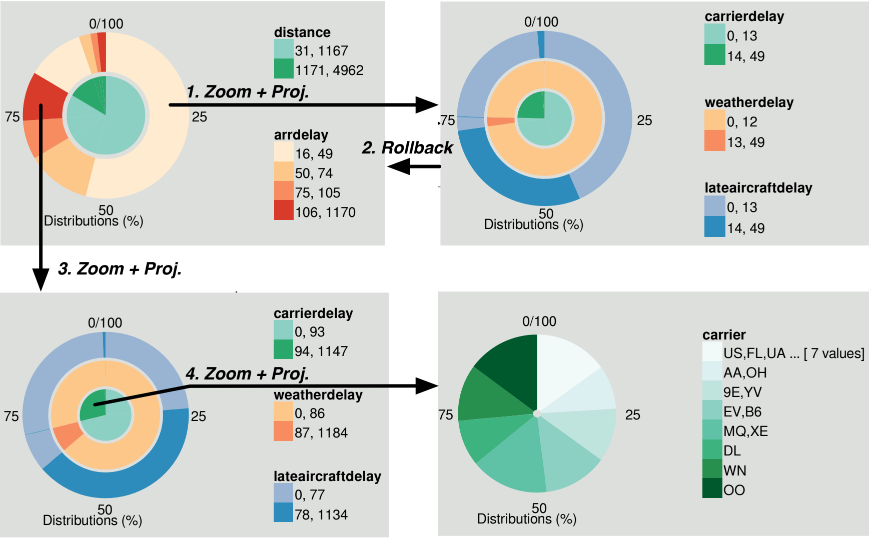
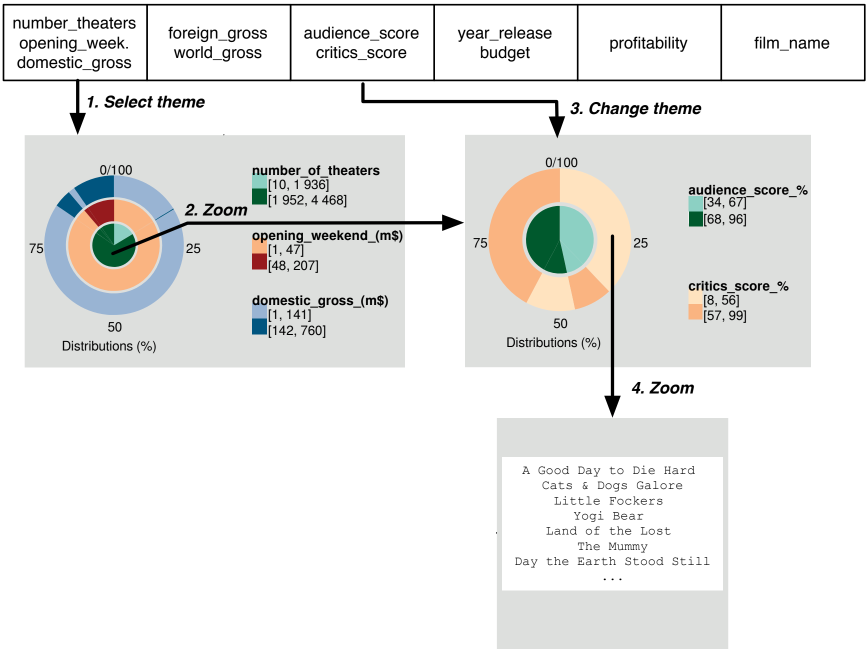


Figure 6.17: Running Blaeu on the Ontime database



number_theaters opening_week. domestic_gross	foreign_gross world_gross	audience_score critics_score	year_release budget	profitability	film_name

A Good Day to Die Hard
Cats & Dogs Galore
Little Fockers
Yogi Bear
Land of the Lost
The Mummy
Day the Earth Stood Still
...

Figure 6.18: Running Blaeu on the Hollywood database

We confront Blaeu to state-of-the-art algorithms from the clustering literature. We used two families of algorithms: *subspace search* and *subspace clustering.* Methods from the first family return only subspaces, the user is responsible for the actual clustering. Typically, they seek "high contrast" subspaces, that is, subspaces with clusters and outliers. To do so, they enumerate different combinations of variables with level-wise search, and return the top k best candidates. Algorithms from the second family seek clusters and subspaces simultaneously. For each cluster, they return a set of tuples and a set of variables.

To represent the subspace search family, we used CMI [68]. CMI is theoretically well founded, and it resembles Blaeu: like DM, CMI maximizes a mutual information criterion (the Cumulative Mutual Information). To get the actual clusters, we use SimpleMap. For the subspace clustering family, we chose PROCLUS [2] and FIRES [51]. These algorithms have been shown to be fast and accurate [66]. Also, they are close to our approach: like LightMap, FIRES combines one-dimension clusterings.

Contrary to our system, CMI, PROCLUS and FIRES target data miners. They value exhaustivity, while we seek simplicity and speed. Consequently, they rely on a broad range of parameters (up to 9 for FIRES), while we need just one (the maximum tree size). Also, their search space is larger. With these algorithms, a variable can appear in several subspaces, or not at all. With Blaeu, each variable appears exactly once in the whole result set. Our approach may miss some interesting subspaces, but it has two advantages. First, it completely eliminates redundancy in the result set (Blaeu's competitors often return dozens, sometimes hundreds of highly redundant subspaces on small datasets). Second, it reduces latency.

Our code, settings and datasets are available online[2]. We used Open Subspace's implementation of PROCLUS and FIRES [66] (written in Java). For CMI, we used the author's Java implementation[3]. We implemented S-Map, M-Map and L-Map in R 3.0.1, but some of the critical parts are either native C primitives or our own C function. We use three functions from the R repository: `CLARA` for the cluster analysis, `DynamicTreeCut` to chose a number of subspaces and `rpart` for detection tree inference. The function `CLARA` is a randomized variant of PAM, and `rpart` is an implementation of CART. Our test system is based on a 3.40 GHz Intel(R) Core(TM) i7-2600 processor. It is equipped with 16 GB RAM, but the Java heap space is limited to 12 GB. The operating system is Fedora 16. Unless written otherwise, we disable sampling.

[2] `http://homepages.cwi.nl/~sellam/blaeu.html`

[3] `http://www.ipd.kit.edu/~muellere/CMI/`

Table 6.1: Parameters of the Data Generator

Parameter	**Distribution**	**Value**
Tuples	Constant	20,000
Columns	Constant	40
Columns per subspace	Uniform	[2,10]
Clusters per subspace	Constant	5
Centroid position	Uniform	[1,100]
Cluster radius	Uniform	[2,10]

Table 6.2: Characteristics of the datasets

Name	Rows	Col.
breast	198	33
diabetes	768	8
communities	1,994	127
internet	7,390	70
pendigits	7,494	16
adult	32,561	14
covertype	581,012	54
gisette	7,000	2,500
mutant1	16,592	4,000

8.1 Synthetic Data

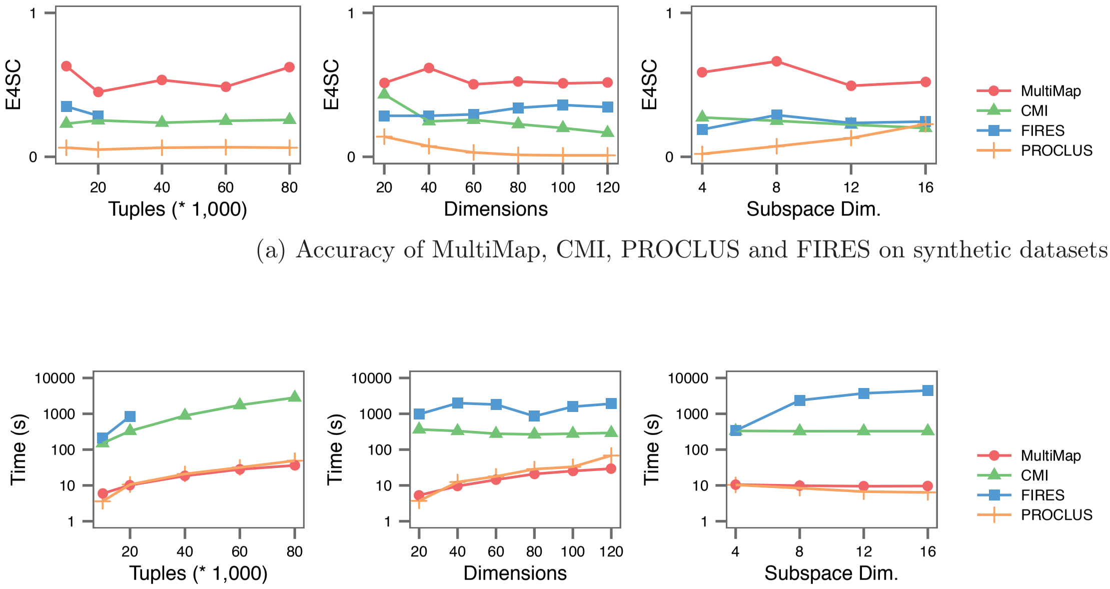
In this section, we report the results of experiments on synthetic data. We create datasets for which we know the “truth”, and evaluate Blaeu’s output. We generate columns by groups, such that *the data is clustered differently on each group of columns.* Therefore, each dataset contains several clustered subspaces. Note that the groups are not overlapping. The clusters are based on multinomial Gaussian distributions, with random parameters. The default options of the generator are reported in Table 6.1. The subspaces are isolated, and the clusters are well separated: this is an “easy” scenario for Blaeu, but also for its competitors.

We made significant efforts to tune FIRES, PROCLUS and CMI correctly. As the data is synthetic, we can calculate the “ideal” parameters. We report them on our website. Also, recall that S-Map, M-Map and L-Map are hierarchical. For a meaningful comparison, we altered them so that they return a fixed number of clusters. For each setup, we run each algorithm with five different datasets, and report the average performance. For practical reasons, we interrupt the experiments which take more more than 10,000 seconds (2 hours and 46 minutes).

As in the recent subspace clustering literature, we measure clustering quality with a variant of the F1 measure called E4SC [66][35]. The traditional F1 score focuses on tuples only: two clusters are similar if they contain the same data points. With the E4SC, two clusters are similar if they contain the same tuples and they appear on the same dimensions. Therefore, the E4SC considers both the clusters and the subspaces in which they were found. The measure varies between 0 and 1; a high value indicates that the found clusters are similar to the true clusters, a low value shows discrepancies.

Figure 6.19a compares MultiMap to CMI, PROCLUS and FIRES. We observe that MultiMap is accurate. The results are stable with regards to the number of tuples and columns. In most cases, FIRES comes second, closely followed by CMI, and PROCLUS comes last. Figure 6.19b shows the runtime of the algorithms. MultiMap and PROCLUS are orders of magnitudes faster than CMI and FIRES. M-Map is almost always faster that PROCLUS. Our approach is validated.

We compare S-Map, M-Map and L-Map in Figure 6.20. As SmallMap cannot detect subspaces, it generates poor E4SC scores. Its performance is particularly low when the subspaces have a low dimensionality, and it gets better when the data contains a few wide subspaces. MultiMap is the most accurate algorithm, but it is also the slowest. Recall that MultiMap is based on a combination of cluster analysis an decision tree classification.



(a) Accuracy of MultiMap, CMI, PROCLUS and FIRES on synthetic datasets.

(b) Execution time of MultiMap, CMI, PROCLUS and FIRES on synthetic datasets.

Figure 6.19: Comparison of Blaeu and subspace clustering on synthetic data.

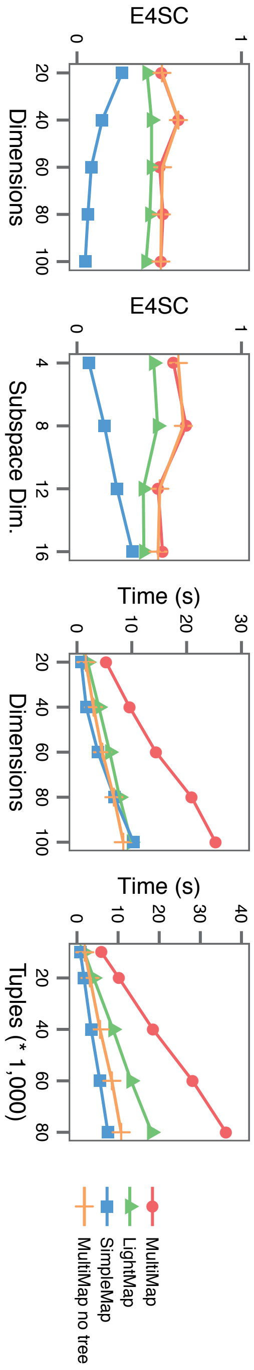
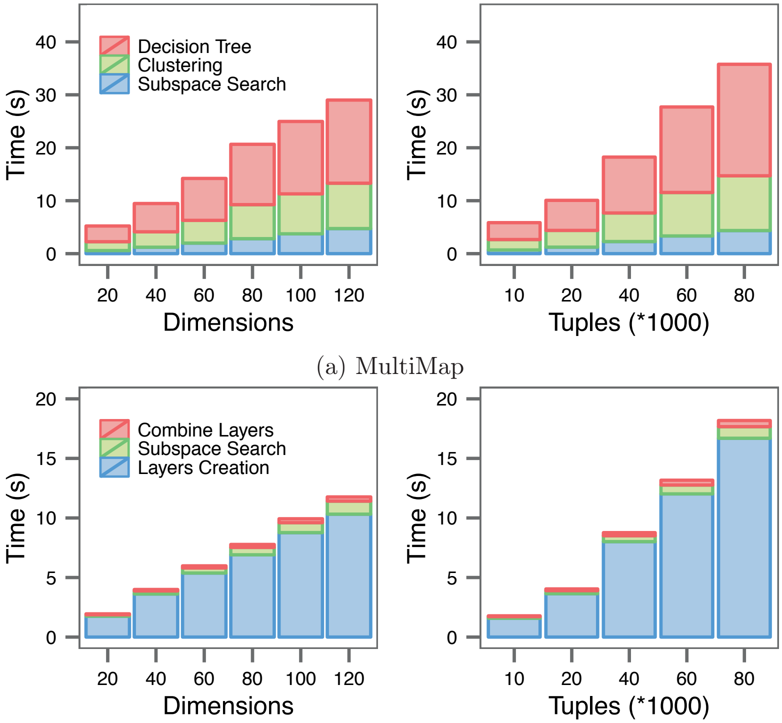


Figure 6.20: Comparison of SimpleMap, MultiMap, and LightMap on synthetic datasets.



(a) MultiMap

(b) LightMap

Figure 6.21: Execution time breakdown of MultiMap and LightMap.

To separate the errors of both components, we introduced a variant in which we removed the decision tree. We observe very little difference in E4SC, which indicates that the accuracy penalty incurred by the decision tree is acceptable. However, the runtime increases more than twofold. Finally, LightMap is at least twice fast as MultiMap, but this comes at the price of inaccuracies.

Figure 6.21 details the time consumption during the execution of M-Map and L-Map. In MultiMap, more than half the time is spent building the decision tree. Blaeu spends comparatively very little time clustering the rows and the columns. Fortunately, this step can easily be accelerated by sampling (cf. 8.3). In LightMap, almost all the time is spent creating layers. The subsequent steps are very fast, except when the data contains many columns.

8.2 Real Data

We showed that S-Map, M-Map and L-Map perform well on synthetic data sets. In this section, we run them on nine databases from the UCI repository [10], described in Table 6.2. As we do not have any base truth, we use a “trick” from the subspace clustering literature: we exploit the class labels originally designed for classification and regression, in the hope that they reflect the structure of the data [66]. We report the F1 between these labels and our predictions. The F1 varies between 0 and 1, higher is better. To tune the algorithms, we generate different sets of parameters and report the best results. For instance, we try 50 combinations for FIRES and 56 for CMI. The details of the experiments are published on our website. We do not enforce any time limit, but we discard the algorithms which saturate the 12GB heap space limit. Table 6.3 reports the accuracy of the algorithms. The highest and lowest values are highlighted. There is no clear winner: each algorithm can perform very well with a dataset, and underperform with an other. For instance, FIRES performs very well with `Communities`, but comes last for `PenDigits`. Similarly, SimpleMap excels with `PenDigits` and `Adult`, but fails with `Breast`, `Diabetes` and `Internet`. CMI and MultiMap are more consistent, and they appear to have similar scores. We observe that MultiMap does not always outperform SimpleMap: when the data contains few columns (less than 16 in our case), MultiMap’s vertical partitioning strategy may be too aggressive. LightMap is slightly over-performed by its competitors. In conclusion, Blaeu does not outperform subspace search and clustering algorithms, but its performance is comparable.

Table 6.4 shows the execution times. Clearly, S-Map and L-Map are the fastest algorithms. They are up to three orders of magnitude faster than CMI and FIRES, and up to five times faster than PROCLUS. In most cases, MultiMap comes third. Also, all the competitors exceed the memory limit for the sets `gisette` and `mutant1`. Remarkably, LightMap overperforms SimpleMap for `mutant1`. We conclude that out algorithms are more efficient than CMI, FIRES and PROCLUS.

8.3 Scaling and Sampling

We now show how sampling helps us deal with very large data sets. For each dataset, we train Blaeu with a small sample, extract the resulting decision tree and apply it to the full dataset. We then compare the result to a base truth: if the labels are similar, then our strategy works. We sampled the data uniformly, with replacement. For each experiment, we

Table 6.3: Accuracy (F1) for SimpleMap, MultiMap, LightMap, CMI, FIRES and PROCLUS with real datasets

	S.Map	M.Map	L.Map	CMI	FIRES	PROC
breast	*0.20*	0.60	0.53	0.69	0.39	0.58
diabetes	*0.12*	0.55	0.40	0.54	0.42	0.37
comm.	0.62	0.60	*0.56*	0.61	0.89	0.57
internet	*0.27*	0.39	0.44	0.40	*	0.38
pendigits	0.91	0.47	0.37	0.36	*0.22*	0.59
adult	0.61	0.59	0.54	0.56	*	*0.52*
covertype	0.48	0.47	*0.44*	0.48	*	0.50
gisette	*0.34*	0.51	0.42	*	*	*
mutant1	*0.43*	0.64	0.51	*	*	*

Table 6.4: Runtimes (seconds) for SimpleMap, MultiMap, LightMap, CMI, FIRES and PROCLUS with real datasets

	S.Map	M.Map	L.Map	CMI	FIRES	PROC
breast	0.18	0.72	0.08	0.87	*1.01*	0.25
diabetes	0.16	0.35	0.05	0.45	*1.98*	0.26
comm.	0.57	4.53	1.22	3.97	*150.8*	1.65
internet	0.43	2.31	0.54	*16.70*	*	3.17
pendig.	0.46	5.69	0.90	40.82	*94.25*	1.00
adult	0.83	6.00	1.32	*683.81*	*	3.31
covert.	17.88	79.77	47.08	*79,069*	*	12,139
gisette	154.4	*688.2*	177.75	*	*	*
mutant1	960.9	*2,155*	829.21	*	*	*

took 5 different samples and averaged the results.

Figure 6.22a shows the results of our experiments with MultiMap on synthetic data. We observe that Blaeu makes considerable progress with the first thousand tuples. Then, the sample size has almost no influence on the accuracy of the results. We conclude that small samples are good enough: Blaeu can very well infer the structure of the data from small subsets of the data. Two factors explain this result. First, Blaeu operates at a very coarse level. It seeks at most a dozen large clusters. Therefore, it is not sensitive to noise and micro-clusters. Second, it works mostly with low dimensional subspaces. Even if the data contains hundreds of columns, a map rarely contains more than a dozen of them. This also explains why the number of variables has such little impact on our results (the top-left and top-right charts are almost similar).

Figure 6.22b presents a similar experiment with real data. We used two public datasets, available through our Website. Here again, we observe an increase of accuracy with the first few hundred samples. Then, the F1 stays almost constant : Blaeu makes little to no progress. However, we do observe that the F1's variability decreases as the sample size increases.

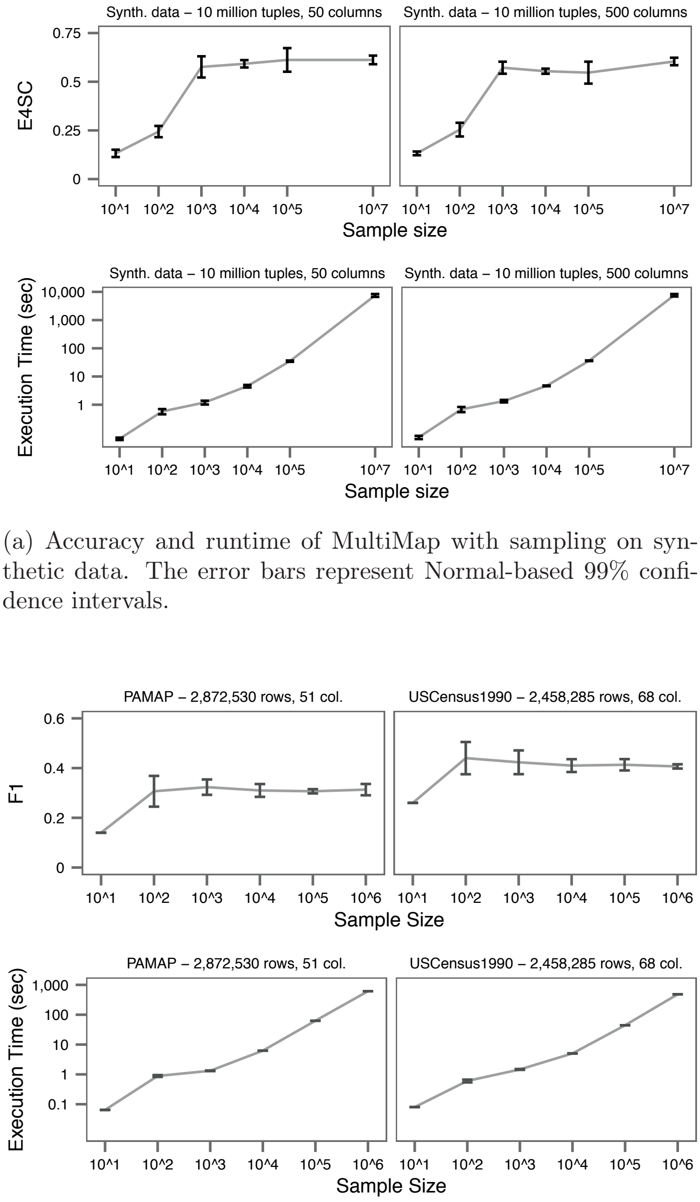
Considering Blaeu's execution times in both experiments, we conclude that Blaeu can produce high precision maps of million-tuples datasets within two seconds. Note however that our measurements do not include the actual sampling: we assume that Blaeu's back-end can produce samples quickly enough (our back-end MonetDB generates samples in less than a second when the data is "hot")

9 Related work

Researchers have recently shown much interest in data exploration. The research effort is multidisciplinary: proposed studies range from core database technology [86] to visualization [71]. A comprehensive overview of the field is provided by Idreos et al. [41].

9.1 OLAP cubes

Drilling and slicing through graphical tools have been common practice for years in the OLAP world. We discuss related techniques in detail in Chapter 5.



(a) Accuracy and runtime of MultiMap with sampling on synthetic data. The error bars represent Normal-based 99% confidence intervals.

(b) Accuracy and runtime of MultiMap with sampling on real data. The error bars represent Normal-based 99% confidence intervals.

Figure 6.22: Experiments with sampling.

9.2 Iterative interaction systems

Our system is the direct descendant of the vision system Charles [85]. Like Blaeu, Charles is based on top-down exploration. However, it does not use any clustering, it only considers partitionings based on medians. Query Steering is related [26], but it models the user rather than the data. Our method is orthogonal, both approaches could be combined. Finally, the vision system SeeDB [71] is close to Blaeu because it choses visualizations automatically. Nevertheless, it focuses on the presentation of the results, not on the user input: it relies on traditional SQL queries.

9.3 Clustering

Our work relies on subspace clustering. An exhaustive review of the field was written by Kriegel et al. [52]. We describe the most relevant approaches in Section 10. All the algorithms mentioned in this chapter are based on axis-parallel subspaces. We chose to discard more general techniques, e.g., based on affine projections or kernel methods [52], because their results are much harder to interpret, especially for non statisticians.

10 Summary

Too often, data exploration tools assume that users know the data and know what they are after. In this chapter , we challenge this assumption. We introduce a new mode of interaction, based on data maps. We formalize the problem of generating maps, study its complexity and present our algorithms. Experiments on real and synthetic data reveal that our framework is fast and accurate.

Chapter 7

Ziggy: What Makes My Query Special?

1 Introduction

In the previous two chapters, we introduced systems to help users write queries. The assumption was that users did not know their data well enough to issue SQL statements. However, if they saw an interesting set of tuples, they could recognize it immediately, possibly endorsing it with a “Aha!” exclamation. But this assumption is simplistic. The usefulness of a set of tuples is rarely obvious. One of the largest obstacle to assessment is the users’ “bandwidth”, i.e., the amount of information that they can grasp without being overwhelmed. A user can analyze a table with a dozen columns, but not one with hundreds of variables. Multidimensional visualization methods can help, but they have their limits too. Consider for instance scatter-plot matrices, presented in Chapter 2. To represent a data set with 50 columns, we would need at least 1,250 different plots. Few users would have the will or the patience to go through all these views. *How can we characterize a set of tuples in medium to high dimension spaces?*

1.1 Contribution

In this chapter, we introduce Ziggy, a subset characterization engine. Ziggy’s aim is to show what makes a given set of tuples “special”. To do so, it detects *characteristic subspaces*, that is, small, non redundant projections which highlight the difference between the selection and the rest of the data. By inspecting those, the users can grasp what makes their tuples unique and assess whether their subset is interesting or not.

We will first present the subset characterization problem in its general form. We will then discuss how to quantify the “peculiarity” of a set of tuples. Next, we will present our system in detail. We will show that Ziggy generates informative views: experiments with statistical classifiers reveal that it captures the distribution of the user's selection accurately. But it is also very fast. Thanks to aggressive optimizations, it can process 10,000s tuples on dozens of variables within a second - an order of magnitude faster than state-of-the-art algorithms, at minimal accuracy costs. Finally, Ziggy is a white box: every choice it makes is fully *explainable*. As a proof of concept, we will present a method to convert its findings into natural language, i.e., plain English. In general, few data mining algorithms provide this functionality, and none of those that do tackle view search.

1.2 Outline

In Sections 2 and 3, we present our general problem. We instantiate this problem in Section 4 and describe our algorithms in Section 5. We discuss how to validate our results and report them in Sections 6 and 7. We describe how to set parameters in Section 8. In Sections 9 and 10, we apply our solution to real and synthetic data. Finally, we present related work and conclude in Sections 11 and 12.

2 Overview

Let us introduce our system through an example. A government analyst tries to understand which demographic, social and economic factors lead to innovation. More specifically, she is interested in patents: in which parts of the world do individuals submit patents? What lessons can she learn? She collects several hundred regional statistics, and loads them in her favorite Business Intelligence tool (possibly based on SQL or spreadsheets). She selects the top 10% regions for which many patents are submitted, and ends up with an overwhelmingly large table, comprising a few dozen tuples and hundreds of columns. How can we help her?

Our idea to detect sets of columns for which the analyst's tuples have an unusual statistical distribution compared to the rest of the database. A fundamental assumption behind our work it that the user is interested in the differences between her selection and the remaining tuples, not the similarities. This is a strong hypothesis, but recent work has shown that it covers a wide range of use cases [97].

Figure 7.1 depicts two of those sets: `{Yearly Production, Average Income}`, and `{Years in Education, Number of Universities}`. We refer to them as *views*, or *subspaces*. We see that on all the variables, the selection has unusually high values. This shows that regions with many

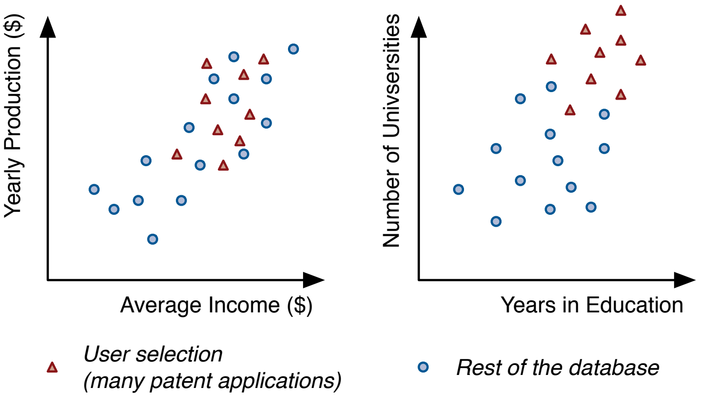


Figure 7.1: Two examples of subspaces for which the user's selection has an unusual distribution (using fictive data).

patent applications tend to be richer and have solid education systems. Observe that these views have low dimensionalities: each of them illustrates one "characteristic" of the selection. In fact, our aim is not to show *all* the variables on which differences appear. Instead, we seek a few small, non redundant subspaces. We call this approach *multi-view characterization*.

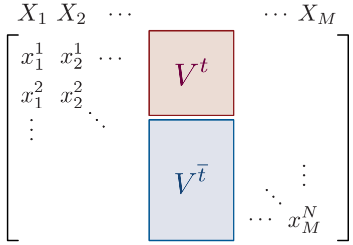
To visualize the subspaces, our analyst may use tables or charts, as in Figure 7.1. She may also wonder *why* the system chose these views. For instance, she may wish to check the sanity of the results, or she may seek guidance. To help her, we provide justifications in natural language. Here is an example:

> "On the columns `Average Income` and `Yearly Production`, your selection has a high average and low variance. The effect is similar and stronger on `Years in Education` and `Number of Universities`."

This paragraph describes the rationale behind our choice. It explains what makes the tuples special, and which characteristics our analyst should check.

3 General Problem Statement

Let us now formalize the multi-view characterization problem. We represent our database with a table containing M columns and N rows. We represent each column by a vector $X_m = (x_m^1, \ldots, x_m^N)^\top$. Let the submatrix $V = [X_1, \ldots, X_D]$ describe a view (i.e., subspace, or set of columns). We split this view in two: V^t contains the tuples in the selection, $V^{\bar{t}}$ the remainder, as shown below:



To measure how much V^t and $V^{\bar{t}}$ differ, we compare the empirical probability distribution of their rows. If these distributions differ, then the view exhibits some "peculiarity" of the data. Therefore, V is likely to be informative. We measure this difference with a **mass dissimilarity measure** $\mathfrak{D}(V^t, V^{\bar{t}})$. This function returns 0 if the tuples come from from the same distribution, it grows otherwise. The statistics literature contains several candidates, for example estimators of the Kullback-Leibler Divergence, already discussed in Chapter 5. We will present our own function in the following section.

We can already propose a first, naive, formulation of our problem:

Problem 7.1. *Consider a distribution dissimilarity function $\mathfrak{D}$ and two integers K and D. Find the top K distinct views V_i with at most D dimensions which maximize $\mathfrak{D}\big(V_i^t; V_i^{\bar{t}}\big)$ (ignoring column permutations).*

This approach is simple, but it yields redundancy: a small number of good columns may dominate the results and appear in all K views. In a data exploration scenario, users may value *diversity*, even if the views are sub-optimal. To enforce this requirement, we introduce a penalty factor in the objective function.

Measuring redundancy between columns is not straightforward, as this notion is partially subjective. In our model, we exploit statistical dependency: if two sets of columns are tightly correlated, then there is a high chance that they describe the same property of the "real world". Oppositely, if they are independent, then they probably convey different types of information. From this observation, we introduce a new version of our problem: we seek views which maximize the dissimilarity measure, while minimizing inter-view dependency. We define this problem in a recursive way:

Problem 7.2. *Suppose that we have already detected $i-1$ views ($i > 1$, $V_0 = \emptyset$). We obtain $V_{1..i-1} = [V_1, \ldots, V_{i-1}]$ by concatenating these views. Let $\mathfrak{S}$ describe a statistical dependency measure. Given a positive real λ, find the view V_i with at most D columns which maximizes:*

$$\mathfrak{D}\big(V_i^t; V_i^{\bar{t}}\big) - \lambda \cdot \mathfrak{S}(V_i; V_{1..i-1}) \tag{7.1}$$

Statistics textbooks offer many options to instantiate the dependency measure $\mathfrak{S}$. Well established examples are multivariate variants of the correlation coefficient, or the mutual information [102]. Here again, we will present our own function in Section 4.2.

In Equation 7.1, the parameter λ controls the trade-off between dissimilarity and diversity: a high value enforces that that the view are diverse, while a low value expresses our preference for maximizing $\mathfrak{D}(V_i^t; V_i^{\bar{t}})$. In practice, this parameter is not convenient because it has no intuitive scale. For some L, an equivalent way to express our problem is the following:

$$\begin{array}{ll} \text{Argmax}_{V_i} & \mathfrak{D}\big(V_i^t; V_i^{\bar{t}}\big) \\ \text{s.t.} & \mathfrak{S}(V_i; V_{1..i-1}) < L \end{array} \tag{7.2}$$

Equation 7.1 in the Lagrangian of Equation 7.2, up to a negligible additive constant. We prefer this form because the trade-off parameter L has the same scale as $\mathfrak{S}(V_i; V_{1..i-1})$. For example, if we instantiate $\mathfrak{S}$ with a measure of correlation, then L will simply describe the maximal acceptable correlation between V_i and $V_{1..i-1}$.

4 Instantiation: Meet Ziggy

We now instantiate the dissimilarity measure $\mathfrak{D}$ and redundancy measure $\mathfrak{S}$.

4.1 Explainable Mass Dissimilarity

Let us begin with the dissimilarity $\mathfrak{D}$. As mentioned previously, we could borrow any general divergence measure from the statistics literature, such as the KL-divergence. However, these measurements operate as "black boxes": they describe the intensity of the differences, but they do not explain how the tuples differ. Our approach is diametrically opposed: we introduce the **Zig-Dissimilarity** $\mathfrak{D}_{Zig}$, a completely *explainable* mass dissimilarity measure.

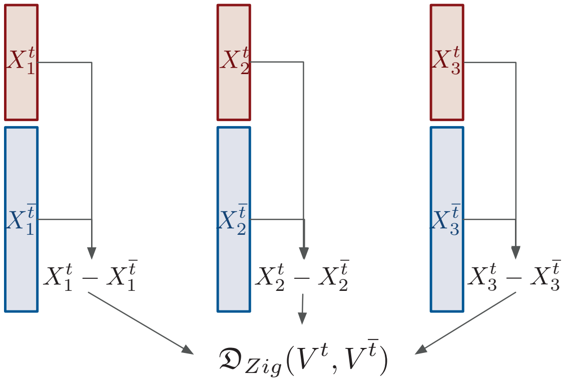


Figure 7.2: Illustration of Ziggy's Dissimilarity.

The idea behind our dissimilarity measure is to compute several simple, interpretable indicators of difference, called **Zig-Components**, and aggregate them in a single composite score, the Zig-Dissimilarity. Consider for instance two sets of tuples V^t and $V^{\bar{t}}$ with V columns. If these subsets have the same average for every column, then they probably come from the same distribution. Oppositely, if the averages are different, then they probably come from different distributions. From this observation, we can build an elementary dissimilarity measure: for each column X, we compute the differences between the means $\bar{X}^t - \bar{X}^{\bar{t}}$, as shown in Figure 7.2. These are our Zig-Components. We then aggregate these scores, by averaging their absolute values: we obtain the Zig-Dissimilarity.

Let us now formalize these ideas. We define Zig-Components as follows:

Definition 7.1. *Let $\mathbb{V}$ describe all possible subsets of tuples from a view V. A Zig-Component is a function $\mathfrak{z} : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ which describes one difference between two sets of tuples. If the tuples are similar, then $\mathfrak{z}(V^t, V^{\bar{t}}) = 0$. If not, the magnitude of $\mathfrak{z}(V^t, V^{\bar{t}})$ grows with the strength of the difference.*

For the sake of presentation, we abbreviate $\mathfrak{z}(V^t, V^{\bar{t}})$ as $\mathfrak{z}$. Once we computed the Zig-Components $\mathfrak{z}_1, \ldots, \mathfrak{z}_Z$, we obtain the Zig-Dissimilarity as follows. First, we compute the absolute values $|\mathfrak{z}_1|, \ldots, |\mathfrak{z}_Z|$ for each component. We then normalize the values across components of same type. We obtain a set of intermediary scores $z_1, \ldots, z_Z$. We aggregate these scores with a weighted sum. These operations give us one scalar indicator, which summarizes the magnitude of all the differences.

Definition 7.2. *Let $z_1, \ldots, z_Z$ represent a set of normalized absolute values of zig-components, and $w_1, \ldots, w_Z$ a set of weights. We define the Zig-Dissimilarity as follows:*

$$\mathfrak{D}_{Zig}(V^t, V^{\bar{t}}) \equiv \sum_{k \in [1,Z]} w_k \cdot z_k(V^t, V^{\bar{t}}) \tag{7.3}$$

In our implementation, we combined five types of Zig-Components, summarized in Table 7.1. A few of them come from the statistics literature, where they are referred to as *effect sizes* [22, 38]. We chose these indicators because they have intuitive interpretations, as well as known asymptotic properties (we will use those in Section 6). A majority, but not all of them are normalized and vary between -1 and 1. None of these functions are symmetric, and most of them change sign according to the direction of the underlying effect.

Observe that Ziggy computes univariate, but also bivariate components: in this case it compares columns by pairs. Our motivation is to check which correlations are present in V^t and not in $V^{\bar{t}}$, or more generally which correlations are either strengthened or weakened across the subsets. In the discrete case, we substitute the correlation coefficient by Cramér' s V. This function also measures the associations between two variables, and it varies similarly between 0 and 1 [22]. Its value is $\sqrt{\chi^2/(N(\min(N^t, N^{\bar{t}}) - 1))}$, where χ^2 is Pearson's χ^2 statistic, and $N^t, N^{\bar{t}}$ are the number of distinct values in X^t and $X^{\bar{t}}$ respectively.

In principle, we could test differences in spaces with more than two dimensions. We chose not to do so, for two reasons. First, the number of relationships to be inspected grows exponentially with the dimensionality of the Zig-Component. This hurts Ziggy's runtime, and leads to much longer outputs. Second, relationships in three dimensions or more are harder to convey and understand. We will show in Section 10 that this restriction has surprisingly little effect on Ziggy's accuracy in practice.

The aim of the weights w_k is to balance the effects across column types. For instance, we measure two components for the numerical columns, and only one for categorical data. Thus, we set $w_k = 1/2$ for the former and $w_k = 1$ for the latter. These parameters also let users express their preferences: for example, a novice user may value one-dimension Zig-Components over those based on correlation.

Property	Type 1	Type 2	Zig-Component	Comments
Mean	Contin.	-	$\mathfrak{z}_m^X = (\bar{X}^t - \bar{X}^{\bar{t}})/s^{\bar{t}}$	Known as Glass' Δ [38]
Stand. Dev.	Contin.	-	$\mathfrak{z}_\sigma^X = (s^t - s^{\bar{t}})/s^{\bar{t}}$	
Frequencies	Discrete	-	$\mathfrak{z}_\chi^X = \sqrt{\chi^2}$	Based on Pearson's χ^2 goodness-of-fit test [102]
Dependence	Contin.	Contin.	$\mathfrak{z}_r^{X_1,X_2} = r^t - r^{\bar{t}}$	$r_t$is the correlation coefficient between X_1^t and X_2^t $r_{\bar{t}}$ is the correlation coefficient between $X_1^{\bar{t}}$ and $X_2^{\bar{t}}$ [102]
Dependence	Discrete	Both	$\mathfrak{z}_V^{X_1,X_2} = V^t - V^{\bar{t}}$	V^t is Cramér's V coefficient between X_1^t and X_2^t $V^{\bar{t}}$ is Cramér's V coefficient between $X_1^{\bar{t}}$ and $X_2^{\bar{t}}$ [22]

Table 7.1: Our choice of Zig-Components for different data types. Each component describes a difference, to be evaluated for either each column X or each couple of columns X_1, X_2 in the view. The notations $\bar{X}$ and s respectively represent the sample mean and sample standard deviation. In the mixed-types case, we discretize the continuous column with equi-width binning.

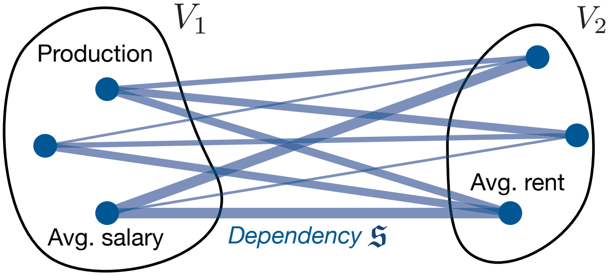


Figure 7.3: Illustration of Ziggy's redundancy measure. Vertices represent columns, edges represent pairwise dependencies.

4.2 Dependency Measure

We now present two instantiations for the measure of redundancy, $\mathfrak{S}_{hard}$ and $\mathfrak{S}_{soft}$. Both measures are a variant of the same principle, and we will refer to them collectively as the Zig-Dissimilarity $\mathfrak{S}_{Zig}$. Here again, our motivation is to aggregate several simple interpretable indicators.

Let V_1 and V_2 represent two views. We compute the Zig-Dependency in two steps. First, we compute the pairwise dependency $\mathfrak{s}(X_1, X_2)$ between every column X_1 of V_1 and every column X_2 of V_2, as shown in Figure 7.3. The measure $\mathfrak{s}$ is a convenience function, which allows us to combine different types of variables:

$$\mathfrak{s}(X_1, X_2) = \begin{cases} r(X_1, X_2) \text{ if } X_1, X_2 \text{ are continuous (correlation)} \\ V(X_1, X_2) \text{ otherwise (Cramér's V, cf. Sec. 4.1)} \end{cases} \tag{7.4}$$

During the second step, we aggregate these dependencies. The function $\mathfrak{S}_{hard}$ utilizes the maximum, while $\mathfrak{S}_{soft}$ uses the mean. If D_1 and D_2 respectively represent the number of columns in V_1 and V_2:

$$\mathfrak{S}_{hard}(V_1, V_2) = \max_{X_1 \in V_1, X_2 \in V_2} |\mathfrak{s}(X_1, X_2)| \tag{7.5}$$

$$\mathfrak{S}_{soft}(V_1, V_2) = \frac{\sum_{X_1 \in V_1, X_2 \in V_2} |\mathfrak{s}(X_1, X_2)|}{D_1 \cdot D_2} \tag{7.6}$$

Both functions $\mathfrak{S}_{soft}$ and $\mathfrak{S}_{hard}$ vary between 0 and 1, 0 indicating no dependency. The crucial difference lies in how they treat overlap. If one column is present in both V_1 and V_2, then $\mathfrak{S}_{hard}$ is at its maximal value 1, regardless of the other columns in the views. It is not necessarily so for

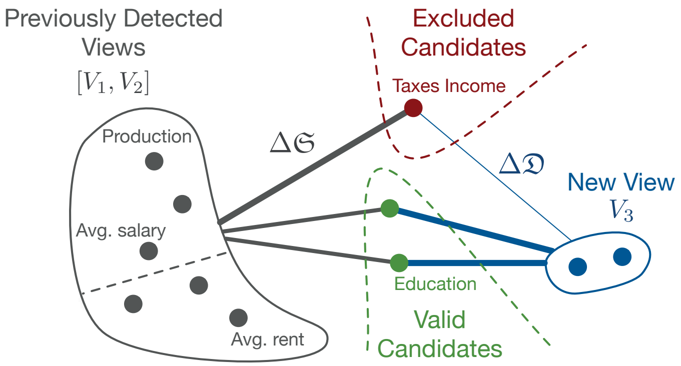


Figure 7.4: Ziggy's greedy view composition algorithm, with $i = 3$ and $D = 3$. The grey edges represent the amount of redundancy added by each candidate. The blue edges represent the amount of dissimilarity added by each candidate.

$\mathfrak{S}_{soft}$. Thus $\mathfrak{S}_{hard}$ leads to less redundancy, while $\mathfrak{S}_{soft}$ is more flexible. We will demonstrate these effect in our Experiments section.

Finally, observe that our choice for $\mathfrak{S}$ is computationally efficient: we get the pairwise correlations "for free", because we need to compute them anyway to obtain the Zig-Components $\mathfrak{z}_{\chi}$ and $\mathfrak{z}_{r}$.

5 Algorithms To Detect Views

We introduced a mass dissimilarity measure $\mathfrak{D}_{Zig}$ and a view dependency measure $\mathfrak{S}_{Zig}$. We now discuss how to maximize the former while constraining the latter, as exposed in Equation 7.2.

5.1 Base algorithm

Once the user has provided a selection of tuples, Ziggy performs two steps. First, it computes the Zig-Components, for each column and each pair of columns in the database. Then, it composes the views greedily: it successively picks the columns associated with the highest components, while ensuring that the dependency threshold L is not breached.

The first step is straightforward, but it is also by far the most time consuming. We discuss in Section 5.2 how to optimize it. During the second step Ziggy creates the views by adding columns in a best-first fashion. Figure 7.4 illustrates this approach. Suppose that Ziggy has

Algorithm 2 View Construction

```
function DetectViews($K$, $D$, $L$)
    for $k \in [1, K]$ do
        Cand $\leftarrow \{X_1, \ldots, X_M\}$
        Red.Gains $\leftarrow \{\Delta\mathfrak{S}(X, V_{1..k}),$ for $X \in$ Cand$\}$
        Cand $\leftarrow \{X$ where not Red.Gains$(X) > L\}$
        Dis.Gains $\leftarrow \{\Delta\mathfrak{D}(X, X'),$ for $X, X' \in$ Cand$\}$
        $V_k \leftarrow V_k \cup \{\text{argmax}_{X,X'}$ Dis.Gains$\}$
        for $d \in [3, D]$ do
            Cand $\leftarrow \{X_1, \ldots, X_M\} \setminus V_k$
            Red.Gains $\leftarrow \{\Delta\mathfrak{S}(X, V_{1..k}),$ for $X \in$ Cand$\}$
            Cand $\leftarrow \{X$ where not Red.Gains$(X) > L\}$
            Dis.Gains $\leftarrow \{\Delta\mathfrak{D}(X, V_k),$ for $X \in$ Cand$\}$
            $V_k \leftarrow V_k \cup \{\text{argmax}_{X}$ Dis.Gains$\}$
        end for
    end for
end function
```

previously obtained two views, V_1 and V_2, and it is currently building a third one, V_3. For each column in $V \setminus V_3$, our algorithm computes two scores: the gain of dissimilarity $\Delta\mathfrak{D}$ induced by adding the candidate to V_3, and the gain of redundancy $\Delta\mathfrak{S}$ induced by the same. Ziggy excludes the columns which exceed the redundancy capacity, e.g., those for which $\mathfrak{S}([V_1, V_2], V_3) + \Delta\mathfrak{S} > L$. It then detects the best candidate among those that remain and adds it to the view. It repeats the process until either the maximal number of columns D is met, or there are no more eligible columns. We present the pseudo-code in Algorithm 2.

To compute $\Delta\mathfrak{S}$, we apply Equations 7.5 and 7.6 directly. Computing $\Delta\mathfrak{D}$ requires more care, because each column is involved in several Zig-Components, and the bivariate components depend on the current state of the algorithm. Suppose for instance that we are building a view V_i, and we wish to compute $\Delta\mathfrak{D}$ for a given candidate X. In our implementation, if X contains numeric values, then it is associated with at least two components: the difference between the means $\mathfrak{z}_\mu$, and the difference between the standard deviations $\mathfrak{z}_\sigma$. We obtain those from the first step of the algorithm. But we must also account for the difference between dependencies, for all the columns already included in V_i. Thus, if z describes the normalized absolute value of a Zig-Component $\mathfrak{z}$, and if V_i^{num} and V_i^{cat} respectively represent the numerical and categorical variables of V_i the final score is:

$$\Delta\mathfrak{D} = z_{\mu}^{X} + z_{\sigma}^{X} + \sum_{X' \in V_i^{num}} z_r^{X,X'} + \sum_{X' \in V_i^{cat}} z_V^{X,X'} \tag{7.7}$$

The last two terms of the equation depend on the current state of V_i. Therefore we must update $\Delta\mathfrak{D}$ after each step.

If we use the redundancy measure $\mathfrak{S}_{hard}$, then we can avoid computing $\Delta\mathfrak{S}$ altogether: Ziggy can discard the redundant candidates before it starts building the view. Consider a candidate column X, and let $V_{1..i-1}$ describe the union of all previously built views. If $\mathfrak{S}_{hard}(V_{1..i-1}, X) > L$, then the column is not exploitable: adding it to the current view V_i will breach the threshold, regardless of V_i's current state. Conversely, if we have $\mathfrak{S}_{hard}(V_{1..i-1}, X) < L$, then the candidate is "safe", it will never breach the dependency threshold. Thus Ziggy, builds the view in two separate steps: first it eliminates the redundant columns (i.e., those for which $\mathfrak{S}_{hard}(V_{1..i-1}, X) > L$), then it selects the top D candidates.

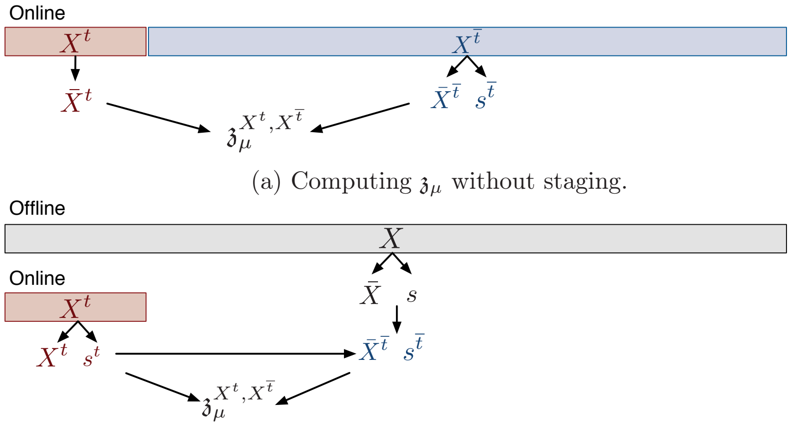
To conclude, our greedy algorithm is not exact, but it runs in $\mathcal{O}(KMD)$. Thanks to this heuristic, we avoid an exhaustive search of $\binom{M}{D}$ possible view combinations, which would lead to an unpractical $\mathcal{O}(KM^D)$ runtime.

5.2 Staging Computations

We now discuss how to compute the Zig-Components for each column of the database. This task is critical: in the best case, it requires a full scan of the database. In the worst case it runs in $\mathcal{O}(NM^2)$, because we need to compute and compare every possible pair of correlations in V^t and $V^{\bar{t}}$ to obtain the scores $\mathfrak{z}_r$ and $\mathfrak{z}_V$.

Our idea is to prepare some computations offline, before the user starts submitting queries. Let us focus on the Zig-Component $\mathfrak{z}_\mu$, which reports the difference $(\bar{X}^t - \bar{X}^{\bar{t}})/s^{\bar{t}}$, for a given column X of the database. Figure 7.5a presents the naive method to compute this component. As soon as our user submits a selection, we compute the average $\bar{X}^t$ for those tuples, the values $\bar{X}^{\bar{t}}$ and $s_{\bar{t}}$ for the rest of the database, and we apply the formula $(\bar{X}^t - \bar{X}^{\bar{t}})/s^{\bar{t}}$ directly. Thus, we read the whole column.

Figure 7.5b illustrates our staging strategy. Offline, we compute the mean $\bar{X}$ and the standard deviation s for the whole column X. Online, when the user submits a selection, we compute $\bar{X}^t$ and s^t only - thus, we read the selection and ignore the rest of the database. We then reconstitute



(a) Computing $\mathfrak{z}_\mu$ without staging.

(b) Computing $\mathfrak{z}_\mu$ with staging.

Figure 7.5: Our staging strategy.

$\bar{X}^{\bar{t}}$ and $s^{\bar{t}}$, as follows:

$$N^{\bar{t}} = N - N^t \tag{7.8}$$

$$\bar{X}^{\bar{t}} = \frac{N \cdot \bar{X} - N^t \cdot \bar{X}^t}{N - N^t} \tag{7.9}$$

$$(s^{\bar{t}})^2 = \frac{N}{N^{\bar{t}}} \cdot s^2 - \frac{N^t}{N^{\bar{t}}} \cdot (s^t)^2 - \frac{N^t}{N} \cdot (\bar{X}^t - \bar{X}^{\bar{t}})^2 \tag{7.10}$$

We now have all the elements to compute the Zig-Component. To obtain these equations, we used formulas designed to compute the mean and variance incrementally [73], and we reversed them - in fact we compute $\bar{X}^{\bar{t}}$ and $s^{\bar{t}}$ in a "decremental" fashion.

In effect, our staging scheme does not reduce the complexity of the algorithm, but it greatly reduces the amount of tuples to read. The smaller the user's selection is, the greater is the performance gain. Fortunately, we managed to extend it for all the Zig-Components presented in Table 7.1. We can derive similar computations to update correlation coefficients. If q represents the covariance between X_1 and X_2:

$$q^{\bar{t}} = \frac{N}{N^{\bar{t}}} \cdot q - \frac{N^t}{N^{\bar{t}}} \cdot q^t - \frac{N^t}{N} \cdot (\bar{X}_1^t - \bar{X}_1^{\bar{t}}) \cdot (\bar{X}_2^t - \bar{X}_2^{\bar{t}}) \tag{7.11}$$

To cope with categorical data, our approach is slightly different. Offline, we compute a histogram for X. Online we compute another histogram for X^t. From those, we can infer the distribution of $X^{\bar{t}}$'s values, and compute $\mathfrak{z}_X$ and $\mathfrak{z}_V$.

6 Model Validation

We now focus on the following problem: for a given view V, how *significant* is the Zig-Dissimilarity $S = \mathfrak{D}(V^t; V^{\bar{t}})$? A high value may indicate that V^t and $V^{\bar{t}}$ come from two different distributions. But it could also be caused by chance. How confident are we of this result? Confidence scores help Ziggy decide which Zig-Components to show, and it helps users interpret its results.

The statistics literature proposes a completely generic way to solve this problem: permutation testing. This methods works with the Zig-Dissimilarity, but it can also handle other divergence measures. In consists in repeatedly shuffling the rows of V, such that tuples are randomly affected to V^t or $V^{\bar{t}}$. We then observe how the dissimilarity S varies: if the permutations have no effect on S, then there is high chance that the dissimilarity was caused by chance. Oppositely, if S is very sensitive, then we have a high confidence in our result. We refer the interested reader to Wasserman [102] for more details.

Permutation testing offers plenty of advantages, but shuffling the rows is computationally heavy. In our implementation, we used an alternative approach: we exploit the composite nature of the Zig-Dissimilarity, and test each Zig-Component individually. We then aggregate these scores in a synthetic confidence indicator. Therefore we do not test the Zig-Dissimilarity directly - instead we focus on its underlying effects. This method is much lighter because we can use known asymptotic results, or at least approximations thereof. For instance, we know that under certain assumptions, we can test the difference between the means with a Wald test [102], which only requires an extra $\mathcal{O}(1)$ computation. Similarly, we can use a F-test for the ratio of the variances, or a χ^2 test for the differences in categorical distributions. Table 7.2 summarizes our choices. To aggregate the tests, we report the minimum observed confidence, a purposely conservative approach [102].

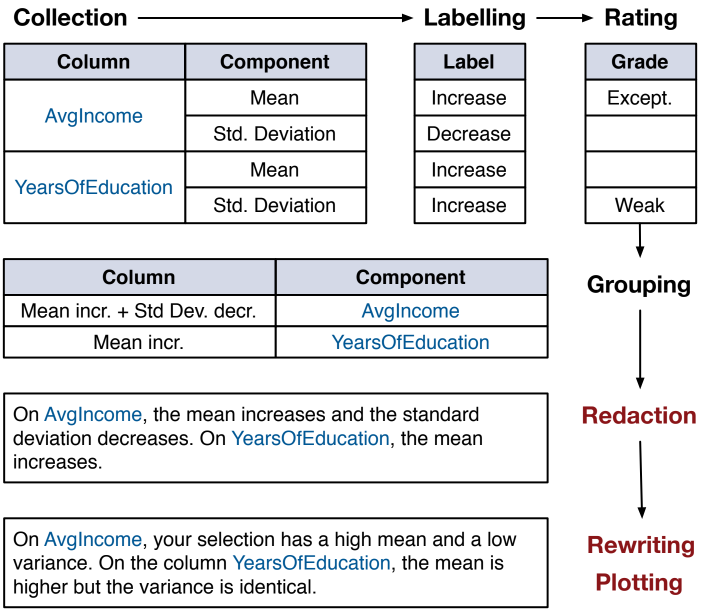
7 Report Generation

We explained how our system detects views. We now detail Ziggy's report generation pipeline, through which it justifies its choices.

Ziggy describes its views one by one. For each view, it generates a report, comprising a few sentences and several charts. The aim of the text is to give a general presentation of how the tuples differ from the rest of the data. The charts let users inspect the details. Figure 7.6 illustrates how

Property	Type 1	Type 2	Test Statistic	Comment
Mean	Contin.	-	$(\bar{X}^t - \bar{X}^{\bar{t}})/s^{\bar{X}^t - \bar{X}^{\bar{t}}}$	Wald test [102]
Stand. Dev.	Contin.	-	$s^t/s^{\bar{t}}$	F-test [22]
Frequencies	Discrete	-	χ^2	Pearson's χ^2 test [102]
Dependence	Contin.	Contin	$(Z^t - Z^{\bar{t}})/s^{Z^t - Z^{\bar{t}}}$	Z is the Fisher Z-transformation of the correlation coefficient r between X_1 and X_2 [31]
Dependence	Contin.	Both	$(V^t - V^{\bar{t}})/s^{V^t - V^{\bar{t}}}$	V is Cramér's V between X_1 and X_2. We obtain its distribution with Fisher's Normal approximation of $\sqrt{\chi^2}$ [72].

Table 7.2: Our choice of tests, corresponding to each Zig-Component. We use the same notations as in Table 7.1.



Column	Component	Label	Grade
AvgIncome	Mean	Increase	Except.
	Std. Deviation	Decrease	
YearsOfEducation	Mean	Increase	
	Std. Deviation	Increase	Weak

Column	Component
Mean incr. + Std Dev. decr.	AvgIncome
Mean incr.	YearsOfEducation

On AvgIncome, the mean increases and the standard deviation decreases. On YearsOfEducation, the mean increases.

On AvgIncome, your selection has a high mean and a low variance. On the column YearsOfEducation, the mean is higher but the variance is identical.

Figure 7.6: Ziggy's report generation pipeline.

Description	Exceptional?	Weak?
$\mathfrak{z}_\mu \geq 0 \wedge \bar{X^t} \geq 0$: “increase” $\mathfrak{z}_\mu \geq 0 \wedge \bar{X^t} < 0$: “decrease” $\mathfrak{z}_\mu < 0 \wedge \bar{X^t} \geq 0$: “decrease” $\mathfrak{z}_\mu < 0 \wedge \bar{X^t} < 0$: “increase”	z^X_μ in top 5%	$\lvert\mathfrak{z}^X_\mu\rvert <$ 0.2 or $p^X_\mu <$ 0.01

Table 7.3: Example of handwritten rules for the difference of means $\mathfrak{z}_\mu$. The variable p_μ describes the confidence (p-value), as described in Section 6.

Ziggy proceeds. First, it gathers all the Zig-Components associated with each columns of the database. It then generates a short description for each component using hand-written rules such as those presented in the first column of Table 7.3. It also rates each component, with three possible grades: “weak”, “neutral”, and “exceptional”. These grades are based on both the values of the components and their confidence, as illustrated in Table 7.3. To set the thresholds, we took inspiration from classic statistics textbooks [22], and chose conservative options. Nevertheless, these quantities are inherently arbitrary. It is therefore important to communicate them to the end users, and motivate each decision with its corresponding rule.

Once Ziggy has collected, labeled and rated the Zig-Components, it groups the columns which have the same “profile”, that is, the same variations on the same components. It then generates one sentence for each group, such that each sentence has the same structure: `on [column_names], the [zig-component] [label]`. If several components are involved, as in Figure 7.6, the Ziggy enumerates all the pairs `[zig-component] [label]`, separated by `and`. At this point, the text produced is understandable, but it is also highly redundant and possibly grammatically incorrect. During the last phase, Ziggy rewrites it with a set of hand-written rules. Such rules include inserting connectors (e.g., “Additionally”, “Finally”), using random synonyms (e.g., “the tuples”, “the selection”, “your data”) or replacing wider chunks (e.g., “has a lower variance” by “spreads more widely”).

By default, Ziggy only produces visualizations for the variables associated with exceptional components. It plots the remainder on demand. To determine which type of visualization to use, it checks the type of the columns, and applies usual techniques: it uses density plots and histograms for one dimensional data, and scatterplots and heat maps for two dimensional data.

8 Setting Parameters

Ziggy's model relies on three parameters: the total number of views K to generate, the width of the views D, and the dependency threshold L.

Number of Views K. When should we stop producing views? We propose to generate as many views as possible, and delegate the final decision to the users - after all, we have little idea about what they are seeking. In practice, we do so lazily: we start with a small selection of views (e.g., as many as the display can contain), and we generate the remainder on demand, for instance with a "show me more" button. In our experience, the number of views stays manageable because the algorithm is blocked by the redundancy constraint: after a certain number of views, Ziggy finds no more columns to exploit.

Size of the Views D. In our implementation, we set this parameter in an adaptive fashion, using a method presented by Zhu and Ghodsi [111]. We summarize it as following. When building a view, we keep track of the gains $\Delta\mathfrak{D}_d$ induced by each column d. We then detect change points in the sequence (e.g., "jumps" or "elbows"). If we observe such a behavior, we truncate the current view and start a new one. The advantage of this method is that D can adapt to each subspace. However, it is only a heuristic. Fortunately, inaccuracies have little consequence in practice: if the dependency constraint is weak, then the excluded columns are simply pushed to the next view.

Dependency threshold L. Admittedly, there is no optimal way to set this parameter: it depends entirely on the user and the exploration context. By default, we use $\mathfrak{S}_{hard}$, and we limit to $L = 0.99$. This setting enforces that the views are non-overlapping, but it adds no other constraint - we see this as a safe option.

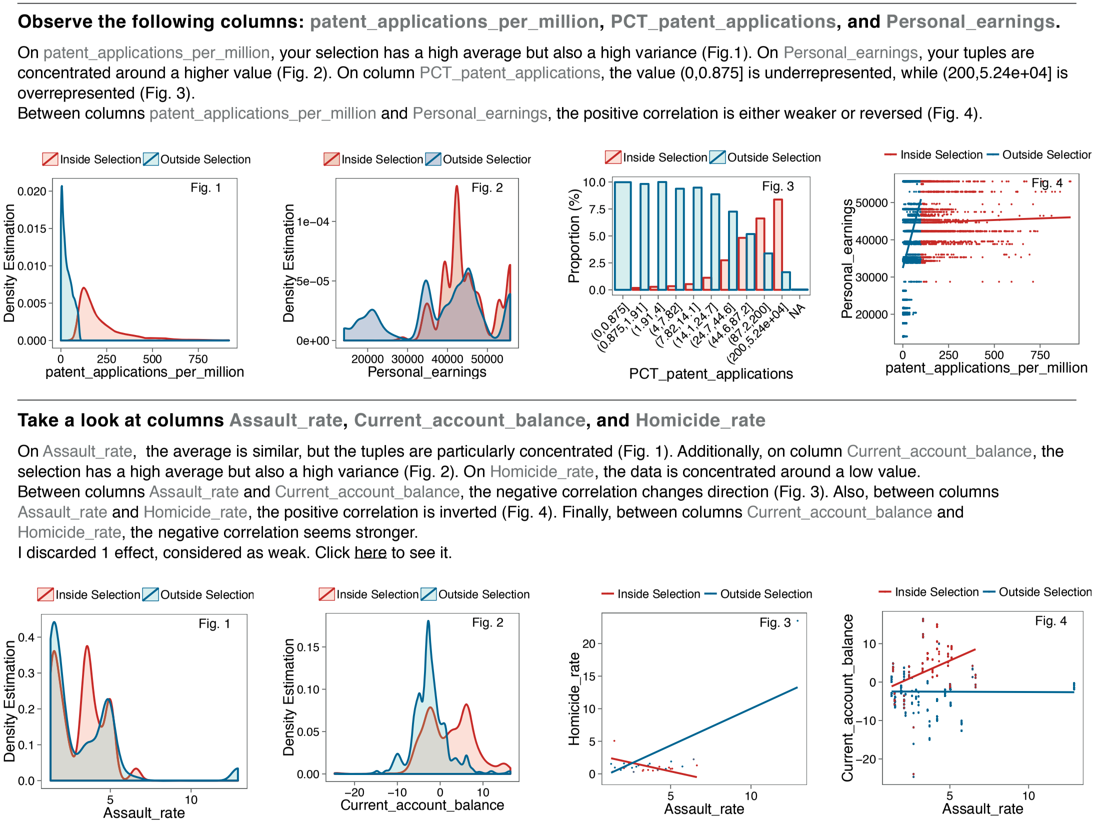
Zig-Components. Previously, we proposed several Zig-Components, in order to capture a wide range of effects. Nothing forces to use them all. For instance, a user interested in quick and concise results may prefer to ignore bivariate components.

9 Use Case

We now apply Ziggy on the data which inspired our running example. Our aim is to understand which factors lead to innovation, and more specifically patents. To answer this question, we aggregated several databases from the OECD, an international economic organization. All the data we used

Columns	Except. Comp. (%)	Weak Comp. (%)	$\mathfrak{D}_{Zig}$
Patent_applications_p_inhabitant PCT_patent_applications Personal_earnings	83.3	0	25.4
Dwellings_no_basic_facilities Educational_attainment Emp._work_very_long_hours Life_expectancy	64.2	7.1	18.9
Average_hours_worked Population_growth_rates Working_age_population Pop_under_the_age_of_15	50	42.2	12.4
Assault_rate, Homicide_rate Current_account_balance	77.7	11.1	12.1
Export_Pharmaceutical Incidence_part_time_emp Long_term_unemp Production_crude_oil	57.1	57.1	11.4
Air_pollution, Job_security Student_skills	77.7	11.1	10.1
Employment_rate Total_primary_energy_supply Trade_Balance._Pharmaceutical	66.6	33.3	7.1
Triadic_patent_year Time_devoted_to_leisure	28.5	75.5	6.2
Renewable_energy_supply Voter_turnout Total_tax_revenue	33.3	55.5	5.5
Consultation_on_rule_making Implicit_GDP_Price_Indices Years_in_education	7.1	78.5	5.3
Quality_of_support_network Taxes_on_income_and_profits	33.3	55.5	4.5
Value_Added_of_Industry Exchange_Rates Gross_Domestic_Product	0	100	2.9

Table 7.4: Detail of Ziggy's views, by decreasing order of dissimilarity. The two middle columns indicate the proportion of Zig-Components marked as Exceptional and Weak.



Observe the following columns: patent_applications_per_million, PCT_patent_applications, and Personal_earnings.

On patent_applications_per_million, your selection has a high average but also a high variance (Fig.1). On Personal_earnings, your tuples are concentrated around a higher value (Fig. 2). On column PCT_patent_applications, the value (0,0.875] is underrepresented, while (200,5.24e+04] is overrepresented (Fig. 3).
Between columns patent_applications_per_million and Personal_earnings, the positive correlation is either weaker or reversed (Fig. 4).

Take a look at columns Assault_rate, Current_account_balance, and Homicide_rate

On Assault_rate, the average is similar, but the tuples are particularly concentrated (Fig. 1). Additionally, on column Current_account_balance, the selection has a high average but also a high variance (Fig. 2). On Homicide_rate, the data is concentrated around a low value.
Between columns Assault_rate and Current_account_balance, the negative correlation changes direction (Fig. 3). Also, between columns Assault_rate and Homicide_rate, the positive correlation is inverted (Fig. 4). Finally, between columns Current_account_balance and Homicide_rate, the negative correlation seems stronger.
I discarded 1 effect, considered as weak. Click here to see it.

Figure 7.7: Ziggy's explanations for Views 1 and 4

may be found online[1]. Our core dataset is the Patents per Region database, which contains 15 years of patent statistics for 2,180 regions in 31 countries. We augmented this set with several other region-level databases (Demographics per Region and Labour per Region) and country-level indicators (Better Life, Well Being and Innovation Indicators). We obtain a table with about 6,823 rows and 519 columns, including mixed types and missing values. We filtered out the categorical columns with more than 20 distinct values (e.g., primary keys, countries and region names).

Our selection of tuples contains the top 10% regions for the statistic patent applications per inhabitant. We set $D = 6$, with the adaptive stopping method described in Section 8. We use $\mathfrak{S}_{hard}$, with a maximum value of $L = 0.75$. Ziggy detects a total of 12 views, which columns are reported in Table 7.4.

Some of Ziggy's choices are not surprising. For instance, the first column mentioned in the Table is precisely the variable we used to define our selection. The second one, PCT patent applications is highly similar (the PCT is an international patent treaty, which allows transnational patents). Likewise, we expected some relationship between education and innovation (views 2, 6 and 10). However, some effects are less straightforward. How does Employees working very long hours impact innovation? Are patents produced at work, or during hobby time? Similarly, how do our regions behave on the variable Job security?

Figure 7.7 presents Ziggy's explanations for two views. To obtain this figure, we made only two edits: we removed some plots to save space, and we inserted references to the charts in the text. Consequently, the figure illustrates accurately what Ziggy's users would obtain. The first view reflects the fact that innovative regions usually offer high incomes. Ziggy expresses this through its comments about the variable Personal earnings, but also through the last chart. As it points out, there is a correlation between patent applications and income, but this correlation disappears when we focus on innovative regions. Then, the regression line is flat, with a high offset, which indicates that all these regions are relatively rich, regardless of their patent statistics.

The second view offers a different perspective. The first and third charts show that innovative regions tend to be safer, but the relationship is not straightforward. Ziggy shows that these regions have less extreme values on Assault rate, but the mean is similar. Also, the expected correlation between Assault rate and Homicide rate is inverted. This

[1] http://stats.oecd.org/

reflects the fact that assaults do exist in our innovative regions, but little of them actually lead to a homicide. The last chart is more puzzling: normally, there exists no relationship between `Assault rate` and `Current account balance`. And indeed, we expect these variables to be independent, because they describe completely different topics. Yet, in our regions, a clear correlation appears. How can we interpret this effect? If this dependence completely spurious? Are those variables cofounded by a third, hidden variable? Or maybe sane public accounts causes anger among inventors? As implausible as this last hypothesis may seem, the chart gives us no way to discard it. We leave the question open for future investigations.

10 Experiments

10.1 Setup

Metrics. We now present our experimental results. We evaluate three aspects of our system: the *quality* the views, their *diversity*, and Ziggy's *runtime*. To evaluate the quality of the views, we simulate users with statistical classifiers. We assume that if a classifier can learn from a view, then so can a real user. Technically, we materialize the user's selection into a vector $\mathbf{t} = (t_1, \ldots, t_n)^\top$: $t_i = 1$ if the tuple is chosen, 0 otherwise. We then train a classifier, using the view $\mathbf{V}$ as feature set and the user's selection $\mathbf{t}$ as target. To obtain our quality score, we check if the classifier can accurately model the user's selection $\mathbf{t}$ with the variables in $\mathbf{V}$. If so, we deduce that the selection has a "peculiar" structure in the subspace, and therefore the view contains exploitable information. Oppositely, if the classifier cannot reconstruct the user's selection, then either the classifier is poor, or the view contains no information about the selection. We use a 5-Nearest Neighbor (5-NN) classifier, and we report the F1 measure with 5 cross-validation (higher is better). We chose the 5-NN algorithm for convenience: it is fast, and it gave us good predictive performance.

To measure diversity, we report the number of distinct columns used by the views in the result set (higher is better). We measure runtime with the total wall clock time, including preprocessing in Ziggy's case (lower is better).

Baselines. We compare Ziggy to four subspace detection methods: `Claude`, `Clique`, `Wrap-5NN` and `4S`. The first three methods are supervised: their aim is to detect informative columns for a classification or regression task. We adapt them to our problem by setting $\mathbf{t}$ as the target column. The last algorithm is unsupervised.

We already presented `Claude` in Chapter 5. Although its objectives are different, Claude resembles Ziggy on many aspects. Like Ziggy, Claude seeks "informative" subspaces. Also, it uses a multi-view approach: it returns a fixed number of subspaces, with a user-specified number of dimensions. But it differs on two points. First, it uses a different notion of interestingness: it seeks groups of columns which are strongly dependent to the target, dependency being measured with the mutual information. In contrast, Ziggy seeks subspaces where tuples with different target values behave differently, using simple, interpretable indicators of difference. Second, Claude builds all the subspaces simultaneously, using a beam search algorithm. Ziggy generates the views one after the other. We expect Claude to be accurate, but slower than Ziggy - after all, Claude was designed to generate queries, while Ziggy's aim is to support real-time interaction.

As in Chapter 5, we compare Ziggy and Claude to `Clique`, `4S` and `Wrap-5NN`. The first two algorithms are fast, but possibly approximative. The last one is very accurate, because it optimizes exactly what we are measuring. However, it is also very slow. Therefore, we only use it as a "gold standard" for our experiments with real data. We refer the reader to page 69 for more details about these algorithms.

Setup. We implemented Ziggy in R, exploiting its native primitives for critical operations (namely computing means, covariance matrices and cross-tabulation). We interrupted all the experiments which lasted more than 1 hour. Our test system is based on a 3.40 GHz Intel(R) Core(TM) i7-2600 processor. It is equipped with 16 GB RAM, but the Java heap space is limited to 8 GB. All the algorithms we present are single-threaded. The operating system is Fedora 16.

10.2 Synthetic Data

In this set of experiments, we benchmark our algorithms in a synthetic environment. Since we know the structure of the data, we can tune the competitors optimally. For instance, if we generate a dataset with 3 subspaces of 5 columns, then we set $K = 3$ and $D = 5$. We must however report that `4S` tunes itself: it computes how many subspaces to generate, and how large these should be. We use two versions of Ziggy: for `Ziggy-Soft` we use the dependency measure $\mathfrak{S}_{soft}$ and we limit it to 0.1. For `Ziggy-Hard`, we use $\mathfrak{S}_{hard}$ and we limit it to 0.9.

Our generator produces columns by groups. It yields two types of sub-

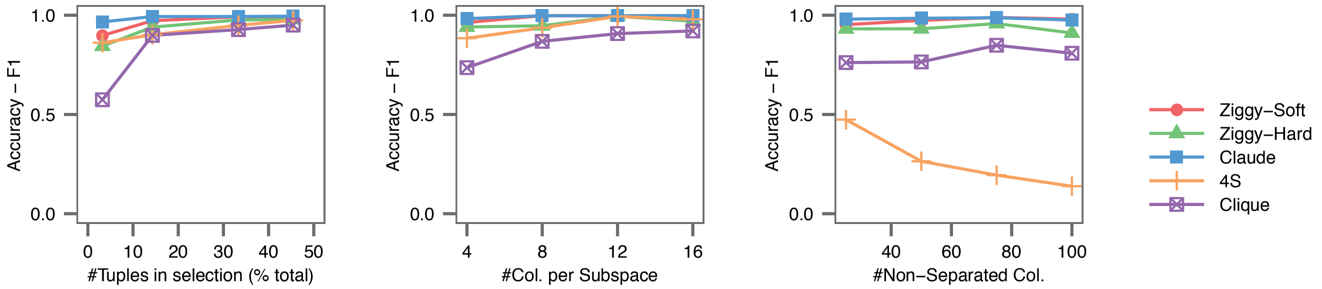


Figure 7.8: Average quality of the views varying the data generator parameters.

Parameter	Value
Selection (tuples)	3,000
Tuples from Gauss. mixture	15,000
Tuples from Unif. distrib.	15,000
Sep. / Non-sep. variables	20 / 4
Num. dimensions subspaces	4
Num. components Gaussians	5
Mean / Variance Gaussians	Unif. in [-10,10] / [1, 20]
Uniform noise	Unif. in [-45,45]

Table 7.5: Default parameters for our data generator.

space: *non-separated* subspaces, and *separated* subspaces. In the non-separated case, the selection and the rest of the data are sampled from the same distribution, namely a mixture of multivariate Gaussians with random parameters. In the separated case, the selection is sampled from a separate Gaussian distribution, with its own random parameters. Additionally, our generator produces uniform noise on all the columns. Table 7.5 presents our parameters. For each experiment, we generate 4 random data sets and report the average F1 of all the views.

10.2.1 Quality of the Views

The first chart in Figure 7.8 presents the robustness of the algorithms with regards to the size of the selection. For all our baselines, raising this parameter ameliorates the views, but the quality converges fast (at around 15% of the database size). The algorithm `Claude` comes first, but it is very closely followed by the two instances of `Ziggy` and `4S`. The ranking is almost identical in the second chart, which displays the accuracy of the algorithms varying the dimensionality of the subspaces. All algorithms involved seem robust, but `Ziggy-Soft`, `Ziggy-Hard` and `Claude` are above, followed closely by `4S`, then `Clique`. The last graph illustrates the robustness of the views with regards to the number of non-separated columns. All our baselines achieve good scores, except `4S`. We interpret this effect by the fact that `4S` is unsupervised, it has therefore no way to detect which subspaces are interesting and which are not. In conclusion, despite its simple assumptions, Ziggy delivers high quality, robust views, largely comparable to state-of-the-art feature selection algorithms.

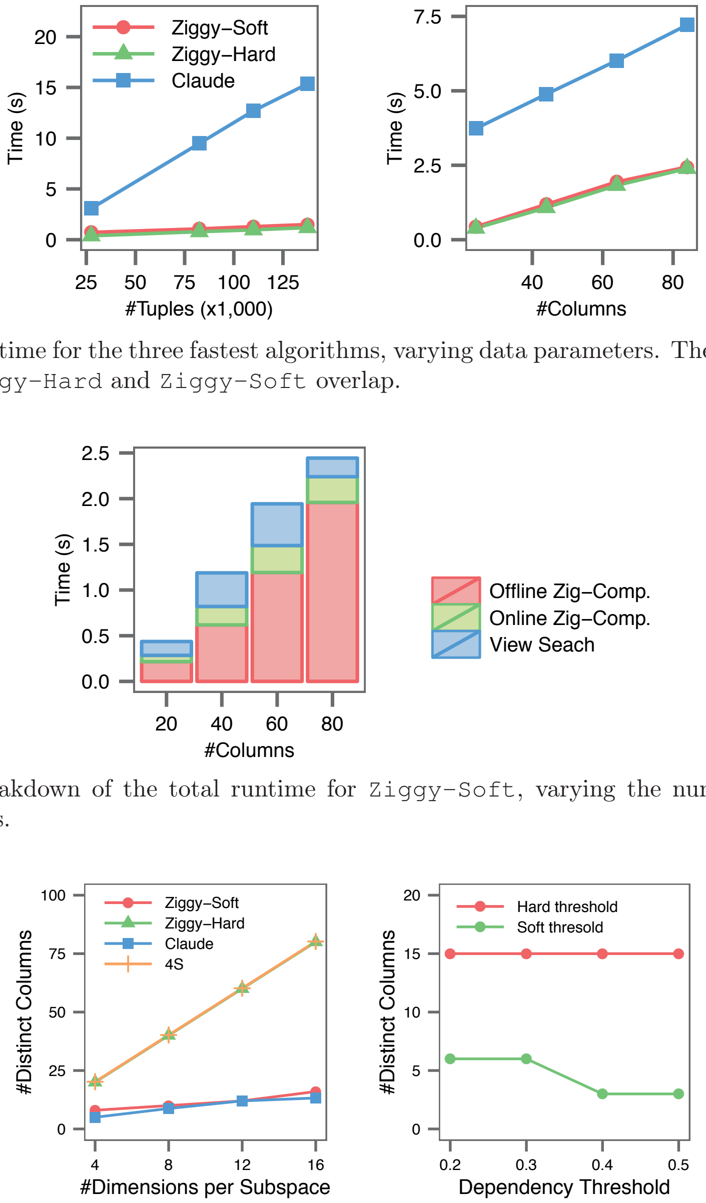
10.2.2 Runtime

Figure 7.9a illustrates the total runtime with regards to the number of rows and columns in the database. We ignored the algorithms 4S and Clique, which were slower than the other candidates. We observe that Ziggy's performance is spectacular: it is an order of magnitude faster than Claude, which is itself faster than the other competitors. And yet, all the algorithms involved in our benchmark have the same $\mathcal{O}(ND^2)$ complexity. We explain the difference with Ziggy's simpler computations. Ziggy relies on means and correlations, which are much lighter than Claude's information theoretic estimators. Besides, when evaluating its views, Ziggy considers at most two dimensions at a time, while its competitors test higher level dependencies.

Figure 7.9b shows where Ziggy-Soft spends its time. By far, the most expensive operations are the offline computations. This validates our staging strategy. Table 7.5 reports that the database contains 1 selected tuple for every 10 non selected tuples. Therefore we expected the online phase to be about 10 times faster, ignoring overhead costs. The figure confirms this estimation.

10.2.3 Diversity

Figure 7.9c presents the number of distinct columns mentioned in the views. In the leftmost chart, we compare four competitors, varying the width of the subspaces. We notice two distinct groups of algorithms. The approaches Ziggy-Hard and 4S offer a very high diversity, because they enforce that no column is shared among subspaces. The approaches Claude and Ziggy-Soft offer much less variety. The rightmost figure illustrates the effect of the dependency threshold. In the Hard case, it has no apparent effect because the views are completely disjoint anyway (the threshold does decorrelate the views, but this does not influence our metric). In the Soft case, we see that our deduplication strategy works: a lower threshold forces Ziggy to introduce variety.



(a) Runtime for the three fastest algorithms, varying data parameters. The points for `Ziggy-Hard` and `Ziggy-Soft` overlap.

(b) Breakdown of the total runtime for `Ziggy-Soft`, varying the number of columns.

(c) Diversity of the views.

Figure 7.9: Experiments with synthetic data.

Dataset	Columns	Rows	#Views	Dim Views
MuskMolecules	167	6,600	22	18
Crime	128	1,996	20	17
BreastCancer	34	234	10	13
PenDigits	17	7,496	9	10
BankMarketing	17	45,213	11	8
LetterRecog	16	20,000	10	12
USCensus	14	32,578	10	7
MAGICTelescope	11	19,022	1	10

Table 7.6: Characteristics of the datasets.

10.3 Real Data

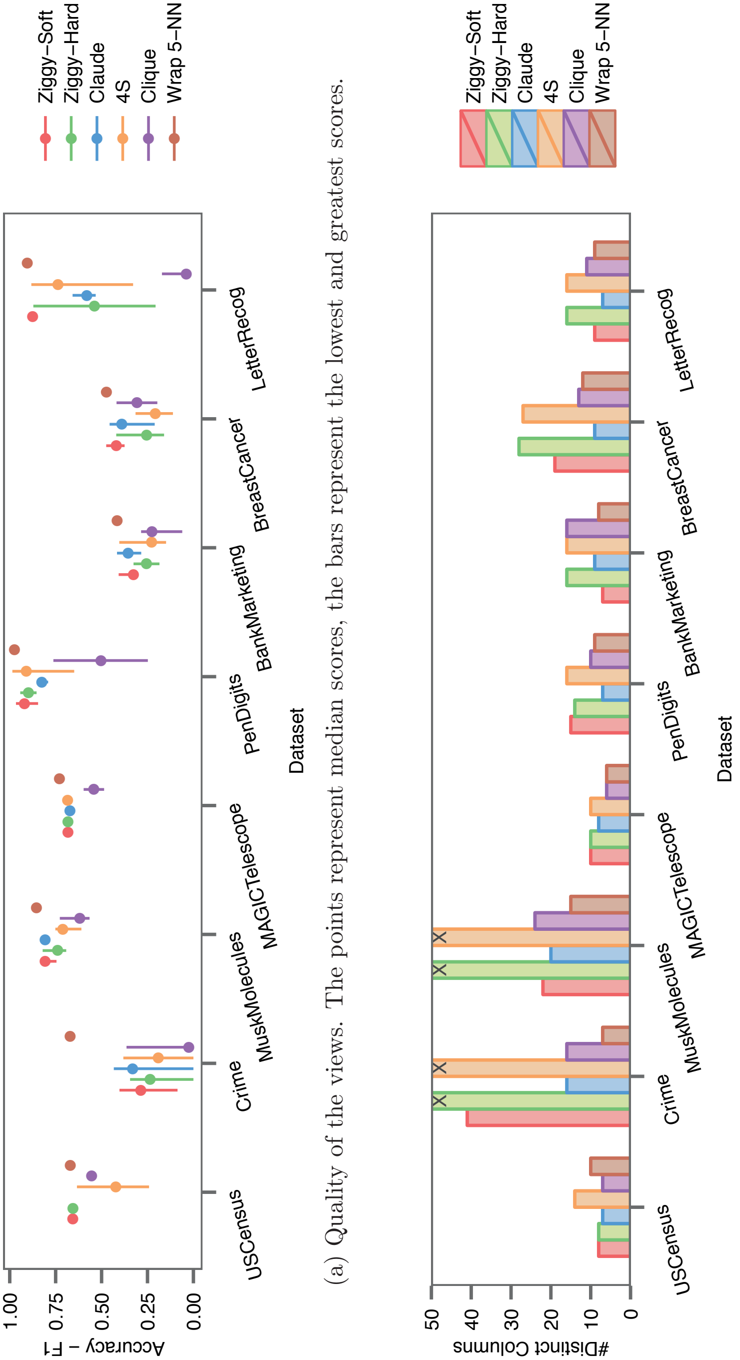
We now presents our experiments on real data from the UCI Repository. We use the algorithm `Wrap-5NN` as "gold standard", since it optimizes precisely the metric we report. To set the number and width of the subspaces, we rely on `4S`. Table 7.6 describes the datasets and our settings. Because all the competitors have the same parameters, the comparison is fair.

10.3.1 Accuracy

Figure 7.10a illustrates the quality of the views for each algorithm. We observe that `Wrap 5-NN` always comes first. It is closely followed by `Ziggy-Soft`, `Claude` and `Ziggy-Hard`, in different orders (although `Crime` is a striking counter-example). The unsupervised `4S` follows in most cases, tailed by `Clique`. Here again, we conclude that Ziggy's performance is largely comparable to good feature selection algorithms. However, we observe that `Ziggy-Hard` is often below `Ziggy-Soft`, the most extreme case being the `LetterRecog` set. This is a consequence of `Ziggy-Hard`'s diversity. In the synthetic case, it had several equally good subspaces to chose from. In the last three datasets, it seems that some subspaces are better than others, and therefore non-redundancy induces an accuracy loss (observe that `4S` suffers from the same problem).

10.3.2 Diversity

Figure 7.10b presents the diversity results. As with synthetic data, we observe that `4S` and `Ziggy-Hard` dominate the other algorithms, particularly with wide tables such as `Crime` and `MuskMolecules`. The algo-



(a) Quality of the views. The points represent median scores, the bars represent the lowest and greatest scores.

(b) Diversity of the views. The X mark indicates that we truncated the column.

Figure 7.10: Experiments on real data.

rithms `Ziggy-Soft` and `Clique` follow, then `Claude` and `Wrap-5NN` comes last - which we expected since they mostly target accuracy. This chart, in conjunction with Figure 7.10a, shows that the algorithms which generate the best F1 rarely generate the best diversity, and reciprocally. This motivates our choice to offer both $\mathfrak{S}_{hard}$ and $\mathfrak{S}_{soft}$.

10.3.3 Runtime

We present the runtime of our algorithms in Figure 7.11. The results are consistent with our previous conclusions: Ziggy outperforms all of its competitors, and the speedup gets more dramatic as the size of the datasets increase.

11 Related Work

11.1 Dimensionality Reduction

We mentioned dimensionality reduction techniques such PCA in Chapter 3. These methods differ with our work on three points. First, they are unsupervised: they process the data independently of any user selection. Second, they transform the columns of the database, by rotating and scaling them. Hence, their output can be difficult to interpret. In contrast, Ziggy operates on the raw data. Finally, they return a unique subspace, while Ziggy returns several non-redundant views.

11.2 Outlier Description

Outlier description consists in finding subspaces inside which a particular tuple is an outlier [9, 28, 50, 110]. This task inspired our work, but its objectives are different. Outlier description describes single objects, while we describe sets of tuples. It focuses on distance-based outliers, while we focus on probability distributions, regardless of how central or isolated the user's selection is. Authors focus on specific data types (either numerical or categorical, but not both), and little of them mention redundancy (an exception is [28], which seeks closed sets). None of these works discuss how to report results.

11.3 Contrast Mining

Contrast mining is a similar task, but in a pattern mining context: the aim is to differentiate two or more populations by identifying patterns

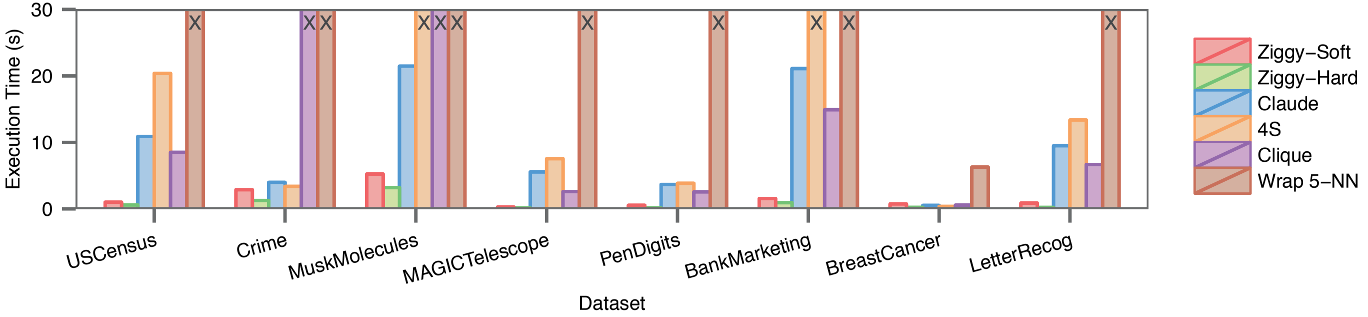


Figure 7.11: Runtime. The X mark indicates that we truncated the column.

which are present (e.g., have a high support) in one population but not in the other [99, 103]. This line of work is related but orthogonal, because we deal with neither itemsets nor patterns. Notably, Loekito and Bailey introduced a method to identify discriminative variables [57]. Yet, their work focuses on "contrast of contrasts" for categorical data, a close but different task

11.4 Feature Selection

As noted in Section 10, our work is intimately related to feature selection. Feature selection seeks predictive variables for a classification or regression task [36]. We can map this task to our problem by treating the user's selection as a column to be predicted. The main difference is that feature selection targets statistical accuracy, while we target interpretation. Thus, most feature selection algorithms seek one optimal set of variables, while we seek several small sets of variables. Also, feature selection tends to optimize class separability, while we are interested in any difference in distribution. Nevertheless, we acknowledge the similarity between our work and some algorithms, and compare them directly in Section 10.

11.5 Other Data Exploration Approaches

Recent works in data exploration have tackled different but related problems. Similarly to our work, SeeDB [97] recommends visualizations by seeking columns on which a set of tuples have an unusual behavior. However, it focuses on `GROUP BY` aggregates in data warehouses, while we focus on describing the general distribution of the selection, independently of any sub-grouping. Li and Jagadish [55] tackle the counterpart of our problem: they describe how to issue queries in natural language. Lloyd et al.'s automated statistician [56] describes regression models in natural language. These approaches are somewhat complementary to ours, and there would be much to gain by combining them.

12 Summary

In this chapter, we reversed the query recommendation problem. Instead of assuming that the users are looking for a query, we assumed that they already have one but they do not know what is interesting about it. To solve this problem, we introduced the idea of tuple characterization and presented a practical solution, Ziggy. Our experiments show that Ziggy can produce informative, non-redundant results within a fraction of the time taken by state-of-the-art machine learning algorithms.

Chapter 8

Raimond: Exploring Event Reports

1 Introduction

Previously, we presented software tools to help users explore structured data, that is, tables. We now extend our study to unstructured data, that is, plain text. How can we help users make sense of such datasets? How can we adapt semi-automated exploration to natural language documents? In fact, answering these questions would require at least a thesis on its own. For this chapter, we chose to focus on one specific scenario, inspired by an industrial use case. We show that extending semi-automated exploration to unstructured is indeed feasible, and we open the way for future research.

Let us now describe our use case. An analyst is interested in *news events*, and more specifically the series of earthquakes which shook Japan in March 2011. To understand these events, she has access to an archive of microblogs, i.e., tweets. Microblogging platforms provide access to an incredible volume of data. This data was produced as the events unfolded by individuals, official organizations, and news agencies from all around the world. It is diverse, timely, and exhaustive. But this volume is a mixed blessing. In the few days which followed the first shocks, several million posts were written and shared on Twitter. No reasonable human would consider reading them all. And yet, these posts may contain precious information.

In this chapter, we focus on one particular type of information: quantitative data. In our Japan example, how many earthquakes actually stroked the country? How many casualties were reported? How much funds were unlocked to help? We see that the event has a number of objective *quanti-*

This work was carried out during a summer internship at Microsoft Research (Moutaint View, CA), under the supervision of Dr Omar Alonso.

tative properties, such as cardinalities and measures. Collecting and analyzing these properties could lead to valuable insights. Our aim is to devise a systematic mechanism to extract and browse this information.

1.1 Contributions

We present Raimond, a virtual text curator for event reports. Raimond has two aims. First, it extracts quantities from microblogs. Second, it organizes its findings and presents them with thematic time lines. Thus, it builds veritable *narratives*, to be browsed by our analyst.

Raimond collects, cleans, organizes and recommends fragments of text which contain quantitative information. It operates as a pipeline, where each stage solves a different sub-problem. First, it identifies relevant tweets which contain quantitative data. Then, it groups those tweets into subtopics, removes the low quality content, and display the results. Given the complexity of the problem, we designed Raimond as a hybrid system. On one hand, we automated the data intensive parts of the extraction process. On the other hand, we let humans interpret the text through a crowdsourcing platform. Experiments with real-life data will reveal that Raimond can effectively capture and organize useful facts about news events. Hence, we show that generalizing data exploration beyond simple tables is feasible and promising.

1.2 Outline

In Section 2, we present the notion of quantfrag. In Section 3, we detail how Raimond extracts quantitative data. Section 4 showcases Raimond with real-word examples. We present our evaluation in Section 5, related work in Section 6 and conclude in Section 7.

2 Introducing the Quantfrag

Overview. The central concept behind Raimond is the *quantfrag*. A quantfrag is a snippet of text which contains a piece of quantitative information. Observe for instance the following tweets, recorded after the 2011 earthquakes in Japan[1]:

```
"Breaking News:  A 8.8 earthquake just hit #Japan."
"At least 2,369 are missing after #quake.  I have no
```

[1] All the examples in this section are based on actual tweets. Nevertheless, we took the liberty to truncate the original posts to shorten the presentation.

```
words."
"This is insane.  The Earth's rotation sped up by 1.6
microseconds.  #japan #planet"
```

Each post contains some quantitative information, surrounded by comments or details about the context. We call quantfrags the fragments of text which contain the quantities. We highlight these fragments in bold in the example. Ideally, a quantfrag should contain enough information to understand the quantity, but no more. It should be self-contained, but short. This leads to our first definition:

Definition 8.1. *A quantfrag is a complete, minimal piece of text which describes a fact based on a quantity.*

Not all tweets contain quantfrags. We use the term *qweets* for those which do. We illustrate our terminology in Table 8.1. Raimond's aim is to detect qweets, extract quantfrags, and present the collection in a browsable form.

Natural catastrophes are not the only events which yield quantitative data. The following quantfrags describe the 2014 World Cup Brazil-Germany game:

```
"BRA undefeated in 62 straight competitive home games
since 1975"
"GER have now scored 221 goals in WorldCup history"
```

These quantfrags were produced during the 2014 Ukraine political crisis:

```
"EU to provide $15 billion help package to Ukraine"
"Crimea referendum:  97% voted to join Russia"
```

We will present these two topics in detail in Section 4.

Detection. We now discuss how to detect qweets and quantfrags algorithmically. Most qweets contain numbers, written with letters or digits. However, Twitter data also contains a plethora of counter-examples. A post can describe a quantity without using any number:

```
"the country's strongest earthquake on record"
```

Also, it is not difficult to find numbers without quantities:

```
"Japan I pray 4 U"
"Please text the words Text Red Cross to 90999"
"Barack Obama will give a special address at 1130"
```

To complicate the matter further, many fragments form valid quantfrags, but they teach us little about the event:

```
"A fire has broken out at Cosmo Oil’s 220,000 b/d Chiba
refinery after earthquake."
"I have a friend in japan.  And he actually owes me ten
bucks."
```

These examples show that reporting all tweets which contain numbers is a very naive solution. Raimond relies on the combination of several methods, which we will discuss thoroughly in the following section.

Single Quantfrags, Serial Quantfrags. During our experiments, we encountered two types of quantfrags. *Single quantfrags* state independent, self-contained facts. For instance, the following quantfrag is single:

```
"The Pacific Plate slid west by 79 feet"
```

Oppositely, *serial quantfrags* describe the same property of the event, but at different points in time. Therefore, they describe a time series. Here is an example of such fragments:

```
11 March 2011 - "530 people were reported missing after
#earthquake in Japan"
12 March 2011 - "about 1800 missing in #japan as a result
of #earthquake"
15 March 2011 - "at least 3,743 are missing #earthquake
#tsunami"
```

These three quantfrags describe the number of people reported missing after the earthquakes, but at different points in time. They are particularly interesting because they let us reconstitute the original time series, as shown in Figure 8.1. One of Raimond’s functions is to organize the quantfrags in subtopics, such that serial quantfrags are displayed together.

Validity. In general, qweets may contain approximations, omissions, exaggerations or time lags. Unfortunately, this noise is inherent to social data. For instance, thousands of tweets mentioned 88,000 missing people during the Japan earthquakes. We found no trace of the original report, and official sources hint that this number is largely overestimated[1]. Our aim is to depict microbloggers’ views on events, regardless of their overlap with objective truth. Fact checking is, for now, beyond the scope of this study.

[1] www.jst.go.jp/pr/pdf/great_east_japan_earthquake.pdf, page 13

Table 8.1: Illustration of our terminology.

Tweet	`Japan update:` **`five nuclar plants shut down in Japan,`** `tsunami waves continue to hit`
Event	2011 Japan Earthquakes
Quantfrag	`five nuclear plants shut down in Japan`
Property	Nuclear plants shut down
Quantity	5
Is a Qweet?	Yes

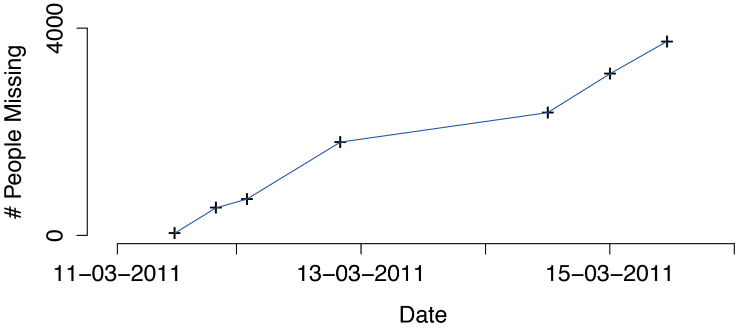


Figure 8.1: Time series reconstituted from seven serial quantfrags.

Table 8.2: Seeding the Raimond pipeline.

Type	Input field
Content	Hashtags
	Keywords
	Language
Network	Twitter's `verified` flag
	Account's followers
	Message retweets

3 Methodology

Raimond's goal is to detect and organize quantfrags. To do so, it operates in four consecutive stages, pictured in Figure 8.2. First, Raimond detects the most promising tweets, and extracts the quantfrags. Then, it groups the fragments which cover the same topic. During the third phase, Raimond filters out the fragments which are irrelevant or not informative with a combination of coded rules and crowdsourcing. Finally, it labels and displays the clean groups.

3.1 Extracting Quantitative Data

During this first phase, Raimond detects the tweets associated to the event of interest, parses them and retrieves the quantfrags.

Setup. To seed the Raimond pipeline, we define an event configuration. The configuration specifies which authors to follow and which tweets to select. For our Japan example, we tracked the hashtag `#japan` during 5 days, and selected the posts with more than 25 retweets. Table 8.2 shows all the settings offered by Raimond. The aim of content-related parameters is to spot relevant tweets. Network-related parameters measure trust and influence.

Selection. Once the event configuration is defined, Raimond fetches the tweets from our archive, and it applies a filter to discard tweets with no quantities. At this point, we include every tweet which could potentially be interesting, regardless of it its quality - we value recall much more than precision. The filter relies on two tests, assembled in a disjunction. The first test uses a quantity classifier. The classifier is based on statistical learning, and it was trained by Microsoft staff. For the second test, we wrote a set of regular expressions. These regular expressions detect cardinal and ordinal numbers, expressed with letters or numbers. At the end

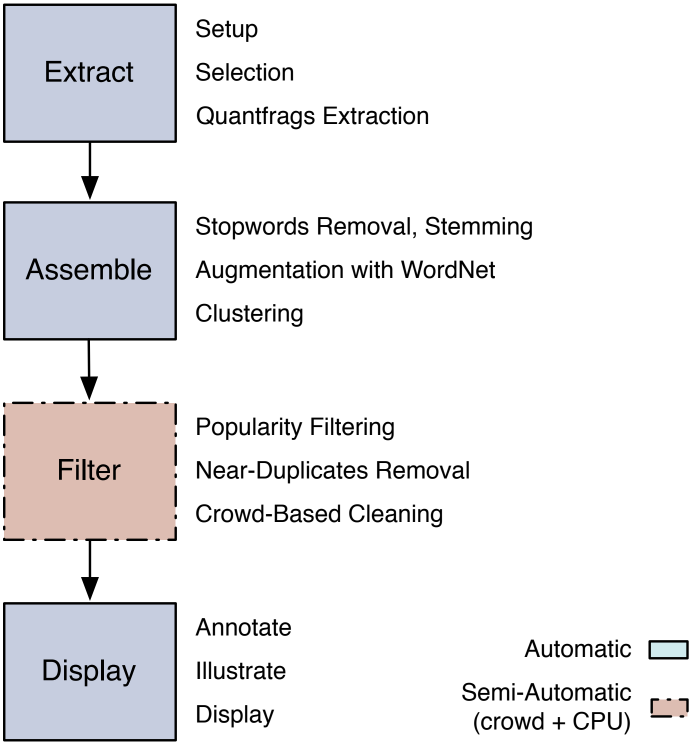


Figure 8.2: Overview of the Raimond pipeline

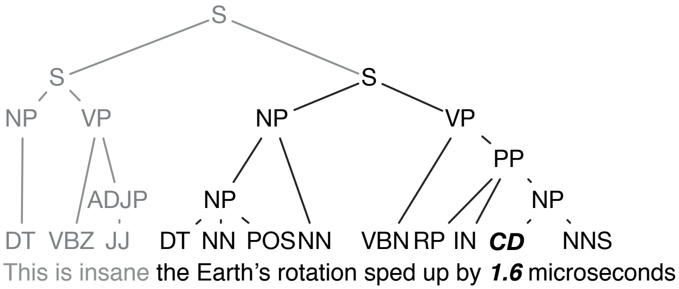

S
S
S
NP
VP
NP
VP
PP
ADJP
NP
NP
DT
VBZ
JJ
DT
NN
POS
NN
VBN
RP
IN
CD
NNS
This is insane the Earth's rotation sped up by 1.6 microseconds

Figure 8.3: Extracting quantfrags from the parse tree. The nodes of the tree represent constituent tags, as defined by the Penn Treebank. Our aim is to extract the subtree which contains the quantfrag. The quantfrag is highlighted in bold.

of this phase, we obtain a set of potential qweets, which typically contains lots of false positives.

Extraction. During this phase, Raimond extracts the quantfrags. Previously, we defined quantfrags as complete, minimal pieces of texts which convey a quantity. Unfortunately, evaluating whether a quantfrag is complete and minimal depends a lot on the user and the use case. Our definition is not practical. We propose to operationalize the notion as follows:

Definition 8.2 (operational)**.** *A quantfrag is a grammatical clause which contains a quantity.*

To detect clauses with numbers, we use a grammatical parser. The parser takes a tweet as input, and returns a tree, as pictured in Figure 8.3. In this tree each node represents a grammatical constituent. We check if the tree contains a quantity, tagged `CD` (Cardinal number) in the example. If it does, we extract the smallest clause which contains this quantity (`S` in our example). If we detect several numbers, we extract one clause for each. We used an internal Microsoft parser trained specifically for tweets, but several open source NLP suites can handle this type of task (e.g., Stanford NLP).

3.2 Assembling Quantfrags

In this phase, Raimond aggregates the quantfrags which describe the same topic, or, in some cases, the same variable (cf. serial quantfrags in Section 2). To achieve this, Raimond uses cluster analysis. As the quantfrags are short and noisy, preprocessing is crucial.

Preprocessing and Augmentation. To clean the quantfrags, we apply classic preprocessing operations: we replace smileys by keywords, we remove punctuation symbols and stop words, and we stem every term. Typically, the quantfrags we obtain are very short. This is problematic for clustering, because they are not likely to share terms. Consider for instance the following two quantfrags:

```
"Troops of 500+ to provide help"
"More than 500 militaries sent for assistance"
```

Both phrases have exactly the same meaning, yet they do not have any word in common. We use a lexical database, WordNet [64], to tackle this problem. For a given term, WordNet gives us hypernyms. Intuitively, a hypernym is a semantic superclass of a term. For instance, `army unit` is a hypernym of `troop`. Thanks to hypernyms, we can *augment* our quantfrags. We query the WordNet database for each noun and append the results to the fragment. This increases the chance that similar tweets share words. For instance, if we augment the first noun in each of our example tweets, we obtain:

"Troops *army unit* ***military force*** of 500+ to provide help"
"More than 500 militaries ***military force*** *organization* sent for assistance"

WordNet entries are organized in a hierarchy: hypernyms themselves have hypernyms. Therefore, we can expand our terms with several levels of generality. We used two levels of recursion in the example, we use three in our system.

In many cases, nouns have several competing WordNet entries. Each entry is represented by a set of synonyms, such as `assistance - aid - help`, or `assistance - financial aid - economic aid`. To resolve the ambiguity, we check how many of the synonyms are contained in the corpus, and keep the entry with the highest count. If the procedure finds no match, we take the most frequent sense. We refer the reader to the work of Hotho et al. for an empirical validation of this method [39].

Clustering. We represent the quantfrags with bags of words, and cluster them with agglomerative clustering [87]. We chose this approach because it is simple enough to be tuned by non-technical users. Recall from Chapter 3 that agglomerative clustering operates bottom-up. To initialize the algorithm, we assign each quantfrag to its own cluster. Then, at each iteration, we detect which two clusters are the closest, and merge them. As the algorithm runs, the clusters get larger. We stop when we reached a threshold. The algorithm requires three parameters, summarized in Table 8.3.

3.3 Filtering Irrelevant Quantfrags

During the two first phases, Raimond typically accumulates lots of false positives. Some quantfrags do not contain any quantity (`"Japan, I pray 4 u"`), are not related to the topic (`"Japan, thank you for Playstation 4!"`), are not informative (`"3 reasons why we must help Japan"`) or simply redundant. To make things worse, the clusters we detect are rarely perfect, as they may combine unrelated but lexically similar topics. To address this problem, we developed a cascade of filters, based on automatic rules and crowdsourcing. We summarize the filters in Table 8.4, and detail them below.

Filtering on Popularity. Typically, the size of the clusters obey approximately a power-law distribution. We observe a few large clusters, and a long tail of micro-topics. Raimond gives the the option to select the large clusters (the head of the distribution) and discard the smaller groups. The rationale is that large clusters describe popular topics, while smaller clusters may contain noise, such as personal reaction or irrelevant facts.

Near-Duplicates Removal. So far, we have kept (near) duplicates to assess the popularity of the topics. We now eliminate the redundancy. In fact, this task is close to the clustering phase, described in 3.2. We detect near-duplicates with the exact same method, but we operate at a thinner granularity. We reuse the dendrogram structure produced at the end of the clustering phase, and we cut it at a low level of dissimilarity (by default, 0.1). We obtain lots of micro-clusters, we represent each of them by a representative quantfrag (by default, the most frequent one).

Crowd-Based Cleaning. At this stage, the collection of quantfrags still contains false positives, with numbers but no quantities. It also contains uninformative quantfrags, i.e., quantfrags which are technically valid but provide no useful information about the event. We discard those with human computation.

Our crowdsourcing strategy is based on two consecutive tasks. During the first task, workers evaluate the overall quality of the clusters. They assign a grade to each cluster, based on a relevance. We aggregate the scores, and check if the value is above a certain threshold. If not, we

Table 8.3: Parameters for the cluster analysis.

Parameter	Range	Default
Distance	Cosine, Euclidean, p-Minkowski	Cosine
Linkage	Single, Complete, Average	Average
Maximum distance	0 - 1.0	0.9

Table 8.4: Sequence of filters used to remove false positives.

Precision Level	Filter	Computation
Cluster	Size	Machine
Quantfrag	Near-duplicates	Machine
Cluster	Relevance	Machine + Crowd
Quantfrag	Relevance	Machine + Crowd

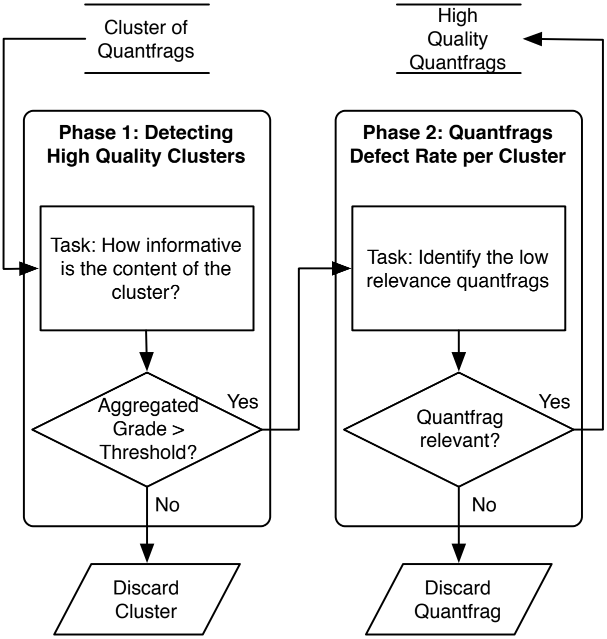

Cluster of Quantfrags
High Quality Quantfrags
Phase 1: Detecting High Quality Clusters
Task: How informative is the content of the cluster?
Aggregated Grade > Threshold?
Yes
No
Discard Cluster
Phase 2: Quantfrags Defect Rate per Cluster
Task: Identify the low relevance quantfrags
Quantfrag relevant?
Yes
No
Discard Quantfrag

Figure 8.4: Flow-Chart illustrating our crowdsourcing strategy to select high quality quantfrags.

discard the cluster. We then run another task, in which the goal is to identify low quality quantfrags within the clusters. Figure 8.4 describes the overall process. We can think of this approach as a two-step quality control: the first phase checks if the cluster is relevant to the event. The second phase provides a defect rate per cluster. The final output is a set of high quality clusters, with useful quantfrags. In our surveys, we avoid spammers with purposely trivial questions and redundancies.

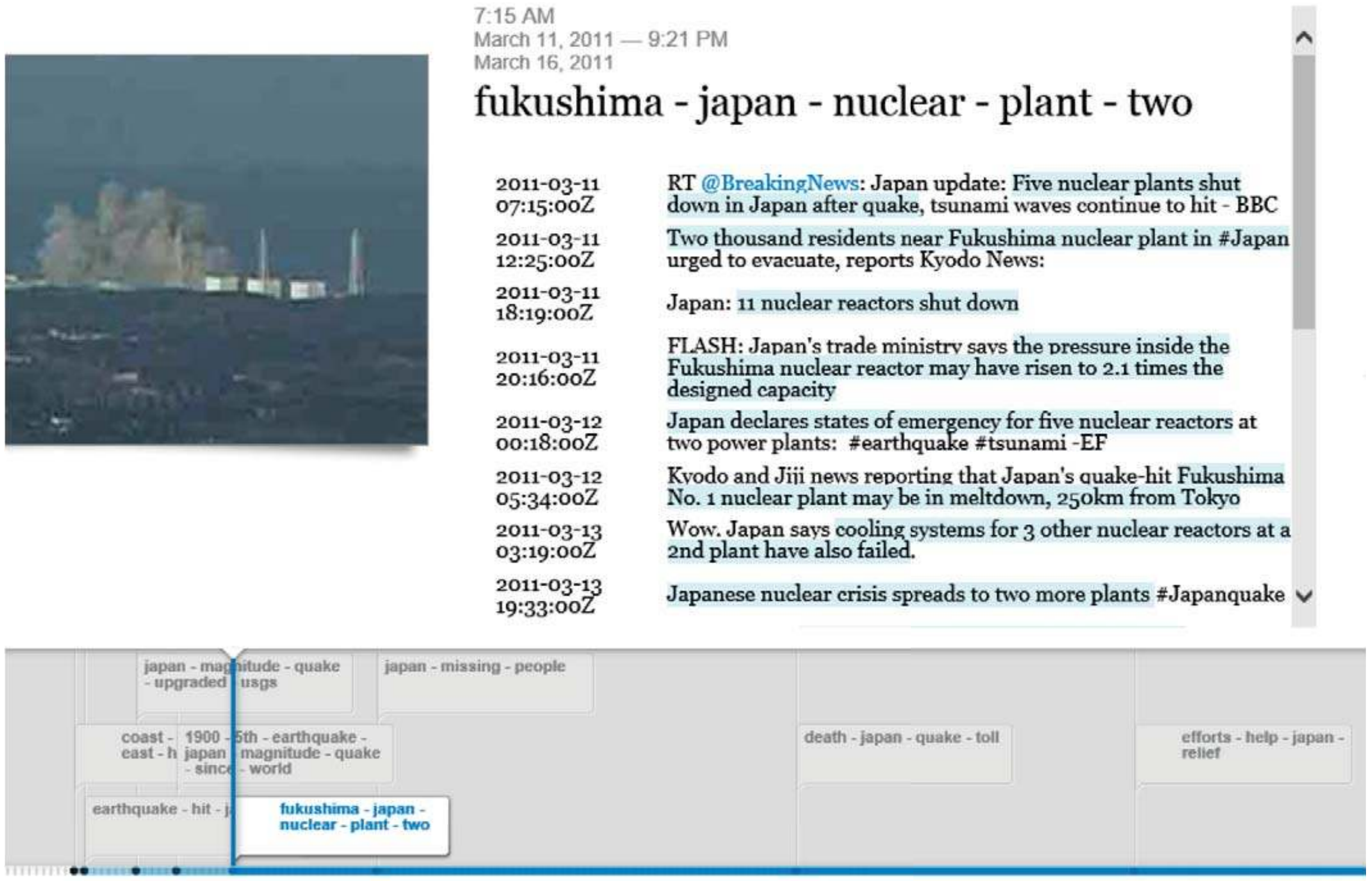
3.4 Annotation and Visualization

The aim of the last step is to annotate and display the clusters of quantfrags. The operations described in this section do not add content, but they enhance the presentation of the quantfrags.

Title. Raimond summarizes each cluster with a title. To produce the title, it creates documents by concatenating the quantfrags of each cluster. Then, it computes a tf-idf matrix, and reports the top k terms for each cluster/document (we set $k = 5$ for the rest of this chapter).

Illustration. We observed that many qweets contain links to images. Our idea is to exploit these links to illustrate the clusters. Raimond parses the tweets for image URLs with a set of regular expressions. If it encounters such URLs, it tries to download the documents. It then presents the images side-by-side with the quantfrags in the interface. If a cluster links to several images, Raimond presents them sorted by decreasing order of popularity (using the number of retweets).

Display. Raimond's last task is to display the quantfrags. We provide a screenshot of the interface in Figure 8.5. The bottom part of the display presents the titles of the clusters on a timeline. To anchor the labels, we calculate *peak dates*. The peak date of a cluster is the timestamp at which it is the most popular. To calculate it, we retrieve the dates at which the quantfrags of the cluster are mentioned, estimate a density function with Gaussian density estimation and compute the mode of this distribution. We will present some examples in Section 4. Users can focus on a cluster by clicking on its title. Then, Raimond displays the quantfrags with their timestamps and the qweets from which they were extracted.



7:15 AM
March 11, 2011 — 9:21 PM
March 16, 2011

fukushima - japan - nuclear - plant - two

2011-03-11 07:15:00Z	RT @BreakingNews: Japan update: Five nuclear plants shut down in Japan after quake, tsunami waves continue to hit - BBC
2011-03-11 12:25:00Z	Two thousand residents near Fukushima nuclear plant in #Japan urged to evacuate, reports Kyodo News:
2011-03-11 18:19:00Z	Japan: 11 nuclear reactors shut down
2011-03-11 20:16:00Z	FLASH: Japan's trade ministry says the pressure inside the Fukushima nuclear reactor may have risen to 2.1 times the designed capacity
2011-03-12 00:18:00Z	Japan declares states of emergency for five nuclear reactors at two power plants: #earthquake #tsunami -EF
2011-03-12 05:34:00Z	Kyodo and Jiji news reporting that Japan's quake-hit Fukushima No. 1 nuclear plant may be in meltdown, 250km from Tokyo
2011-03-13 03:19:00Z	Wow. Japan says cooling systems for 3 other nuclear reactors at a 2nd plant have also failed.
2011-03-13 19:33:00Z	Japanese nuclear crisis spreads to two more plants #Japanquake

japan - magnitude - quake - upgraded usgs
japan - missing - people
coast - east - h
1900 - japan - since
5th - earthquake - magnitude - quake - world
death - japan - quake - toll
efforts - help - japan - relief
earthquake - hit - j
fukushima - japan - nuclear - plant - two

Figure 8.5: Quantfrags presentation.

Table 8.5: Filtering and extraction of quantfrags. The sets are sorted by inclusion - each set is a refinement of the previous one. The Japan set was filtered and deduplicated before our experiments.

Dataset	Ukraine	BRAvGER	Japan
Tweets	7,326,838	16,481,551	3,049,463
. Trusted	441,151	992,980	NA
.. Unique	10,508	6,438	6,210
... Contain quantities	1,093	1,207	1,729
.... Quantfrags	718	762	1,354

4 Use Cases

In this section, we present our experiments with three datasets. The first dataset is based on the 2011 Japan earthquakes, discussed throughout the chapter. The second dataset describes the political crisis in Ukraine, still ongoing at the time of writing. To obtain it, we tracked the hashtag `#ukraine` during 134 days. The third dataset contains tweets about the Brazil-Germany football game of the 2014 World Cup. Using five hashtags, we gathered approximately 16 millions of Tweets in less than 24 hours. We detail our data collection methodology and event configurations in Table 8.6.

Table 8.5 shows the size of the data as Raimond processes the tweets. We start with several million tweets. We tuned the pipeline to extract only those that come from official sources and news accounts (cf. Table 8.6). We obtain less than a million tweets (about 5% of the initial volume). This number includes the tweets written by official sources, but relayed by non-trusted individuals. After removing the retweets and the duplicates, we obtain less than 10,000 posts. This decrease is spectacular, but not surprising: by definition, popular accounts are massively retweeted. For instance, in the `BRAvGER` dataset, posts about spectacular actions and goals are retweeted by thousands of supporters. At the end of the pipeline, after filtering, cleaning and aggressive deduplication, we obtain a few hundred quantfrags.

Table 8.6: Data collection methodology and event configuration.

	Ukraine	BRAvGER	Japan
Hashtags	#ukraine	#bra #ger #bravger #brazil #germany	#japan
Start date	1 Jan 2014	08 Jul 2014	10 Mar 2011
End date	15 May 2014	08 Jul 2014	15 Mar 2011
Author checks	Min. 200,000 followers Verified	Min. 200,000 followers Verified	Min. 25 retweets
#tweets	7,362,838	16,481,551	3,049,463

Table 8.7: Hints about resource consumptions.

Phase	Computation	Runtime	Resources
Extraction	Machine	10-120 min	<500 nodes
Selection			
Extractiom			
Preprocessing	Machine	30-90 min	1 node
Augmentation			
Clustering		1-5 min	
Popularity	Machine	<2 min	1 node
Deduplication			
Cleaning	Human	1-5 hours	>100 workers
Annotation	Machine	<1 min	1 node
Illustration		5-10 min	
Display		<1 min	

In terms of implementation, Raimond runs partly on a cluster, and partly on a local machine. The cluster gives a huge throughput, but a low latency. The local machine operates the other way around. Therefore, we implemented the operations which require no user intervention on the cluster (in particular the extraction). We run the Clustering step and parts of the Filtering step on the local machine, because these tasks require several rounds of trial and error. We provide hints about the execution times and resource consumptions in Table 8.7 (as Raimond runs on a shared production cluster, its exact runtime depends on the on resources available).

Table 8.8 displays the labels of a few clusters generated by Raimond for the Japan dataset. As in our interface, we ordered the clusters by peak date. We observe that the topics are semantically intelligible. The first cluster describes physical properties of the earthquake. The second one mentions the nuclear plant explosion which followed. Twitter users discuss the impact of the disaster on people and on the environment. Then, they give more details about casualties, and encourage donations.

Table 8.8: Clusters from the Japan Dataset

Keywords	Peak	Size
quake, magnitude, upgraded, usgs, felt	11/03	4,029
nuclear, fukushima, plant, two, explosion	11/03	2,464
axis, moved, shifted, feet, earths	12/03	5,771
people, missing, tsunami, dead, quake	12/03	10,761
toll, death, quake, missing, tsunami	13/03	5,007
effort, help, donate, relief, redcross	13/03	7,414
plant, radiation, nuclear, fukushima, says	15/03	3,062

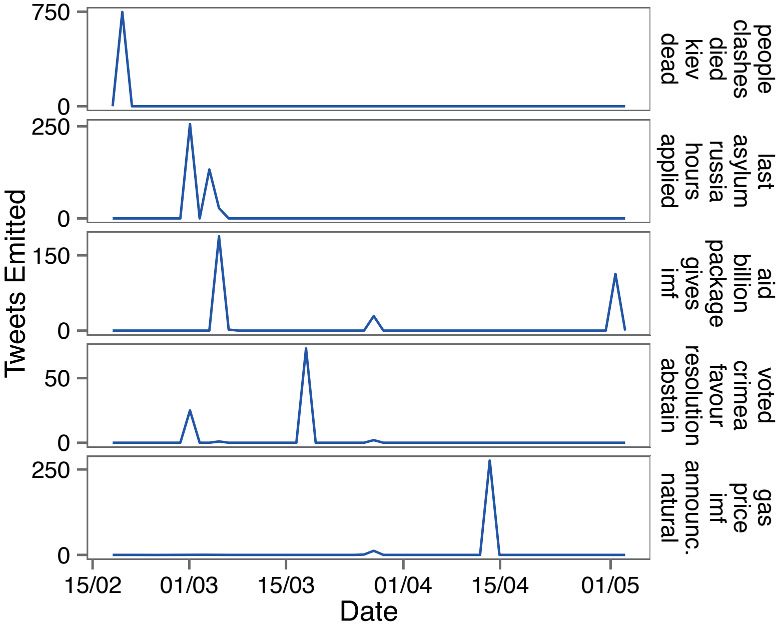


Figure 8.6: Ukraine Data Set.

We show a few clusters created from the Ukraine dataset in Table 8.9. The quantfrags spread across a variety of small topics, such as casualties ("people, clashes, died"), international help ("aid, billion, package"), gas markets ("gas, price, imf") or sanctions ("imposes, officials, sanctions"). We describe the dynamics of the five first clusters in Figure 8.6. To obtain these charts, we tracked the number of quantfrags produced for each cluster. We observe bursts, which last several hours, sometimes days. These bursts actually reflect real events. The first cluster describes the clashes which took place on February 18th and 20th. According to the quantfrags, this was the worse day of violence that Ukraine had known in 70 years. During the followed two weeks, several hundred thousands Ukrainians asked for asylum to Russia and a $15 billion Dollars help package was approved by the European Union. The fourth cluster describes the outcome of the Crimean status referendum, which happened on March 16th. Finally, the last cluster discusses a raise in consumer gas tariffs, requested by the IMF in exchange for a rescue loan.

Table 8.9: Examples of clusters for the Ukraine dataset.

Keywords	Peak Date	Size	Qweet
people, clashes, died, kiev, dead	18/02	1,128	"#Ukraine police say four officers have died in today's riots, 39 have sustained gunshot wounds and more than 100 others have been injured"
last, asylum, russia, hours, applied	01/03	451	"#UKRAINE: 143,000 Ukrainians have asked for asylum in #Russia for last two weeks"
aid, billion, package, gives, imf	05/03	330	"BREAKING: Top official says EU to provide #Ukraine $15 billion aid package in loans and grants"
voted, crimea, favour, resolution, abstained	14/03	765	"#Crimea parliament declares independence from #Ukraine after referendum. Final tally shows 97% voted to join #Russia"
gas, price, imf, announces, natural	13/04	406	"As the IMF announces aid package of $14-18bn for #Ukraine, the Ukrainian PM warns the price paid to Russia for gas will rise 79% from 1 Apr"
imposes, officials, sanctions, entry, russia	28/04	587	"BREAKING NEWS: #EU imposes sanctions on 21 officials from #Russia and #Ukraine over Crimea. More soon..."
donetsk, ballots, region, results, selfdefense	11/05	578	"Preliminary results show 89.7% support of self-rule in #Donetsk region, #referendum election commission says"

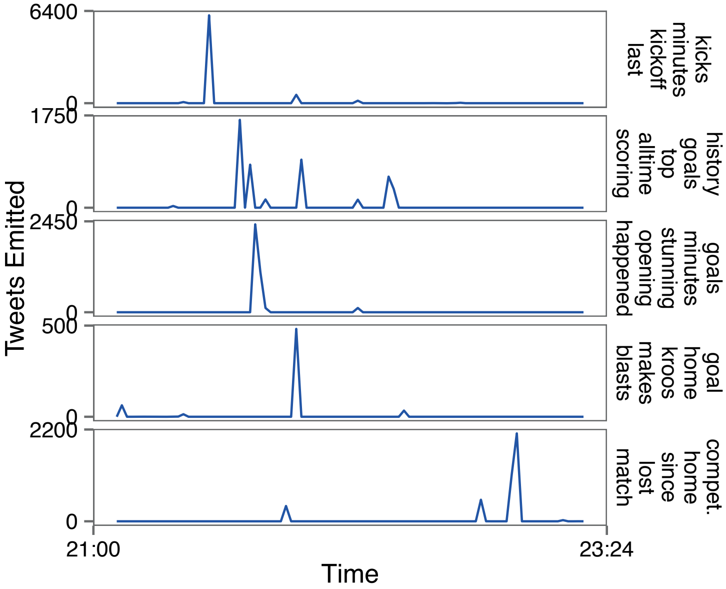


Figure 8.7: Brazil-Germany dataset.

Table 8.10 presents our Brazil-Germany dataset. As opposed to our previous example, the clusters are semantically close to each other - they are all somehow related to scoring goals. We highlight serial quantfrags in the fifth cluster (“goal, home, kroos”): the count of German goals is regulary incremented, finally reaching seven goals. We detail the dynamics of the clusters in Figure 8.7. We see that they appear in short, intense bursts of several minutes. The game starts at 21.00, the first cluster discusses the kick-off. Within the first 30 minutes, the German team scores five goals. This triggers two consecutive clusters, explaining with quantities why the event is “historical” and “stunning”. For instance, Germany is the first country to score 221 goals in a World Cup. With two goals in two minutes, the main attacker, Tony Kroos, has a cluster on his own. The last cluster shows that Brazil had not been defeated at home since 1975.

Table 8.10: Examples of clusters for the BRAvGER dataset.

Keywords	Peak Time	Size	Qweet
reach, semifinals, first, country, consecutive	17:40:22	2,671	"GER is the first team ever to reach four straight #WorldCup semifinals."
kicks, minutes, every, kickoff, less	18:01:22	7,095	"Still more than two hours to go until kick-off... #Copacabana #Brazil "
history, goals, top, alltime, scoring	21:37:26	4,683	"#GER have now scored 221 goals in #WorldCup history, more than any other side and one ahead of #BRA."
goals, minutes, stun, opening, happened	21:47:02	3,535	" That. Just. Happened. Germany stun Brazil with 5 goals in the opening 29 minutes."
goal, home, kroos, makes, blasts	21:51:09	921	"#GER 5 goals in the first 29 minutes!" "6-0... Germany got once and GOAL... #Amazing" "GOAL!!!! ’79 Schurrle blasts home a pitch-perfect pass from Mueller to make it 7-0."
klose, record, now, miroslav, goals	22:57:58	5,331	"#GER’s Mirsolav Klose has a chance to break his record of 15 #WorldCup goals against Brazil."
competitive, home, since, lost, match	23:00:02	4,275	"Entering this match, Brazil had not lost a competitive game on home soil in 14,161 days. Until today.... #BRAvsGER"

5 Crowdsourcing Experiments

In this Section, we evaluate the effectiveness of Raimond's output. We process the three datasets introduced in Section 4, and present the clusters to a set of crowdworkers. We ask them if the quantfrags contain quantitative information, and how *informative* this data is, with a grade between 1 (not informative) and 5 (very informative). As we only have a limited pool of workers, we decided to remove the crowd-based filtering step from the pipeline - to avoid having workers check their own work. Thus, our evaluation is conservative. We evaluated 70 clusters (20 for Ukraine and Brazil-Germany, 30 for Japan), containing between 2 and 75 quantfrags. Each cluster is reviewed by at least two workers.

Figure 8.8a represents the overall distribution of the grades. The neat dominance of the value 4 indicates that most clusters are informative. Nevertheless, Raimond also returns some noise: about a fifth of the clusters have a grade lower than 2.

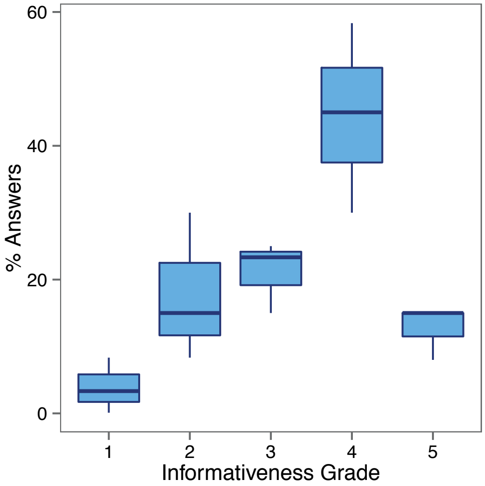
Figure 8.8b shows the grades for each dataset. The Ukraine and Brazil-Germany clusters have good scores. In the Ukraine case, more than 90% of the clusters have at least a grade of 3. Most of the noise comes from the Japan dataset. There are many informative clusters, but there are about as many irrelevant clusters. Further inspection revealed lots of calls for donations, such as:

```
"Txt ASIA to 30333 to donate $5."
"100% donations go to Canadian Red Cross"
"text REDCROSS to 90999 to donate $10 from your phone"
```

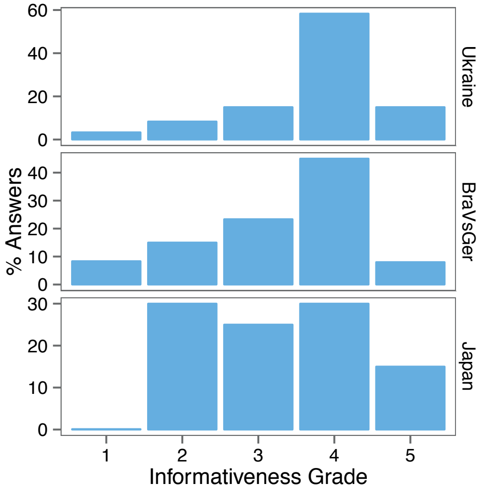
Also, some personalities are so popular that any quantfrag involving them will be retweeted thousands of times:

```
"Justin Bieber donated $1,000,000 to Japan."
"Lady Gaga donated 16 million to Japan"
"Disney made a $2.5 million donation to the Red Cross"
```

Such fragments are difficult to filter programmatically, because they form valid quantfrags and they are extremely popular. To conclude, Raimond does generate useful clusters. Nevertheless, with popular events such Japan earthquake, the diversity and noise in the data justifies the need for human intervention.



(a) Distribution of the grades for every datasets combined.



(b) Grades for each dataset.

Figure 8.8: Crowdsourcing experiment results.

6 Related Work

Studying events on social media has gained considerable interest in the last five years. In particular, catastrophes and emergency situations have attracted lots of attention [43]. The resulting works can be classified in four categories: event detection, event summarization, information extraction and visualization (note that these areas overlap). We describe these works below. There is to our knowledge no previous work on quantitative data extraction from Twitter.

6.1 Event Detection

Sayyadi et al. have published one of the first studies on event detection with social media, based on lexical community detection [84]. Sakaki et al. use microblogging to detect earthquakes and track their location [81]. Popescu and Pennachiotti focus on controversial events, which they recognize with supervised learning [77]. Petrović et al. focus on computational efficiency. They present a scalable algorithm based on Locality-Sensitive Hashing [75].

6.2 Event Summarization

Authors have investigated how to extract key sentences to summarize a text for decades [59]. In 2001, Allan and Khandelwal proposed a method to summarize news coverage. They decompose a main event in sub-events with language models, and describe each sub-topic with a piece of news [4]. Several studies have extended this method to social data with more advanced statistical models. For instance, Chakrabarti et al. use a custom version of Hidden Markov Models to segment the events [18].

6.3 Information Extraction

Extracting structured information about events from social media involves complex NLP methods. One of first the research effort on the topic was presented by Popsecu et al., who use entity extraction to recognize actors [78]. Benson et al. go one step further, as they infer structured records about entertainment events from Twitter [12]. Imran et al. combine several classifiers and a sequence labelling algorithm to extract structured information about disasters [44]. These approaches are generalized by Ritter et all, who introduce a method to analyze events in open domains. They present a pipeline, somehow similar to Raimond, which extracts named entities, event phrases, calendar dates and event type. Their pipeline combines custom NLP tools and unsupervised learning [79].

6.4 Event Visualization

Finally, several authors have studied how to create visual dashboards from Twitter to describe events. Diakopoulos et al. combine raw data, automatically-generated statistics (such as sentiment or relevance) and timelines to help journalists [25]. Marcus et al. propose a similar system, with geographical information and peak detection [60]. Alonso and Shiells introduce a display based on multiple timelines, and illustrate their method with sports events [6].

6.5 Social Media Analysis

A number of studies resemble ours by their methods, but target other problems. Alonso et al. study to what extent crowdsourcing can be used to assess the interestingness of tweets [5]. For instance, NIFTY by Suen et al. is also an information extraction pipeline based on Twitter and unsupervised learning. However, it focuses meme-tracking [89]. More generally, news processing is an active related domain of research [3, 76].

7 Summary

In this chapter, we introduced a semi-automatic solution to explore non structured data. We focused on quantitative information, enclosed within microblogs. We presented Raimond, a pipeline to extract, process and present this information. The central concept behind our approach is the quantfrag, We illustrated the concept with a number of examples. We presented how to extract those with the help of NLP techniques, how to organize them with clustering, and how to clean them with a hybrid automatic/crowdsourcing approach. Finally, we showcased quantfrags about three real events. We described their semantics, their dynamics and evaluated their content. We showed that semi-automated exploration of plain text document is indeed feasible, opening the way for exciting developments.

Chapter 9

Conclusion

1 The Big Picture

In the first chapter of this thesis, we presented three fields of research concerned with the exploration of large databases : machine learning, query languages and visualization. We showed that these fields solve the same problem. Yet, each has taken a life on its own, with its own abstractions, its own tools and its own body of experts. A few software packages combine them all. Popular examples are R, SAS or Matlab. But those target precision and exhaustivity, not exploration. They provide all the "bricks" necessary to build exploration pipelines, but they completely rely on the users' patience and expertise to assemble them. This paradigm leaves space for imagination and flexibility, but it cannot deliver quick insights.

Our proposal is to reverse this logic. Each assistant is based on a tightly coupled, pre-packaged combination of queries, machine learning and visualizations. Admittedly, our systems are less expressive than general purpose statistical software. But they are much easier to use. They only require little knowledge about the dataset or about data processing in general. Furthermore, they are faster. Since their architectures are fixed, we can organize their computations efficiently. We can stage execution times, or share intermediate results.

A major challenge was to define exploration tasks that would be general enough to be useful, but narrow enough to be solved in a reasonable amount of time. To tackle it, we surveyed both the database and the machine learning literatures and identified synergies. We discovered that feature selection and data warehousing deal with similar problems but in different contexts. This led us to build Claude. We also realized that cluster analysis could help explorers issue Select-Project-Join statements. This inspired us for Blaeu. But practice also helped us discover use cases that had never been studied before. For instance, Ziggy solves the inverse of the exploration problem: instead of considering that users seek queries,

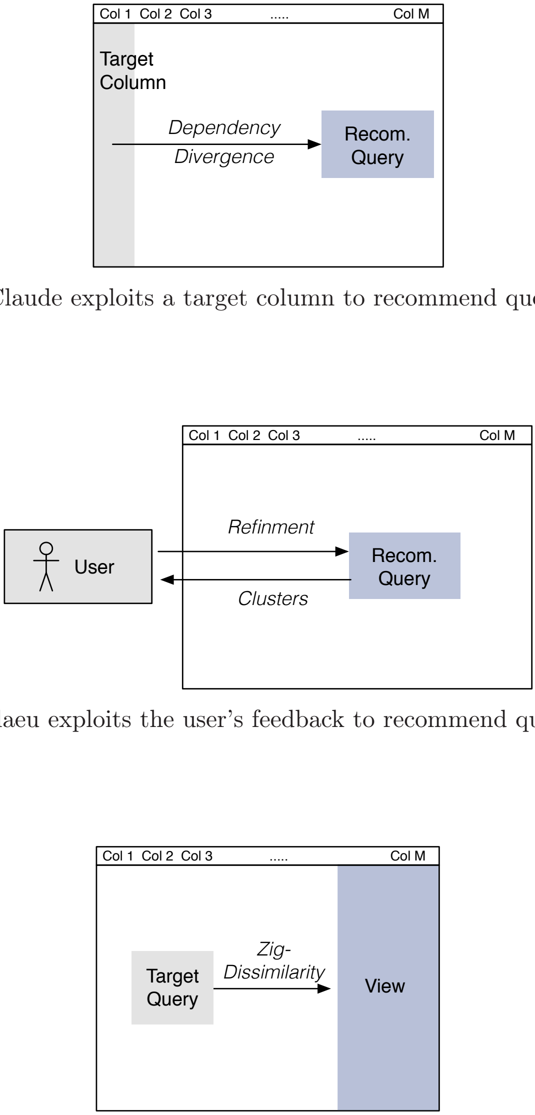
System	Type Data	Input	Output	Type Learning
Claude	Table	Target Column	Views, POIs	Supervised
Blaeu	Table	User feedback	Data Maps	Unsupervised
Ziggy	Table	Target query	Views	Supervised
Raimond	Text	Filters	Timelines	Unsupervised

Table 9.1: Concepts and methods used by each assistant.

it assumes that they already have them, but they do not how interesting they are. This problem was dictated by usage, during the development of Blaeu. Also, Raimond helps users extract and explore quantitative information in tweets. This work was carried out during an internship at Microsoft, and indeed it was inspired by practical and commercial needs. We summarize these scenarios and the main ideas behind each system in Table 9.1 and Figure 9.1.

Once we identified our use cases, we established that existing machine learning algorithms did not suffice to solve them. Indeed, the requirements of data exploration are different from those of classic statistical analysis. For a start, accuracy is not the first citizen. Instead, users value speed and diversity. A quick, approximate answer is better than a slow, exact one. And users may prefer a dozen alternative solutions to an optimal one. Furthermore, interpretability matters. The aim of exploration is to discover the data, not to make predictions. Therefore, users may prefer simple structures such as boxes and decision trees, rather than complex, high-dimension probabilistic models. Finally, our algorithms must be robust. They must cope with mixed types, noise, missing values and sparsity. To address these requirements, we had to transform existing techniques (we did so with Blaeu and Raimond) or invent our own (as we did with Claude and Ziggy).

The last challenge we faced was to evaluate the quality of our systems. For a start, we had to check whether the assumptions we made about the users held. To do so, we introduced practical use cases. Among others, we discussed Hollywood movies, crime in large cities and innovation in European countries. The hope was that our systems' findings would surprise the reader as much they did us. Then, we had to ensure that our systems made good recommendations within these frameworks. We did so with public datasets, already annotated by experts. We also used synthetic databases in which we controlled the occurrence of statistical patterns. In all cases, we found out that our systems could produce quick, accurate and useful results.



(a) Claude exploits a target column to recommend queries.

(b) Blaeu exploits the user's feedback to recommend queries.

(c) Ziggy exploits a target query to recommend views.

Figure 9.1: Overview of Claude, Blaeu and Ziggy. Each assistant exploits a hint from the user (colored in grey) from which they make recommendations (colored in blue).

One danger with semi-automated solutions is that the users may blindly trust the system's recommendations and accept them as ground truth. This would be a mistake. No model is perfect, especially not one that represents human users. Furthermore, our assistants rely on heuristics and approximations to beat huge search spaces. Hence, they are free of neither false positives nor false negatives. The correct way to use them is to consider them as *hypothesis generators*. They produce rules about the data, which hopefully the user did not know before. But those rules must pass the test of common sense and critical appreciation. More generally, the necessity to inspect and verify statistical models is one more argument in favor of "white box", interpretable machine learning. Data analytics should be transparent and open for skeptical inspection, and our work made a step in this direction.

At the time of writing this thesis, many considered data scientist as "the sexiest job of the 21st century" (the Harvard Business Review [92]). Large Internet firms were initiating gigantic AI projects and many science graduates were steering their careers towards data analytics. But it should not be so. At the risk of being self-destructive, we believe that analyzing data should not require a doctoral degree. It should be quick, easy and accessible to users with all kind of backgrounds. In fact, it should be commonplace, as commonplace as using a smartphone or a search engine. This thesis opened the way for a new generation of user-friendly tools to discover large databases. It showed that achieving this vision is indeed possible.

2 Future Research

The road for more intelligent interfaces lies wide open. We now present opportunities for future research.

2.1 Human-Centered Analytics

Our four systems rely on user models. In each case, we make assumptions about what interests the users, and develop tools to help them within these frameworks. Our assumptions are grounded in statistical theory: we assume that if a pattern (e.g., a correlation, a cluster) is interesting for a data miner, then it will be interesting for a data explorer too. But more work is necessary to understand what real users need and how they interact with data. We can do so in two ways. We can collect this information *offline*, with user studies and benchmarks. We can also gather it *online*, by incorporating feedback collection mechanisms directly in our solutions. Let us consider these options in turn.

2.1.1 User Studies, Benchmarks

The first step to ameliorate our assistants is to conduct user studies with a broad base of real users. So far, we mostly focused on issues related to back-end engineering. In the future, we will study how humans interact with data exploration tools, and we will use our findings to build more practical, intuitive interfaces.

In general, data exploration has only interested the database community for a few years [41]. Although a dozen papers were submitted about this problem, it is not yet clear how to evaluate data exploration systems in a standardized, rigorous, and exhaustive way. In fact, there exists no benchmark for data exploration. But can such a benchmark really exist? On one hand, data exploration is an open-ended, subjective process. On the other hands, scientists from other fields have successfully managed to standardize tasks which seemed equally difficult (e.g., the TREC benchmark for text mining [98]). As authors publish more and more studies about data exploration, the need for a standardized set of metrics to evaluate its success is getting pressing.

2.1.2 Human-In-The-Loop Systems

Another way to narrow the gap between users and systems is to incorporate the users' preferences directly in the mining algorithms. We can achieve this by studying the users' past interactions: where they clicked, what they selected and what they inspected. Based on those hints, we can infer what type of knowledge interests them, and make appropriate recommendations. For instance, we envision an adaptive system that considers which views the user has inspected before, and recommends either "different" or "similar" visualizations. To implement this function, we could exploit the intrinsic features of the views or set up relevance feedback mechanisms. Thus, we could chose the variables to show, personalize the type of charts to represent them or set an optimal level of detail to display.

An other promising research direction is to involve the users in the execution of the exploration algorithm. If we know that a query or a mining algorithm needs to explore a large search space, we can interrupt it in "mid-flight", ask the user whether they are interested in the portion of the space being inspected, and steer the execution accordingly.

2.1.3 Alternative Devices

In this thesis, we focused on traditional desktop or laptop-based interfaces. But alternatives exist, involving for instance, tablets, large touch screens, motion capture or even virtual reality headsets. The questions of how to explore data with these devices and whether this is actually desirable is still open.

2.2 More Exploration Assistants

In this thesis, we only covered a limited number of statistical methods and exploration scenarios. We now discuss how to go further.

2.2.1 Other Statistical Methods

A promising research direction would be to target the same exploration scenarios as those described in this thesis, but involve other statistical models than those used so far. Take Claude, which recommends OLAP views. Instead of using information theory, we could exploit on the recent and quickly expanding field of *causality detection* [65]. For Blaeu, we could incorporate other data sources: for instance, we could exploit the meta-data associated with the database (e.g., textual description of the columns) or knowledge bases to make better groups of columns. Also, we could borrow concepts from social network analysis, such as community detection or edge recommendation, to explore the graph formed by the correlations between the columns.

2.2.2 New Exploration Tasks

Clearly, our four assistant are far from covering all the possible tasks of data exploration. Many scenarios remain to be solved. To find those, we could survey companies and users and understand how to help them. Alternatively, we could study the data mining tasks that we have not yet covered. For instance, this thesis has made no mention of *outlier detection*, *semi-supervised learning* or *frequent patterns*. Yet, those are important problems in data mining research. They could probably find applications in data exploration too.

A significant research effort is also necessary to deal with non tabular data. In Chapter 8, we introduced a semi-automated exploration tool to analyze text. But we only focused on a specific use case. Plenty of other problems arise with this natural language, involving for instance exploring the relationship between people, extracting and plotting references to places, or identifying topics and their evolution. Furthermore, many other data types remain to be tackled. We have not discussed images, videos, genetic sequences and the combination of all of those. We have not mentioned multi-database settings either.

We believe that data exploration has a bright future ahead. Eventually, our task will only truly be complete when casual users will be able to process and understand Terabytes of noisy, non structured, mixed-type data within micro-seconds on a laptop. The challenge is open.

Bibliography

[1] Azza Abouzied, Joseph Hellerstein, and Avi Silberschatz. Dataplay: interactive tweaking and example-driven correction of graphical database queries. In *USIT*, pages 207–218. ACM, 2012.

[2] Charu C Aggarwal, Joel L Wolf, Philip S Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *SIGMOD*, 1999.

[3] Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric Xing, Alexander J Smola, and Choon Hui Teo. Unified analysis of streaming news. In *Proc. WWW*, pages 267–276, 2011.

[4] James Allan, Rahul Gupta, and Vikas Khandelwal. Temporal summaries of new topics. In *Proc. SIGIR*, pages 10–18. ACM, 2001.

[5] Omar Alonso, Catherine C Marshall, and Marc Najork. Are some tweets more interesting than others? #hardquestion. In *HCIR*, page 2. ACM, 2013.

[6] Omar Alonso and K Shiells. Timelines as summaries of popular scheduled events. In *Proc. WWW*, pages 1037–1044, 2013.

[7] Chris Anderson. The end of theory: The data deluge makes the scientific method obsolete. *Wired*, June 23rd, 2008.

[8] Erik Brynjolfsson Andrew McAfee. Big data: The management revolution. *Harvard Business Review*, Oct. 2012.

[9] Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Detecting outlying properties of exceptional objects. *ACM TODS*, 2009.

[10] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[11] Jan Beirlant, Edward J Dudewicz, László Györfi, and Edward C Van der Meulen. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997.

[12] Edward Benson, Aria Haghighi, and Regina Barzilay. Event discovery in social media feeds. In *Proc. ACL*, pages 389–398. Association for Computational Linguistics, 2011.

[13] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *ICDT*. 1999.

[14] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[15] Angela Bonifati, Radu Ciucanu, and Slawomir Staworko. Interactive inference of join queries. In *EDBT*, pages 451–462, 2014.

[16] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees.* CRC press, 1984.

[17] Francesco Camastra. Data dimensionality estimation methods: a survey. *Pattern recognition*, pages 2945–2954, 2003.

[18] Deepayan Chakrabarti and Kunal Punera. Event summarization using tweets. In *Proc. ICWSM*, pages 66–73. AAAI Press, 2011.

[19] C-I Chang, Yingzi Du, J Wang, S-M Guo, and PD Thouin. Survey and comparative analysis of entropy and relative entropy thresholding techniques. In *IEEE Proc. Vision, Image and Signal Processing*, pages 837–850, 2006.

[20] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, 2009.

[21] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26:65–74.

[22] Jacob Cohen. *Statistical power analysis for the behavioral sciences.* Lawrence Erlbaum Associates, 1977.

[23] Thomas M Cover and Joy A Thomas. *Elements of information theory.* John Wiley & Sons, 2012.

[24] Debabrata Dash, Jun Rao, Nimrod Megiddo, Anastasia Ailamaki, and Guy Lohman. Dynamic faceted search for discovery-driven analysis. In *ACM CIKM*, pages 3–12, 2008.

[25] Nicholas Diakopoulos, Mor Naaman, and Funda Kivran-Swaine. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *Proc. VAST*, pages 115–122. IEEE, 2010.

[26] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proc. SIGMOD*, 2014.

[27] Guozhu Dong, Jiawei Han, Joyce Lam, Jian Pei, and Ke Wang. Mining multi-dimensional constrained gradients in data cubes. In *VLDB*, pages 321–330, 2001.

[28] Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. Mining outlying aspects on numeric data. *Data Mining and Knowl. Discovery*, pages 1–36, 2014.

[29] Wouter Duivesteijn, Arno Knobbe, Ad Feelders, and Matthijs van Leeuwen. Subgroup discovery meets bayesian networks–an exceptional model mining approach. In *IEEE ICDM*, pages 158–167, 2010.

[30] Niklas Elmqvist, Pierre Dragicevic, and Jean-Daniel Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *TVCG*, pages 1539–1148, 2008.

[31] Ronald A Fisher. Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population. *Biometrika*, 1915.

[32] JH Friedman and JW Tukey. A projection pursuit algorithm for exploratory data analysis. In *IEEE TC*, page 149, 1974.

[33] Parke Godfrey, Ryan Shipley, and Jarek Gryz. Algorithms and analyses for maximal vector computation. *Proc. VLDB*, 2007.

[34] D Graham-Rowe, D Goldston, C Doctorow, M Waldrop, C Lynch, F Frankel, R Reid, S Nelson, D Howe, SY Rhee, et al. Big data: science in the petabyte era. *Nature*, 455, 09 2008.

[35] Stephan Günnemann, Ines Färber, Emmanuel Müller, Ira Assent, and Thomas Seidl. External evaluation measures for subspace clustering. In *CIKM*, 2011.

[36] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, pages 1157–1182, 2003.

[37] Tim Harford. Big data: are we making a big mistake? *Financial Times Magazine*, March 28, 2014.

[38] Larry V Hedges and Ingram Olkin. *Statistical method for meta-analysis.* Academic press, 1985.

[39] Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Proc. ICDM*, pages 541–544. IEEE, 2003.

[40] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li. Toward scalable systems for big data analytics: A technology tutorial. *Access, IEEE*, 2:652–687, 2014.

[41] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *Proc. SIGMOD*, pages 277–281, 2015.

[42] Tomasz Imieliński, Leonid Khachiyan, and Amin Abdulghani. Cube-grades: Generalizing association rules. *Data Mining and Knowledge Discovery*, pages 219–257, 2002.

[43] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Processing social media messages in mass emergency: A survey. In *CoRR*. arXiv preprint: 1407.7071, 2014.

[44] Muhammad Imran, S Elbassuoni, and Carlos Castillo. Practical extraction of disaster-relevant information from social media. In *Proc. WWW*, pages 1021–1024, 2013.

[45] John PA Ioannidis. Why most published research findings are false. *Chance*, pages 40–47, 2005.

[46] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis.* John Wiley & Sons, 1990.

[47] D.A. Keim. Visual techniques for exploring databases. In *Tutoriak Notes, KDD*, 1997.

[48] Fabian Keller, Emmanuel Muller, and Klemens Bohm. Hics: high contrast subspaces for density-based outlier ranking. In *ICDE 2012*, 2012.

[49] Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. *AAAI Advances in Knowledge Discovery and Data Mining*, pages 249–271, 1996.

[50] Edwin M Knorr and Raymond T Ng. Finding intensional knowledge of distance-based outliers. In *Proc. VLDB*, pages 211–222, 1999.

[51] H-P Kriegel, Peer Kroger, Matthias Renz, and Sebastian Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *ICDM*, 2005.

[52] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *TKDD*, 2009.

[53] Peter Langfelder, Bin Zhang, and Steve Horvath. Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for r. *Bioinformatics*, 2008.

[54] David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. The parable of google flu: traps in big data analysis. *Science*, 343(14 March), 2014.

[55] Fei Li and HV Jagadish. Constructing an interactive natural language interface for relational databases. In *Proc. VLDB*, pages 73–84, 2014.

[56] James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and Natural-Language description of nonparametric regression models. In *AAAI*, 2014.

[57] Elsa Loekito and James Bailey. Mining influential attributes that capture class and group contrast behaviour. In *Proc. CIKM*, pages 971–980, 2008.

[58] Steve Lohr. The age of big data. *The New York Times*, February 12, 2012.

[59] HP Luhn. The automatic creation of literature abstracts. In *IBM Journal of research and development*, pages 159–165, 1958.

[60] Adam Marcus, MS Bernstein, and Osama Badar. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Proc. CHI*, pages 227–236. ACM, 2011.

[61] Gary Marcus and Ernest Davis. Eight (no, nine!) problems with big data. *The New York Times*, Apr. 6, 2014.

[62] Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[63] Marina Meilă. Comparing clusterings – an information based distance. *Journal of Multivariate Analysis*, 2007.

[64] George A Miller. Wordnet: a lexical database for english. In *CACM*, volume 38, pages 39–41. ACM, 1995.

[65] Joris M Mooij, Jonas Peters, Dominik Janzing, Jakob Zscheischler, and Bernhard Schölkopf. Distinguishing cause from effect using observational data: methods and benchmarks. *arXiv preprint arXiv:1412.3773*, 2014.

[66] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *Proc. VLDB*, 2009.

[67] Emmanuel Muller, Stephan Gunnemann, Ines Farber, and Thomas Seidl. Discovering multiple clustering solutions: Grouping objects in different views of the data. In *ICDE*, 2012.

[68] Emmanuel Müller, Hoang Vu Nguyen, Fabian Keller, Klemens Böhm, and Jilles Vreeken. Cmi: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection. In *ICDM*, 2013.

[69] Arnab Nandi and HV Jagadish. Guided interaction: Rethinking the query-result paradigm. In *Proc. VLDB*, pages 1466–1469, 2011.

[70] Hoang Vu Nguyen, E Muller, and K Bohm. 4s: Scalable subspace search scheme overcoming traditional apriori processing. In *IEEE Big Data*, pages 359–367, 2013.

[71] Aditya Parameswaran, Neoklis Polyzotis, and Hector Garcia-Molina. Seedb: Visualizing database queries efficiently. *Proc. VLDB*, 2013.

[72] Jagdish K Patel and Campbell B Read. *Handbook of the normal distribution*, pages 204–205. CRC Press, 1996.

[73] Philippe Pébay. Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. *Tech. report, Sandia National Laboratories*, 2008.

[74] Hanchuan Peng, Fulmi Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. In *IEEE PAMI*, pages 1226–1238, 2005.

[75] S Petrović, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *NAACL*, pages 181–189. Association for Computational Linguistics, 2010.

[76] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proc. WWW*, pages 91–100, 2008.

[77] Ana-Maria Popescu and Marco Pennacchiotti. Detecting controversial events from twitter. In *Proc. CIKM*, page 1873. ACM, 2010.

[78] Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. Extracting events and event descriptions from Twitter. In *Proc. WWW*, page 105, 2011.

[79] Alan Ritter, Oren Etzioni, and Sam Clark. Open domain event extraction from twitter. In *KDD*, page 1104. ACM, 2012.

[80] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, pages 53–65, 1987.

[81] Takeshi Sakaki, M Okazaki, and Y Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proc. WWW*, pages 851–860, 2010.

[82] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. *EDBT*, 1998.

[83] Sunita Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, pages 307–316, 2000.

[84] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event Detection and Tracking in Social Streams. In *Proc. ICWSM*, pages 311–314. AAAI Press, 2009.

[85] Thibault Sellam and Martin Kersten. Meet charles, big data query advisor. *CIDR*, 2013.

[86] Lefteris Sidirourgos, Martin L Kersten, and Peter A Boncz. Sciborq: Scientific data management with bounds on runtime and quality. In *CIDR*, 2011.

[87] Robert R Sokal. A statistical method for evaluating systematic relationships. In *U. Kansas Scientific Bulletin*, volume 38, pages 1409–1438, 1958.

[88] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *TVCG*, 2002.

[89] Caroline Suen, Sandy Huang, Chantat Eksombatchai, Rok Sosic, and Jure Leskovec. Nifty: a system for large scale information flow tracking and clustering. In *Proc. WWW*, pages 1237–1248, 2013.

[90] Deborah F Swayne, Duncan Temple Lang, Andreas Buja, and Dianne Cook. Ggobi: evolving from xgobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, pages 423–444, 2003.

[91] Andrada Tatu, Georgia Albuquerque, Martin Eisemann, Peter Bak, Holger Theisel, Marcus Magnor, and Daniel Keim. Automated analytical methods to support visual exploration of high-dimensional data. In *IEEE TVCG*, pages 584–597, 2011.

[92] DJ Patil Thomas Davenport. Data scientist: The sexiest job of the 21st century. *Harvard Business Review*, Oct. 2012.

[93] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society*, pages 411–423, 2001.

[94] Unsigned. The data deluge. *The Economist*, Feb. 25th, 2010.

[95] Matthijs van Leeuwen and Arno Knobbe. Non-redundant subgroup discovery in large and complex data. In *ECML PKDD*, pages 459–474. 2011.

[96] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes. Compression picks item sets that matter. In *PKDD*, pages 585–592. 2006.

[97] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. Seedb: Efficient data-driven visualization recommendations to support visual analytics. *Proc. VLDB*, pages 2182–2193, 2015.

[98] Ellen M Voorhees and Donna Harman. Overview of trec 2001. In *Trec*, 2001.

[99] Jilles Vreeken, Matthijs Van Leeuwen, and Arno Siebes. Characterising the difference. In *Proc. SIGKDD*, pages 765–774, 2007.

[100] Joseph Walker. Meet the new boss: Big data. *Wall Street Journal*, Sept. 20, 2012.

[101] Matthew O Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In *Proceedings of the Conference on Visualization'94*, pages 326–333. IEEE Computer Society Press, 1994.

[102] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer, 2013.

[103] Geoffrey I Webb, Shane Butler, and Douglas Newlands. On detecting differences between groups. In *Proc. SIGKDD*, pages 256–265, 2003.

[104] Leland Wilkinson, Anushka Anand, and Robert L Grossman. Graph-theoretic scagnostics. In *IEEE Infovis*, page 21, 2005.

[105] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *PKDD*, pages 78–87. 1997.

[106] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 2008.

[107] Yan Xie and Philip S Yu. Max-clique: a top-down graph-based approach to frequent pattern mining. In *IEEE ICDM*, pages 1139–1144, 2010.

[108] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In *VLDB*, pages 709–720, 2005.

[109] J Yang, M Ward, and Elke A Rundensteiner. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In *InfoVis*, 2002.

[110] Ji Zhang and Hai Wang. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*, pages 333–355, 2006.

[111] Mu Zhu and Ali Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, pages 918–930, 2006.

[112] Moshe M. Zloof. QBE/OBE: a language for office and business automation. *Computer*, 1981.

Summary

Database management systems (DBMSs) rely on an implicit pact. They provide quick and correct answers, in exchange for precise, complete and syntactically correct questions. In most cases, database users express those questions with a *query language*, such as SQL. The practical and formal benefits of those languages are well established. But they are also demanding, and rarely forgiving. Mastering them can take days, possibly weeks. Furthermore, query languages require precision. Database users must specify exactly what they want and where to get it. Hence, they need well-defined requirements, along with a solid knowledge of the data.

There exists an important class of scenarios in which this model reaches its limit: *data exploration*. Data explorers interrogate a database to discover its content. Their aim is to get an overview of the data and discover interesting new facts. They have little to no knowledge of the data, and their requirements are often vague and abstract. How can such users match the precision required by a query language? Typically, they must resort to trial and error. Yet, the success of this approach depends entirely on patience and luck. Manual effort may suffice to process small data sets (containing, e.g., a dozen columns and a few hundred tuples), but it can prove tedious and error-prone for larger ones.

This thesis presents four *assistants*, to help users compose and refine interesting new queries: *Claude*, *Blaeu*, *Ziggy* and *Raimond*. These systems operate in a semi-automatic fashion: they infer recommendations from input parameters and interaction. Each assistant focuses on a specific exploration task. Claude helps users analyze data warehouses, by highlighting the combinations of variables which influence a predefined measure of interest. Blaeu helps users build and refine queries, by allowing them to select and project clusters of tuples. Ziggy is a tuple characterization engine: its aim is to show what makes a selection of tuples unique, by highlighting the differences between those and the rest of the database. Finally, Raimond is an attempt to generalize semi-automatic exploration to text data, inspired by an industrial use case.

For each system, we present a user model, that is, a formalized set of assumptions about the users' goals. To express these models, we rely heavily

on information theory and statistics. We then present practical methods to make recommendations. We either adapt existing algorithms from the machine learning literature or present our own. Next, we validate our approaches with experiments. We present use cases in which our systems led to discoveries. We benchmark the quality and the robustness of our assistants' output, using both real and synthetic datasets. Finally, we demonstrate the scalability of our algorithms.

In conclusion, our assistants can quickly deliver informative query recommendations from a wide range of data, thereby fulfilling their promises. Nevertheless, the field of data exploration is immense, far greater than the four scenarios discussed in this study. We close this thesis with a program for future research, as well as early-stage ideas for future solutions.

Samenvatting

Databasesystemen sluiten een impliciete overeenkomst met hun gebruikers. Ze geven snelle en correcte antwoorden, in ruil voor precies geformuleerde vragen. In de meeste gevallen stellen database gebruikers deze vragen met een zogenaamde query-taal, zoals SQL. De praktische en formele voordelen van deze talen zijn nauwkeurig vastgesteld. Maar ze zijn ook veeleisend, en alles behalve vergevend. Gebruikers moeten dagen of zelfs weken besteden voordat ze zo'n taal onder de knie hebben. Verder vereisen deze talen precisie. Gebruikers moeten exact specificeren wat ze willen en waar het vandaan moet komen. Het is daarom noodzakelijk dat gebruikers precies weten wat ze nodig hebben, en wat de structuur van de data is.

Bij data-exploratie is dit model slecht toepasbaar. Hier wil de gebruiker de data verkennen, en interessante informatie in de database ontdekken. In dit scenario weet de gebruiker niks over de database, en hun vragen zijn vaak onduidelijk of abstract. Hoe kunnen zulke gebruikers de vereiste precisie van een query-taal bereiken? Gewoonlijk moeten ze eerst met vallen en opstaan kennis vergaren over de database. Hun succes is volledig gebaseerd op geduld en geluk. Voor kleine datasets kan dit genoeg zijn, maar voor grote datasets wordt dit al snel foutgevoelig monnikenwerk.

Deze scriptie presenteert vier assistenten die zulke gebruikers helpen op hun zoektocht: Claude, Blaeu, Ziggy en Raimond. Deze systemen helpen de gebruiker om makkelijker de data te onderzoeken door gebruik te maken van semi-automatische exploratie. Elke assistent focust op een specifieke exploratietaak. Claude helpt gebruikers data warehouses te analyzeren, door te vertellen welke data invloed op een vooraf gedefiniëerd effect hebben. Blaeu helpt gebruikers om queries te maken en verfijnen, door de gebruiker interessante clusters in de data te tonen. Ziggy laat gebruikers zien wat een bepaalde selectie van tupels interessant maakt door de verschillen met de rest van de dataset weer te geven. Tenslotte proberen we met Raimond semi-automatische exploratie toe te passen op textuele data.

Voor elk systeem presenteren we een gebruikersmodel; dat is een formele set van assumpties die we maken over de doelen van de gebruiker. We stellen deze doelen op met gebruik van informatietheorie en statistiek. We presenteren dan praktische methoden om aanbevelingen te doen. Hiervoor passen we bestaande algoritmes van de machine learning literatuur aan, of gebruiken we een eigen algoritme. Daarna valideren we de systemen met experimenten. We presenteren use cases waarin ons systeem helpt ontdekkingen te maken, en meten de kwaliteit en robuustheid van de aanbevelingen van de systemen. We maken hiervoor gebruik van zowel echte als synthetische datasets. Als laatste demonstreren we de schaalbaarheid van onze algoritmes.

Onze assistenten kunnen snel informatieve aanbevelingen geven over een wijde selectie van data. Hiermee helpen we de gebruiker hun datasets beter te begrijpen en helpen we de gebruiker om antwoorden op hun vragen te vinden. Desalniettemin is het veld van data-exploratie immens; veel groter dan de vier scenarios die we in deze studie hebben laten zien. We sluiten deze scriptie af met suggesties voor toekomstig onderzoek, samen met onuitgewerkte ideeën voor toekomstige oplossingen.

Publications

This section lists the articles on which this thesis is based. The doctoral candidate has carried out the research, implementation, experiments and redaction effort behind the majority of these reports [3, 4, 6, 5, 7], under the supervision of prof. dr. Martin Kersten. In addition, the candidate received the guidance of dr. Omar Alonso (Microsoft Corp.) for the work related to Raimond [1]. Prof. dr. Emmanuel Müller (Hasso Plattner Institute) provided significant research and redactional advise for the Claude project [8]. Finally, Robin Cijvat and Richard Koopmanschap (MonetDB Solutions) have built of a commercial implementation of Blaeu, exhibited at VLDB [2].

[1] Thibault Sellam and Omar Alonso. Raimond: Quantitative data extraction from twitter to describe events. In *Proc. ICWE*, pages 251–268, 2015.

[2] Thibault Sellam, Robin Cijvat, Richard Koopmanschap, and Martin Kersten. Blaeu: Mapping and navigating large tables with cluster analysis [demo paper]. In *Proc. VLDB*, pages 1477–1480, 2016.

[3] Thibault Sellam and Martin Kersten. Meet charles, big data query advisor. In *Proc. CIDR*, 2013.

[4] Thibault Sellam and Martin Kersten. Cluster-driven navigation of the query space. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1118–1131, 2016.

[5] Thibault Sellam and Martin Kersten. Fast, explainable view detection to characterize exploration queries. In *Proc. SSDBM*, 2016.

[6] Thibault Sellam and Martin Kersten. Have a chat with clustine, a conversational engine to query large tables. In *Proc. Workshop on Human-In-the-Loop Data Analytics*, 2016.

[7] Thibault Sellam and Martin Kersten. Ziggy: Characterizing query results for data explorers [demo paper]. In *Proc. VLDB*, pages 1473–1476, 2016.

[8] Thibault Sellam, Emmanuel Müller, and Martin Kersten. Semi-automated exploration of data warehouses. In *Proc. CIKM*, pages 1321–1330, 2015.

SIKS Dissertations

2009

2009-01 Rasa Jurgelenaite (RUN), *Symmetric Causal Independence Models.*
2009-02 Willem Robert van Hage (VU), *Evaluating Ontology-Alignment Techniques.*
2009-03 Hans Stol (UvT), *A Framework for Evidence-based Policy Making Using IT.*
2009-04 Josephine Nabukenya (RUN), *Improving the Quality of Organisational Policy Making using Collaboration Engineering.*
2009-05 Sietse Overbeek (RUN), *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality.*
2009-06 Muhammad Subianto (UU), *Understanding Classification.*
2009-07 Ronald Poppe (UT), *Discriminative Vision-Based Recovery and Recognition of Human Motion.*
2009-08 Volker Nannen (VU), *Evolutionary Agent-Based Policy Analysis in Dynamic Environments.*
2009-09 Benjamin Kanagwa (RUN), *Design, Discovery and Construction of Service-oriented Systems.*
2009-10 Jan Wielemaker (UVA), *Logic programming for knowledge-intensive interactive applications.*
2009-11 Alexander Boer (UVA), *Legal Theory, Sources of Law & the Semantic Web.*
2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin), *Operating Guidelines for Services.*
2009-13 Steven de Jong (UM), *Fairness in Multi-Agent Systems.*
2009-14 Maksym Korotkiy (VU), *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA).*
2009-15 Rinke Hoekstra (UVA), *Ontology Representation - Design Patterns and Ontologies that Make Sense.*
2009-16 Fritz Reul (UvT), *New Architectures in Computer Chess.*
2009-17 Laurens van der Maaten (UvT), *Feature Extraction from Visual Data.*
2009-18 Fabian Groffen (CWI), *Armada, An Evolving Database System.*
2009-19 Valentin Robu (CWI), *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets.*
2009-20 Bob van der Vecht (UU), *Adjustable Autonomy: Controling Influences on Decision Making.*
2009-21 Stijn Vanderlooy (UM), *Ranking and Reliable Classification.*
2009-22 Pavel Serdyukov (UT), *Search For Expertise: Going beyond direct evidence.*
2009-23 Peter Hofgesang (VU), *Modelling Web Usage in a Changing Environment.*
2009-24 Annerieke Heuvelink (VUA), *Cognitive Models for Training Simulations.*
2009-25 Alex van Ballegooij (CWI), *"RAM: Array Database Management through Relational Mapping".*
2009-26 Fernando Koch (UU), *An Agent-Based Model for the Development of Intelligent Mobile Services.*
2009-27 Christian Glahn (OU), *Contextual Support of social Engagement and Reflection on the Web.*
2009-28 Sander Evers (UT), *Sensor Data Management with Probabilistic Models.*
2009-29 Stanislav Pokraev (UT), *Model-Driven Semantic Integration of Service-Oriented Applications.*

2009-30 Marcin Zukowski (CWI), *Balancing vectorized query execution with bandwidth-optimized storage.*
2009-31 Sofiya Katrenko (UVA), *A Closer Look at Learning Relations from Text.*
2009-32 Rik Farenhorst (VU) and Remco de Boer (VU), *Architectural Knowledge Management: Supporting Architects and Auditors.*
2009-33 Khiet Truong (UT), *How Does Real Affect Affect Affect Recognition In Speech?.*
2009-34 Inge van de Weerd (UU), *Advancing in Software Product Management: An Incremental Method Engineering Approach.*
2009-35 Wouter Koelewijn (UL), *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling.*
2009-36 Marco Kalz (OUN), *Placement Support for Learners in Learning Networks.*
2009-37 Hendrik Drachsler (OUN), *Navigation Support for Learners in Informal Learning Networks.*
2009-38 Riina Vuorikari (OU), *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context.*
2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin), *Service Substitution – A Behavioral Approach Based on Petri Nets.*
2009-40 Stephan Raaijmakers (UvT), *Multinomial Language Learning: Investigations into the Geometry of Language.*
2009-41 Igor Berezhnyy (UvT), *Digital Analysis of Paintings.*
2009-42 Toine Bogers (UvT), *Recommender Systems for Social Bookmarking.*
2009-43 Virginia Nunes Leal Franqueira (UT), *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients.*
2009-44 Roberto Santana Tapia (UT), *Assessing Business-IT Alignment in Networked Organizations.*
2009-45 Jilles Vreeken (UU), *Making Pattern Mining Useful.*
2009-46 Loredana Afanasiev (UvA), *Querying XML: Benchmarks and Recursion.*

2010

2010-01 Matthijs van Leeuwen (UU), *Patterns that Matter.*
2010-02 Ingo Wassink (UT), *Work flows in Life Science.*
2010-03 Joost Geurts (CWI), *A Document Engineering Model and Processing Framework for Multimedia documents.*
2010-04 Olga Kulyk (UT), *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments.*
2010-05 Claudia Hauff (UT), *Predicting the Effectiveness of Queries and Retrieval Systems.*
2010-06 Sander Bakkes (UvT), *Rapid Adaptation of Video Game AI.*
2010-07 Wim Fikkert (UT), *Gesture interaction at a Distance.*
2010-08 Krzysztof Siewicz (UL), *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments.*
2010-09 Hugo Kielman (UL), *A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging.*
2010-10 Rebecca Ong (UL), *Mobile Communication and Protection of Children.*
2010-11 Adriaan Ter Mors (TUD), *The world according to MARP: Multi-Agent Route Planning.*
2010-12 Susan van den Braak (UU), *Sensemaking software for crime analysis.*

2010-13 Gianluigi Folino (RUN), *High Performance Data Mining using Bio-inspired techniques.*
2010-14 Sander van Splunter (VU), *Automated Web Service Reconfiguration.*
2010-15 Lianne Bodenstaff (UT), *Managing Dependency Relations in Inter-Organizational Models.*
2010-16 Sicco Verwer (TUD), *Efficient Identification of Timed Automata, theory and practice.*
2010-17 Spyros Kotoulas (VU), *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications.*
2010-18 Charlotte Gerritsen (VU), *Caught in the Act: Investigating Crime by Agent-Based Simulation.*
2010-19 Henriette Cramer (UvA), *People's Responses to Autonomous and Adaptive Systems.*
2010-20 Ivo Swartjes (UT), *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative.*
2010-21 Harold van Heerde (UT), *Privacy-aware data management by means of data degradation.*
2010-22 Michiel Hildebrand (CWI), *End-user Support for Access to Heterogeneous Linked Data.*
2010-23 Bas Steunebrink (UU), *The Logical Structure of Emotions.*
2010-24 Dmytro Tykhonov, *Designing Generic and Efficient Negotiation Strategies.*
2010-25 Zulfiqar Ali Memon (VU), *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective.*
2010-26 Ying Zhang (CWI), *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines.*
2010-27 Marten Voulon (UL), *Automatisch contracteren.*
2010-28 Arne Koopman (UU), *Characteristic Relational Patterns.*
2010-29 Stratos Idreos(CWI), *Database Cracking: Towards Auto-tuning Database Kernels.*
2010-30 Marieke van Erp (UvT), *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval.*
2010-31 Victor de Boer (UVA), *Ontology Enrichment from Heterogeneous Sources on the Web.*
2010-32 Marcel Hiel (UvT), *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems.*
2010-33 Robin Aly (UT), *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval.*
2010-34 Teduh Dirgahayu (UT), *Interaction Design in Service Compositions.*
2010-35 Dolf Trieschnigg (UT), *Proof of Concept: Concept-based Biomedical Information Retrieval.*
2010-36 Jose Janssen (OU), *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification.*
2010-37 Niels Lohmann (TUE), *Correctness of services and their composition.*
2010-38 Dirk Fahland (TUE), *From Scenarios to components.*
2010-39 Ghazanfar Farooq Siddiqui (VU), *Integrative modeling of emotions in virtual agents.*
2010-40 Mark van Assem (VU), *Converting and Integrating Vocabularies for the Semantic Web.*
2010-41 Guillaume Chaslot (UM), *Monte-Carlo Tree Search.*

2010-42 Sybren de Kinderen (VU), *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach.*
2010-43 Peter van Kranenburg (UU), *A Computational Approach to Content-Based Retrieval of Folk Song Melodies.*
2010-44 Pieter Bellekens (TUE), *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain.*
2010-45 Vasilios Andrikopoulos (UvT), *A theory and model for the evolution of software services.*
2010-46 Vincent Pijpers (VU), *e3alignment: Exploring Inter-Organizational Business-ICT Alignment.*
2010-47 Chen Li (UT), *Mining Process Model Variants: Challenges, Techniques, Examples.*
2010-48 Withdrawn, *.*
2010-49 Jahn-Takeshi Saito (UM), *Solving difficult game positions.*
2010-50 Bouke Huurnink (UVA), *Search in Audiovisual Broadcast Archives.*
2010-51 Alia Khairia Amin (CWI), *Understanding and supporting information seeking tasks in multiple sources.*
2010-52 Peter-Paul van Maanen (VU), *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention.*
2010-53 Edgar Meij (UVA), *Combining Concepts and Language Models for Information Access.*

2011

2011-01 Botond Cseke (RUN), *Variational Algorithms for Bayesian Inference in Latent Gaussian Models.*
2011-02 Nick Tinnemeier(UU), *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language.*
2011-03 Jan Martijn van der Werf (TUE), *Compositional Design and Verification of Component-Based Information Systems.*
2011-04 Hado van Hasselt (UU), *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference.*
2011-05 Base van der Raadt (VU), *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline..*
2011-06 Yiwen Wang (TUE), *Semantically-Enhanced Recommendations in Cultural Heritage.*
2011-07 Yujia Cao (UT), *Multimodal Information Presentation for High Load Human Computer Interaction.*
2011-08 Nieske Vergunst (UU), *BDI-based Generation of Robust Task-Oriented Dialogues.*
2011-09 Tim de Jong (OU), *Contextualised Mobile Media for Learning.*
2011-10 Bart Bogaert (UvT), *Cloud Content Contention.*
2011-11 Dhaval Vyas (UT), *Designing for Awareness: An Experience-focused HCI Perspective.*
2011-12 Carmen Bratosin (TUE), *Grid Architecture for Distributed Process Mining.*
2011-13 Xiaoyu Mao (UvT), *Airport under Control. Multiagent Scheduling for Airport Ground Handling.*
2011-14 Milan Lovric (EUR), *Behavioral Finance and Agent-Based Artificial Markets.*
2011-15 Marijn Koolen (UvA), *The Meaning of Structure: the Value of Link Evidence*

for Information Retrieval.
2011-16 Maarten Schadd (UM), *Selective Search in Games of Different Complexity.*
2011-17 Jiyin He (UVA), *Exploring Topic Structure: Coherence, Diversity and Relatedness.*
2011-18 Mark Ponsen (UM), *Strategic Decision-Making in complex games.*
2011-19 Ellen Rusman (OU), *The Mind ' s Eye on Personal Profiles.*
2011-20 Qing Gu (VU), *Guiding service-oriented software engineering - A view-based approach.*
2011-21 Linda Terlouw (TUD), *Modularization and Specification of Service-Oriented Systems.*
2011-22 Junte Zhang (UVA), *System Evaluation of Archival Description and Access.*
2011-23 Wouter Weerkamp (UVA), *Finding People and their Utterances in Social Media.*
2011-24 Herwin van Welbergen (UT), *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior.*
2011-25 Syed Waqar ul Qounain Jaffry (VU)), *Analysis and Validation of Models for Trust Dynamics.*
2011-26 Matthijs Aart Pontier (VU), *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots.*
2011-27 Aniel Bhulai (VU), *Dynamic website optimization through autonomous management of design patterns.*
2011-28 Rianne Kaptein(UVA), *Effective Focused Retrieval by Exploiting Query Context and Document Structure.*
2011-29 Faisal Kamiran (TUE), *Discrimination-aware Classification.*
2011-30 Egon van den Broek (UT), *Affective Signal Processing (ASP): Unraveling the mystery of emotions.*
2011-31 Ludo Waltman (EUR), *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality.*
2011-32 Nees-Jan van Eck (EUR), *Methodological Advances in Bibliometric Mapping of Science.*
2011-33 Tom van der Weide (UU), *Arguing to Motivate Decisions.*
2011-34 Paolo Turrini (UU), *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations.*
2011-35 Maaike Harbers (UU), *Explaining Agent Behavior in Virtual Training.*
2011-36 Erik van der Spek (UU), *Experiments in serious game design: a cognitive approach.*
2011-37 Adriana Burlutiu (RUN), *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference.*
2011-38 Nyree Lemmens (UM), *Bee-inspired Distributed Optimization.*
2011-39 Joost Westra (UU), *Organizing Adaptation using Agents in Serious Games.*
2011-40 Viktor Clerc (VU), *Architectural Knowledge Management in Global Software Development.*
2011-41 Luan Ibraimi (UT), *Cryptographically Enforced Distributed Data Access Control.*
2011-42 Michal Sindlar (UU), *Explaining Behavior through Mental State Attribution.*
2011-43 Henk van der Schuur (UU), *Process Improvement through Software Operation Knowledge.*

2011-44 Boris Reuderink (UT), *Robust Brain-Computer Interfaces.*
2011-45 Herman Stehouwer (UvT), *Statistical Language Models for Alternative Sequence Selection.*
2011-46 Beibei Hu (TUD), *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work.*
2011-47 Azizi Bin Ab Aziz(VU), *Exploring Computational Models for Intelligent Support of Persons with Depression.*
2011-48 Mark Ter Maat (UT), *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent.*
2011-49 Andreea Niculescu (UT), *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality.*

2012

2012-01 Terry Kakeeto (UvT), *Relationship Marketing for SMEs in Uganda.*
2012-02 Muhammad Umair(VU), *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models.*
2012-03 Adam Vanya (VU), *Supporting Architecture Evolution by Mining Software Repositories.*
2012-04 Jurriaan Souer (UU), *Development of Content Management System-based Web Applications.*
2012-05 Marijn Plomp (UU), *Maturing Interorganisational Information Systems.*
2012-06 Wolfgang Reinhardt (OU), *Awareness Support for Knowledge Workers in Research Networks.*
2012-07 Rianne van Lambalgen (VU), *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions.*
2012-08 Gerben de Vries (UVA), *Kernel Methods for Vessel Trajectories.*
2012-09 Ricardo Neisse (UT), *Trust and Privacy Management Support for Context-Aware Service Platforms.*
2012-10 David Smits (TUE), *Towards a Generic Distributed Adaptive Hypermedia Environment.*
2012-11 J.C.B. Rantham Prabhakara (TUE), *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics.*
2012-12 Kees van der Sluijs (TUE), *Model Driven Design and Data Integration in Semantic Web Information Systems.*
2012-13 Suleman Shahid (UvT), *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions.*
2012-14 Evgeny Knutov(TUE), *Generic Adaptation Framework for Unifying Adaptive Web-based Systems.*
2012-15 Natalie van der Wal (VU), *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes..*
2012-16 Fiemke Both (VU), *Helping people by understanding them - Ambient Agents supporting task execution and depression treatment.*
2012-17 Amal Elgammal (UvT), *Towards a Comprehensive Framework for Business Process Compliance.*
2012-18 Eltjo Poort (VU), *Improving Solution Architecting Practices.*
2012-19 Helen Schonenberg (TUE), *What's Next? Operational Support for Business Process Execution.*
2012-20 Ali Bahramisharif (RUN), *Covert Visual Spatial Attention, a Robust Paradigm*

for Brain-Computer Interfacing.
2012-21 Roberto Cornacchia (TUD), *Querying Sparse Matrices for Information Retrieval.*
2012-22 Thijs Vis (UvT), *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?.*
2012-23 Christian Muehl (UT), *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction.*
2012-24 Laurens van der Werff (UT), *Evaluation of Noisy Transcripts for Spoken Document Retrieval.*
2012-25 Silja Eckartz (UT), *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application.*
2012-26 Emile de Maat (UVA), *Making Sense of Legal Text.*
2012-27 Hayrettin Gurkok (UT), *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games.*
2012-28 Nancy Pascall (UvT), *Engendering Technology Empowering Women.*
2012-29 Almer Tigelaar (UT), *Peer-to-Peer Information Retrieval.*
2012-30 Alina Pommeranz (TUD), *Designing Human-Centered Systems for Reflective Decision Making.*
2012-31 Emily Bagarukayo (RUN), *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure.*
2012-32 Wietske Visser (TUD), *Qualitative multi-criteria preference representation and reasoning.*
2012-33 Rory Sie (OUN), *Coalitions in Cooperation Networks (COCOON).*
2012-34 Pavol Jancura (RUN), *Evolutionary analysis in PPI networks and applications.*
2012-35 Evert Haasdijk (VU), *Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics.*
2012-36 Denis Ssebugwawo (RUN), *Analysis and Evaluation of Collaborative Modeling Processes.*
2012-37 Agnes Nakakawa (RUN), *A Collaboration Process for Enterprise Architecture Creation.*
2012-38 Selmar Smit (VU), *Parameter Tuning and Scientific Testing in Evolutionary Algorithms.*
2012-39 Hassan Fatemi (UT), *Risk-aware design of value and coordination networks.*
2012-40 Agus Gunawan (UvT), *Information Access for SMEs in Indonesia.*
2012-41 Sebastian Kelle (OU), *Game Design Patterns for Learning.*
2012-42 Dominique Verpoorten (OU), *Reflection Amplifiers in self-regulated Learning.*
2012-43 Withdrawn, .
2012-44 Anna Tordai (VU), *On Combining Alignment Techniques.*
2012-45 Benedikt Kratz (UvT), *A Model and Language for Business-aware Transactions.*
2012-46 Simon Carter (UVA), *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation.*
2012-47 Manos Tsagkias (UVA), *Mining Social Media: Tracking Content and Predicting Behavior.*
2012-48 Jorn Bakker (TUE), *Handling Abrupt Changes in Evolving Time-series Data.*
2012-49 Michael Kaisers (UM), *Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions.*
2012-50 Steven van Kervel (TUD), *Ontologogy driven Enterprise Information Systems*

Engineering.
2012-51 Jeroen de Jong (TUD), *Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching.*

2013

2013-01 Viorel Milea (EUR), *News Analytics for Financial Decision Support.*
2013-02 Erietta Liarou (CWI), *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing.*
2013-03 Szymon Klarman (VU), *Reasoning with Contexts in Description Logics.*
2013-04 Chetan Yadati(TUD), *Coordinating autonomous planning and scheduling.*
2013-05 Dulce Pumareja (UT), *Groupware Requirements Evolutions Patterns.*
2013-06 Romulo Goncalves(CWI), *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience.*
2013-07 Giel van Lankveld (UvT), *Quantifying Individual Player Differences.*
2013-08 Robbert-Jan Merk(VU), *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators.*
2013-09 Fabio Gori (RUN), *Metagenomic Data Analysis: Computational Methods and Applications.*
2013-10 Jeewanie Jayasinghe Arachchige(UvT), *A Unified Modeling Framework for Service Design..*
2013-11 Evangelos Pournaras(TUD), *Multi-level Reconfigurable Self-organization in Overlay Services.*
2013-12 Marian Razavian(VU), *Knowledge-driven Migration to Services.*
2013-13 Mohammad Safiri(UT), *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly.*
2013-14 Jafar Tanha (UVA), *Ensemble Approaches to Semi-Supervised Learning Learning.*
2013-15 Daniel Hennes (UM), *Multiagent Learning - Dynamic Games and Applications.*
2013-16 Eric Kok (UU), *Exploring the practical benefits of argumentation in multi-agent deliberation.*
2013-17 Koen Kok (VU), *The PowerMatcher: Smart Coordination for the Smart Electricity Grid.*
2013-18 Jeroen Janssens (UvT), *Outlier Selection and One-Class Classification.*
2013-19 Renze Steenhuizen (TUD), *Coordinated Multi-Agent Planning and Scheduling.*
2013-20 Katja Hofmann (UvA), *Fast and Reliable Online Learning to Rank for Information Retrieval.*
2013-21 Sander Wubben (UvT), *Text-to-text generation by monolingual machine translation.*
2013-22 Tom Claassen (RUN), *Causal Discovery and Logic.*
2013-23 Patricio de Alencar Silva(UvT), *Value Activity Monitoring.*
2013-24 Haitham Bou Ammar (UM), *Automated Transfer in Reinforcement Learning.*
2013-25 Agnieszka Anna Latoszek-Berendsen (UM), *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System.*
2013-26 Alireza Zarghami (UT), *Architectural Support for Dynamic Homecare Service Provisioning.*

2013-27 Mohammad Huq (UT), *Inference-based Framework Managing Data Provenance.*
2013-28 Frans van der Sluis (UT), *When Complexity becomes Interesting: An Inquiry into the Information eXperience.*
2013-29 Iwan de Kok (UT), *Listening Heads.*
2013-30 Joyce Nakatumba (TUE), *Resource-Aware Business Process Management: Analysis and Support.*
2013-31 Dinh Khoa Nguyen (UvT), *Blueprint Model and Language for Engineering Cloud Applications.*
2013-32 Kamakshi Rajagopal (OUN), *Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development.*
2013-33 Qi Gao (TUD), *User Modeling and Personalization in the Microblogging Sphere.*
2013-34 Kien Tjin-Kam-Jet (UT), *Distributed Deep Web Search.*
2013-35 Abdallah El Ali (UvA), *Minimal Mobile Human Computer Interaction.*
2013-36 Than Lam Hoang (TUe), *Pattern Mining in Data Streams.*
2013-37 Dirk Börner (OUN), *Ambient Learning Displays.*
2013-38 Eelco den Heijer (VU), *Autonomous Evolutionary Art.*
2013-39 Joop de Jong (TUD), *A Method for Enterprise Ontology based Design of Enterprise Information Systems.*
2013-40 Pim Nijssen (UM), *Monte-Carlo Tree Search for Multi-Player Games.*
2013-41 Jochem Liem (UVA), *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning.*
2013-42 Léon Planken (TUD), *Algorithms for Simple Temporal Reasoning.*
2013-43 Marc Bron (UVA), *Exploration and Contextualization through Interaction and Concepts.*

2014

2014-01 Nicola Barile (UU), *Studies in Learning Monotone Models from Data.*
2014-02 Fiona Tuliyano (RUN), *Combining System Dynamics with a Domain Modeling Method.*
2014-03 Sergio Raul Duarte Torres (UT), *Information Retrieval for Children: Search Behavior and Solutions.*
2014-04 Hanna Jochmann-Mannak (UT), *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation.*
2014-05 Jurriaan van Reijsen (UU), *Knowledge Perspectives on Advancing Dynamic Capability.*
2014-06 Damian Tamburri (VU), *Supporting Networked Software Development.*
2014-07 Arya Adriansyah (TUE), *Aligning Observed and Modeled Behavior.*
2014-08 Samur Araujo (TUD), *Data Integration over Distributed and Heterogeneous Data Endpoints.*
2014-09 Philip Jackson (UvT), *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language.*
2014-10 Ivan Salvador Razo Zapata (VU), *Service Value Networks.*
2014-11 Janneke van der Zwaan (TUD), *An Empathic Virtual Buddy for Social Support.*
2014-12 Willem van Willigen (VU), *Look Ma, No Hands: Aspects of Autonomous Vehicle Control.*
2014-13 Arlette van Wissen (VU), *Agent-Based Support for Behavior Change: Models*

and Applications in Health and Safety Domains.
2014-14 Yangyang Shi (TUD), *Language Models With Meta-information.*
2014-15 Natalya Mogles (VU), *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare.*
2014-16 Krystyna Milian (VU), *Supporting trial recruitment and design by automatically interpreting eligibility criteria.*
2014-17 Kathrin Dentler (VU), *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability.*
2014-18 Mattijs Ghijsen (VU), *Methods and Models for the Design and Study of Dynamic Agent Organizations.*
2014-19 Vincius Ramos (TUE), *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support.*
2014-20 Mena Habib (UT), *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link.*
2014-21 Kassidy Clark (TUD), *Negotiation and Monitoring in Open Environments.*
2014-22 Marieke Peeters (UU), *Personalized Educational Games - Developing agent-supported scenario-based training.*
2014-23 Eleftherios Sidirourgos (UvA/CWI), *Space Efficient Indexes for the Big Data Era.*
2014-24 Davide Ceolin (VU), *Trusting Semi-structured Web Data.*
2014-25 Martijn Lappenschaar (RUN), *New network models for the analysis of disease interaction.*
2014-26 Tim Baarslag (TUD), *What to Bid and When to Stop.*
2014-27 Rui Jorge Almeida (EUR), *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty.*
2014-28 Anna Chmielowiec (VU), *Decentralized k-Clique Matching.*
2014-29 Jaap Kabbedijk (UU), *Variability in Multi-Tenant Enterprise Software.*
2014-30 Peter de Cock (UvT), *Anticipating Criminal Behaviour.*
2014-31 Leo van Moergestel (UU), *Agent Technology in Agile Multiparallel Manufacturing and Product Support.*
2014-32 Naser Ayat (UvA), *On Entity Resolution in Probabilistic Data.*
2014-33 Tesfa Tegegne (RUN), *Service Discovery in eHealth.*
2014-34 Christina Manteli(VU), *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems..*
2014-35 Joost van Ooijen (UU), *Cognitive Agents in Virtual Worlds: A Middleware Design Approach.*
2014-36 Joos Buijs (TUE), *Flexible Evolutionary Algorithms for Mining Structured Process Models.*
2014-37 Maral Dadvar (UT), *Experts and Machines United Against Cyberbullying.*
2014-38 Danny Plass-Oude Bos (UT), *Making brain-computer interfaces better: improving usability through post-processing..*
2014-39 Jasmina Maric (UvT), *Web Communities, Immigration, and Social Capital.*
2014-40 Walter Omona (RUN), *A Framework for Knowledge Management Using ICT in Higher Education.*
2014-41 Frederic Hogenboom (EUR), *Automated Detection of Financial Events in News Text.*
2014-42 Carsten Eijckhof (CWI/TUD), *Contextual Multidimensional Relevance Models.*
2014-43 Kevin Vlaanderen (UU), *Supporting Process Improvement using Method In-*

crements.
2014-44 Paulien Meesters (UvT), *Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden..*
2014-45 Birgit Schmitz (OUN), *Mobile Games for Learning: A Pattern-Based Approach.*
2014-46 Ke Tao (TUD), *Social Web Data Analytics: Relevance, Redundancy, Diversity.*
2014-47 Shangsong Liang (UVA), *Fusion and Diversification in Information Retrieval.*

2015

2015-01 Niels Netten (UvA), *Machine Learning for Relevance of Information in Crisis Response.*
2015-02 Faiza Bukhsh (UvT), *Smart auditing: Innovative Compliance Checking in Customs Controls.*
2015-03 Twan van Laarhoven (RUN), *Machine learning for network data.*
2015-04 Howard Spoelstra (OUN), *Collaborations in Open Learning Environments.*
2015-05 Christoph Bösch(UT), *Cryptographically Enforced Search Pattern Hiding.*
2015-06 Farideh Heidari (TUD), *Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes.*
2015-07 Maria-Hendrike Peetz(UvA), *Time-Aware Online Reputation Analysis.*
2015-08 Jie Jiang (TUD), *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions.*
2015-09 Randy Klaassen(UT), *HCI Perspectives on Behavior Change Support Systems.*
2015-10 Henry Hermans (OUN), *OpenU: design of an integrated system to support lifelong learning.*
2015-11 Yongming Luo (TUE), *Designing algorithms for big graph datasets: A study of computing bisimulation and joins.*
2015-12 Julie M. Birkholz (VU), *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks.*
2015-13 Giuseppe Procaccianti(VU), *Energy-Efficient Software.*
2015-14 Bart van Straalen (UT), *A cognitive approach to modeling bad news conversations.*
2015-15 Klaas Andries de Graaf (VU), *Ontology-based Software Architecture Documentation.*
2015-16 Changyun Wei (UT), *Cognitive Coordination for Cooperative Multi-Robot Teamwork.*
2015-17 Andre van Cleeff (UT), *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs.*
2015-18 Holger Pirk (CWI), *Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories.*
2015-19 Bernardo Tabuenca (OUN), *Ubiquitous Technology for Lifelong Learners.*
2015-20 Lois Vanhée(UU), *Using Culture and Values to Support Flexible Coordination.*
2015-21 Sibren Fetter (OUN), *Using Peer-Support to Expand and Stabilize Online Learning.*
2015-22 Zhemin Zhu(UT), *Co-occurrence Rate Networks.*
2015-23 Luit Gazendam (VU), *Cataloguer Support in Cultural Heritage.*
2015-24 Richard Berendsen (UVA), *Finding People, Papers, and Posts: Vertical Search*

Algorithms and Evaluation.
2015-25 Steven Woudenberg (UU), *Bayesian Tools for Early Disease Detection.*
2015-26 Alexander Hogenboom (EUR), *Sentiment Analysis of Text Guided by Semantics and Structure.*
2015-27 Sandor Héman (CWI), *Updating compressed colomn stores.*
2015-28 Janet Bagorogoza(TiU), *KNOWLEDGE MANAGEMENT AND HIGH PERFORMANCE; The Uganda Financial Institutions Model for HPO.*
2015-29 Hendrik Baier (UM), *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains.*
2015-30 Kiavash Bahreini(OU), *Real-time Multimodal Emotion Recognition in E-Learning.*
2015-31 Yakup Ko c (TUD), *On the robustness of Power Grids.*
2015-32 Jerome Gard(UL), *Corporate Venture Management in SMEs.*
2015-33 Frederik Schadd (TUD), *Ontology Mapping with Auxiliary Resources.*
2015-34 Victor de Graaf(UT), *Gesocial Recommender Systems.*
2015-35 Jungxao Xu (TUD), *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction.*

2016

2016-01 Syed Saiden Abbas (RUN), *Recognition of Shapes by Humans and Machines.*
2016-02 Michiel Christiaan Meulendijk (UU), *Optimizing medication reviews through decision support: prescribing a better pill to swallow.*
2016-03 Maya Sappelli (RUN), *Knowledge Work in Context: User Centered Knowledge Worker Support.*
2016-04 Laurens Rietveld (VU), *Publishing and Consuming Linked Data.*
2016-05 Evgeny Sherkhonov (UVA), *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers.*
2016-06 Michel Wilson (TUD), *Robust scheduling in an uncertain environment.*
2016-07 Jeroen de Man (VU), *Measuring and modeling negative emotions for virtual training.*
2016-08 Matje van de Camp (TiU), *A Link to the Past: Constructing Historical Social Networks from Unstructured Data.*
2016-09 Archana Nottamkandath (VU), *Trusting Crowdsourced Information on Cultural Artefacts.*
2016-10 George Karafotias (VUA), *Parameter Control for Evolutionary Algorithms.*
2016-11 Anne Schuth (UVA), *Search Engines that Learn from Their Users.*
2016-12 Max Knobbout (UU), *Logics for Modelling and Verifying Normative Multi-Agent Systems.*
2016-13 Nana Baah Gyan (VU), *The Web, Speech Technologies and Rural Development in West Africa - An.*
ICT4D Approach **2016-14** Ravi Khadka (UU), *Revisiting Legacy Software System Modernization.*
2016-15 Steffen Michels (RUN), *Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments.*
2016-16 Guangliang Li (UVA), *Socially Intelligent Autonomous Agents that Learn from Human Reward.*
2016-17 Berend Weel (VU), *Towards Embodied Evolution of Robot Organisms.*
2016-18 Albert Meroño Peñuela, *Refining Statistical Data on the Web.*
2016-19 Julia Efremova (Tu/e), *Mining Social Structures from Genealogical Data.*

2016-20 Daan Odijk (UVA), *Context & Semantics in News & Web Search.*
2016-21 Alejandro Moreno Célleri (UT), *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground.*
2016-22 Grace Lewis (VU), *Software Architecture Strategies for Cyber-Foraging Systems.*
2016-23 Fei Cai (UVA), *Query Auto Completion in Information Retrieval.*
2016-24 Brend Wanders (UT), *Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach.*
2016-25 Julia Kiseleva (TU/e), *Using Contextual Information to Understand Searching and Browsing Behavior.*
2016-26 Dilhan Thilakarathne (VU), *In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains.*
2016-27 Wen Li (TUD), *Understanding Geo-spatial Information on Social Media.*
2016-28 Mingxin Zhang (TUD), *Large-scale Agent-based Social Simulation - A study on epidemic prediction and control.*
2016-29 Nicolas Höning (TUD), *Peak reduction in decentralised electricity systems - Markets and prices for flexible planning.*
2016-30 Ruud Mattheij (UvT), *The Eyes Have It.*
2016-31 Mohammad Khelghati (UT), *Deep web content monitoring.*
2016-32 Eelco Vriezekolk (UT), *Assessing Telecommunication Service Availability Risks for Crisis Organisations.*
2016-33 Peter Bloem (UVA), *Single Sample Statistics, exercises in learning from just one example.*
2016-34 Dennis Schunselaar (TUE), *Title: Configurable Process Trees: Elicitation, Analysis, and Enactment.*
2016-35 Zhaochun Ren, *Monitoring Social Media: Summarization, Classification and Recommendation.*
2016-36 Daphne Karreman (UT), *Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies.*
2016-37 Giovanni Sileno (UvA), *Aligning Law and Action - a conceptual and computational inquiry .*
2016-38 Andrea Minuto (UT), *MATERIALS THAT MATTER - Smart Materials meet Art & Interaction Design.*
2016-39 Merijn Bruijnes (UT), *Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect.*
2016-40 Christian Detweiler (TUD), *Accounting for Values in Design.*
2016-41 Thomas King (TUD), *Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance.*
2016-42 Spyros Martzoukos (UVA) , *Combinatorial and Compositional Aspects of Bilingual Aligned Corpora .*
2016-43 Saskia Koldijk (RUN), *Context-Aware Support for Stress Self-Management: From Theory to Practice.*