Search and Retrieval in Massive Data Collections

Pedro Omar Contreras Albornoz
Licenciado, M.Phil.

Submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science
Royal Holloway, University of London

May 2010

Declaration

I declare that this dissertation was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Pedro O. Contreras Albornoz
May 2010

Supervisor and Examiners

Supervisor: *Prof. Fionn Murtagh*
Internal Examiner: *Prof. Boris Mirkin*
External Examiner: *Prof. Xiaohui Liu*

Abstract

THE main goal of this research is to produce a novel and efficient searching application by means of best match and proximity searching with particular application to very large numeric and textual data stores.

In today's world a huge amount of information is produced. Almost every part of our society is touched by systems that collect, store and analyse data. As an example I mention the case of scientific instrumentation: new sensors capture massive amounts of information (e.g. new telescopes acquiring data from different regions of the spectrum).

Description of biological and chemical interactions also produce complex and large amounts of data. It is in this context that a big challenge for current analysis algorithms is presented. Many of the traditional methods for data analysis do not scale well in massive data sets nor in very high dimensional spaces.

In this work I introduce a novel (ultrametric) distance called Baire based on the longest common prefix and show how it can be used to produce clusters through grouping data in 'bins' taking linear or O(n) computational time. Furthermore, it follows that this distance can be strictly fitted to a hierarchy tree. This is a property that proves very useful for classifying, storing, accessing and retrieving information.

I go further to apply this methodology on data from different scientific areas such as astronomy and chemistry to create groups or clusters. Additionally I apply this method to document sets for clustering and retrieval. In particular, I look into the new area of enterprise search to propose a new method to support scalable search and clustering.

This thesis is dedicated to my family
and my home city Talca.

Acknowledgements

First and furthermost I would like thank my supervisor Prof. Fionn Murtagh whose help has been invaluable to provide me with support, good advice and funding. He has always been readily available to guide me and to make me aware of many things that I did not know. I have always been very impressed by his untiring efforts in his work, and the way he applies his knowledge to the most diverse areas in research.

I would like to acknowledge the following people:

- Mapping z_{spec} and z_{phot} in chapter 3 was carried out with data provided by Prof. Giuseppe Longo's research group, with collaboration from Dr Raffaele D'Abrusco. Dipartimento di Fisica (Sezione di Astrofisica) – Università degli Studi di Napoli Federico II, Naples, Italy.
- The chemical dataset, used in chapter 4, was provided by Dr Geoff Downs, Digital Chemistry Ltd. Java code for translating encoded chemical compounds to plain text was implemented by Dimitrios Zervas, Royal Holloway, University of London.
- Work carried out in chapter 5 was done in collaboration with Dr Marcel Ramirez. Laboratório Biologia Molecular de Parasitas e Vetores. Instituto Oswaldo Cruz, FIOCRUZ, Rio de Janeiro, Brazil, who hosted and funded part of my three week stay at FIOCRUZ.
- Semantic preservation work in chapter 7 was carried out in conjunction with Dr Javier Pereira from University Diego Portales, Santiago, Chile.

I would like to thank Julian Dean from ThinkingSAFE Ltd. with whom I had many meetings and very helpful conversations regarding new technologies and what are the problems faced in the data backup industry. From these discussions have followed the central theme of this work in developing a new matching and search algorithm for use in the enterprise setting.

I would like to thank Vera Hazelwood, who was the Smith Institute liaison agent between Royal Holloway and ThinkingSAFE Ltd.

Egham, United Kingdom *Pedro O. Contreras Albornoz*
May 2010

Preface

The main goal of this project is to identify new mathematical approaches to best match and proximity searching with particular application to very large data stores.

In this dissertation I develop a new, linear time hierarchical clustering algorithm and I validate it in a wide range of cases.

This thesis was made possible thanks to the support of the Engineering and Physical Sciences Research Council (EPSRC) through an Industrial CASE award in conjunction with ThinkingSAFE Ltd. The award was available through the Knowledge Transfer Network for Industrial Mathematics with the participation of The Smith Institute for Industrial Mathematics and System Engineering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 About this thesis

THE main goal of this research is to produce a novel and efficient algorithm that can be used for hierarchical classification, matching, search and clustering. In particular we study the application of the proposed distance to very large data stores. Following this we look into application to diverse scientific areas, as well as information retrieval.

We start by giving the background and introduction to the problem which we aim to solve in **Chapter** 2. Furthermore, we look into the need for new algorithms to deal with the large amount of data available, and how the increasing dimensionality in data is a problem that needs addressing. Particular attention is given to clustering techniques, stressing hierarchical methods. Finally the Baire (ultra)metric distance is introduced and explained.

The use of the Baire (ultra)metric technique is explored in different scientific areas such as astronomy and chemistry for clustering.

In the information retrieval context matching of text is addressed. In particular the area of large data stores with semantically organised text is examined.

This thesis is structured in two main parts as follows:

In **Part I** the Baire distance is used in the scientific context. Specifically, we look into applications in astronomy, chemistry and biology. In **Chapter** 3 astronomical data from the Sloan Digital Sky Survey is used to study spectrometric and photometric redshifts. A cluster-wise mapping of one signal into the other is developed based on the Baire distance. Additionally, signals are clustered individually to produce digit distribution maps for comparison. In **Chapter** 4 the Baire distance is applied to clustering of chemical compounds. In this case a random projection is applied in order to reduce data dimensionality prior to clustering.

In **Chapter** 5 we look into genomic data using a modified Baire distance for clustering, namely the longest common substring. In this chapter our aim is to analyse Giardia hypothetical proteins in order to obtain clusters that can be used as targets for drug discovery.

In **Part II** we look into the information retrieval area. Particular attention is given to text matching, clustering and retrieval, all of this based on the Baire distance. In **Chapter** 6 we begin explaining how search engines work, their structure and how this is related to enterprise search, not only highlighting similarities with web search, but also stressing differences. **Chapter** 7 deals with the Baire distance applied in the context of information retrieval. In particular we look into the text retrieval setting.

In **Chapter** 8 overall conclusions are drawn. In addition we provide some discussion regarding possible extensions of our work.

1.2 Main contributions

Massive datasets are everywhere in today's world, and they are growing continually. This entails a number of issues and challenges to current technologies employed to capture, transmit, store and analyse this data. In this work we are particularly concerned with exploratory data analysis, namely clustering.

In this dissertation a novel (ultra)metric distance is proposed for clustering. We validate the clustering algorithm, as well as the context within which they are implemented, by comparing them with long established cluster algorithms such as *k*-means. Moreover, we use real life examples and applications to different scientific areas, as well as information retrieval to exemplify its use.

The proposed method presents a number of advantages when compared with more traditional techniques. When working with numeric data, distances can be interpreted directly and classification carried out. Furthermore, the resulting distances can be strictly fitted to a hierarchical tree. This is very important for classification, storing and fast search.

1.3 Publications by the author related to this thesis

Refereed journal, book compilation and conference proceedings articles

F. Murtagh and P. Contreras, "Hierarchical Clustering for Finding Symmetries and Other Patterns in Massive, High Dimensional Datasets", invited chapter in D. Holmes, editor, Data Mining: Foundations and Intelligent Paradigms, Springer, 2011, submitted.

P. Contreras and F. Murtagh, "A Very Fast, Linear Time p-Adic and Hierarchical Clustering Algorithm Using the Baire Metric", 2010, in preparation.

F. Murtagh and P. Contreras. "Methods of Hierarchical Clustering", in W. Pedrycz, Ed., Data Mining and Knowledge Discovery, Wiley Interdisciplinary Reviews (WIRES), submitted (invited), 2010 (for publication in 2011).

J. Pereira, F. Schmidt, P. Contreras, F. Murtagh, and H. Astudillo. "Clustering and Semantics Preservation in Cultural Heritage Information Spaces", RIAO'2010, 9th International Conference on Adaptivity, Personalization and Fusion of Heterogeneous Information. 28–30 April, 2010. Paris, France.

P. Contreras and F. Murtagh. "Fast Hierarchical Clustering From the Baire Distance", Classification as a Tool for Research. Proceedings of the 11th IFCS Biennial Conference and 33rd Annual Conference of the Gesellschaft für Klassifikation e.V., pp 235–243 March 13–18, 2009. Dresden, Germany.

S. Bormuri, V. Urovi, P. Contreras, and K. Stathis. "A Virtual E-retailing Environment in GOLEM", 4th International Conference on Intelligent Environments (IE 08), 21–22 July 2008. Seattle, USA.

F. Murtagh, G. Downs, and P. Contreras. "Hierarchical Clustering of Massive, High Dimensional Data Sets by Exploiting Ultrametric Embedding", SIAM Journal on Scientific Computing. Vol. 30, No. 2, pp. 707–730. February 2008.

D. Rosenberg, M. Lievonen, P. Contreras, F. Murtagh, G. Kuehn, and R. Doerner. "Application Design of Learning Grid in Computer-Mediated Communication". The Learning Grid Handbook, Concepts, Technologies and Applications, pp. 107–123. IOS Press. March 2008.

Presentations

P. Contreras and F. Murtagh. British Classification Society Meeting and AGM. "Fast Hierarchical Clustering Algorithm". Royal Holloway, University of London. 20 November 2009. Egham, UK.

P. Contreras and F. Murtagh. "Fast Hierarchical Clustering From the Baire Distance". Doctoral Consortium on Computer Science and Informatics. British Computer Society. 28 May 2009. London, UK.

P. Contreras, F. Murtagh, and J. Dean. "Fast Clustering Algorithm for Application in Science". Alan Tayler's Day, organised by Smith Institute for Industrial Mathematics and System Engineering. St. Catherine's College. 24 October 2008. Oxford, UK.

P. Contreras and F. Murtagh. "Fast Clusterwise m-Adic Regression: Application to Redshift Calibration". 5th Astronomical Data Analysis Conference. 7–9 May 2008. Heraklion, Greece.

P. Contreras and F. Murtagh. "Evaluation of Hierarchies Based on the Longest Common Prefix or Baire Metric". The North American Classification Society Annual Meeting (CSNA). University of Illinois. 9 June 2007. Urbana-Champaign, USA.

Posters

P. Contreras and F. Murtagh. “Fast Hierarchical Clustering Algorithm for Redshift Calibration”. 6th Astronomical Data Analysis Conference, 3–7 May 2010. Monastir, Tunisia.

P. Contreras, F. Murtagh, and J. Dean. “Fast Clustering Algorithm for Application in Science”. Tayler’s Day, organised by Smith Institute for Industrial Mathematics and System Engineering, 30 November 2009. Oxford, UK.

I. Evans-Osses, A. Santos e Silva, J. Toscano, P. Contreras, and M. Ramirez. “A Combined in Silico and in Vitro Strategy to Analyze Hypothetical Proteins in Giardia Intestinalis”. XXIV Annual Meeting of the Brazilian Society for Protozoology/XXXV Meeting on Basic Research in Chagas Disease. 27–29 October 2008. Águas de Lindóia. Brasil.

P. Contreras, F. Murtagh, and J. Dean. “Data Matching and Classification”. Alan Tayler’s Day, organised by Smith Institute for Industrial Mathematics and System Engineering. 26 November 2007. Oxford, UK.

Seminars and lectures

P. Contreras. “m-Adic Regression: Application to Redshift Calibration”. Postgraduate Colloquium. Royal Holloway, University of London. 4 June 2008. Egham, UK. http://www.cs.rhul.ac.uk/Internal/For-Students/Postgrads/Colloquium/2008/Talks/pedro.pdf

The following presentations can be found in: http://www.cs.rhul.ac.uk/~pedro/lectures.html

Guest lectures, M.Sc. Business Information Systems. Royal Holloway, University of London: “Data Mining”. 20 February 2008; “Information Retrieval”. 27 February 2008. Egham, UK.

Computer Science Departmental seminar. Royal Holloway, University of London. “Understanding How Search Engines Work, an Information Retrieval Perspective”. 26 February 2008. Egham, UK.

CS253 Group Projects guest lectures. Department of Computer Science. Royal Holloway, University of London. “Organisational Aspects of Software Development”; “Working with Databases and Java”. January 2008, January 2009, . Egham, UK.

Chapter 2
Background and Related Work

2.1 Introduction

IN this work we are concerned with clustering, in particular with fast clustering for massive and high dimensional datasets. This presents a number of problems for traditional clustering methods. Currently there are many clustering algorithms that do not scale well with high volumes of data, and they are further troubled when this data has many features (i.e. dimensions) to consider. This is a very active area of research. Many methodologies have been proposed to address this problem. Normally we will find that the more precise a clustering algorithm is in finding groups, the more calculations are involved. This increases the computational complexity, which is one of the algorithmic issues that we would like to avoid. In general it can be said that we are willing to sacrifice precision for speed. Therefore, the Baire distance proposed in this work can be seen as a fast way to obtain clusters, but not always the optimal way.

In this chapter we introduce the background for this dissertation. We begin by describing how the massive increase in data emphasises the need for new methods that can cope well with the explosion in volume and dimensionality of the available data. The *curse of dimensionality* is explained together with some techniques that help in reducing dimensionality by means of mapping the data space to a lower dimension. Particular attention is given by us to the random projection method.

We follow by presenting some of the most important clustering algorithms, focusing on hierarchical, grid-based and density-based clustering methods. Then the ultrametric Baire distance is explained, which is used to build a hierarchy and in turn for clustering and matching. This is the basis for most of the work carried out throughout this dissertation.

This process is possible because arising directly out of the Baire distance is an ultrametric tree, which also can be seen as a tree that hierarchically clusters data. This presents a number of advantages when storing and retrieving data. When the data source is in numerical form this ultrametric tree can be used as an index structure making matching and search, thus retrieval, much easier. In the case of textual information the longest common prefix can be used to find the best match, making easier the retrieval process since here again the information can be stored in an ultrametric tree.

Finally a note about possible problems arising with working with very small decimal numbers is presented, followed by the conclusions.

2.2 Massive datasets and data growth

A massive dataset can be a difficult concept to define, not only because it means different things to different people but also due to the changing landscape of the technology used to store, retrieve and analyse data. In general we can say that it is the combination of size, complexity and cost when using current technology (e.g., conventional statistical methods) that makes a dataset massive [96].

There are many areas that collect and analyse vast amounts of data, such as (but not limited to): particle physics, astronomy, geology, climatology, medicine, chemistry, genomics, banking, telecommunications, public and private enterprise. This work will look into more detail in some of these areas. Before that let us see how data is growing and with that the need for new techniques and methods that can scale well to tackle this challenge.

With the introduction of new technology and machinery that works in the digital world the inevitable outcome is the capture of data. From digital cameras to satellites, telescopes and sensor networks, the amount of data available is immense and growing continuously. In fact IDC [70,71] calculated that the digital universe by 2007 was of 281 exabytes (281 billion gigabytes) and by 2011 will be 10 times bigger than in 1996. Also, in 2007 for the first time the amount of data created, captured, or replicated exceeded available storage, a trend that will be increased by almost 50% by 2011. Further estimations show that 95% of the digital universe is unstructured data, and as a result extra effort is needed to process and locate information.

Challenges presented by this growth in data and information are many, from the economical and environmental cost to the techniques needed to access, store, visualise and analyse data. It is in the area of data analysis and specifically in exploratory data analysis where we will concentrate our efforts throughout this work.

With the increase in data complexity, scalability becomes a very significant problem when using traditional data analysis methods. It is important to highlight that not only data volume is a factor in complexity (as we have discussed in this section) but also data dimensionality, which we will discuss in section 2.3.

Note that when talking about *data* we refer to data in its raw form, in other words data that does not have any processing, and this is as opposed to information. Therefore we refer to *information* when data has been processed, organised and structured in such a way that it can help in the decision process. In this work we mention *information space* a number of times, if the data has been at least partially processed or structured. In that case we indistinguishably refer to either the *data space* or the *information space*.

2.3 The curse of dimensionality

High dimensionality is a major contributor to data complexity. Data observation with thousands of features (or more) are now common. In many situations the old assumption that the number of observations is bigger than the number of dimensions no longer holds (i.e. $N > d$). Many clustering techniques in the past assumed $N > d$, which works well in low dimensional spaces. For example some of the techniques presented in table 2.1 can deal with a large volume of data, but not necessarily with high dimensionality.

The term “curse of dimensionality” was coined by Richard Bellman [15]. It refers to the exponential growth of volume as a function of dimensionality. This is easily understood if we take a Cartesian grid of spacing 1/10 on the unit cube in 5 dimensions; we then have 10^5 grid cells. The number of grid cells increases to 10^{10} if the cube is in 10 dimensions, and to 10^{20} if in 20 dimensions ([182] p. 238). Thus, the “curse of dimensionality” refers to any problem in data analysis with a large number of variables.

Most clustering techniques are affected by the curse of dimensionality. At the heart of every clustering algorithm lies a distance or similarity used to compare the data points (elements) to compute the clusters. It is this process that suffers the “curse” of high dimensionality data.

We can see in Table 2.1 a number of well known clustering algorithms with their execution times and storage space computer complexities, where K: centroids; d: number of features; and N: input size. Some of these have been specifically developed for large datasets, especially in the context of data mining. Thus, some algorithms scale linearly with the input size. However, their performance suffers with the increase of the input dimensionality. It can be argued that some of the algorithms like DENCLUE and fuzzy clustering have shown some success dealing with both high dimensionality and large datasets, although they are still far from being completely effective.

Algorithm	Complexity
k-means	$O(NKd)$ (time) $O(N+K)$ (space)
H. C.[1]	$O(N^2)$ (time and space)
CLARA[2]	$O(K(40+K)^2 + K(N-K))$ (time)
PAM	$O(K(N-K)^2)$
BIRCH	$O(N)$ (time)
DBSCAN	$O(N\ log\ N)$ (time)
CURE	$O(N_{sample}^2\ log\ N_{sample}$ (time) $O(N_{sample})$ (space)
WaveCluster	$O(N)$ (time)
DENCLUE	$O(N\ log\ N)$ (time)
Fuzzy Clust.	$O(N)$ (time)
STING[3]	$O(K)$
CLIQUE	$O(Nd^2)$
OptiGrid	Between $O(Nd)$ and $O(Nd\ log\ N)$
ORCLUS[4]	$O(K_0^3 + K_0ND + K_0^2d^3)$ (time) $O(K_0d^2)$ (space)

1. Hierarchical clustering includes single-linkage, complete-linkage, average-linkage, etc.
2. Based on a heuristic for drawing a sample from the entire dataset [95].
3. Here K is number of cells at the bottom layer.
4. Here K_0 is the number of initial seeds.

Table 2.1: *Some clustering algorithms and their computational complexities [181, 182].*

2.4 Dimensionality reduction by random projection

It is a well known fact that traditional clustering methods do not scale well in very high dimensional spaces. A standard and widely used approach when dealing with high dimensionality is to apply a dimensionality reduction technique. This consists of finding a mapping F relating the input data from the space $\mathbb{R}^d$ to a lower-dimension feature space $\mathbb{R}^k$. We can denote this as follows:

$$F(x) : \mathbb{R}^d \rightarrow \mathbb{R}^k \tag{2.1}$$

A statistically optimal way of reducing dimensionality is to project the data onto a lower dimensional orthogonal subspace. Principal Component Analysis (PCA) is a very popular choice to do this. It uses a linear transformation to form a simplified dataset retaining the characteristics of the original data. PCA does this by means of choosing the attributes that best preserve the variance of the data. This is a good solution when the data allows these calculations, but PCA as well as other dimensionality reduction techniques remain expensive computationally speaking.

Random projection [20,39,44,57,63,104,105,169] is the finding of a low dimensional embedding of a point set, such that the distortion of any pair of points is bounded by a function of the lower dimensionality.

The theoretical support for random projection can be found in the Johnson-Lindenstrauss Lemma [92]. It states that a set of points in a high dimensional Euclidean space can be projected into a low dimensional Euclidean space such that the distance between any two points changes by a fraction of $1+\varepsilon$, where $\varepsilon \in (0,1)$.

Lemma 2.1. *For any $0 < \varepsilon < 1$ and any integer n, let k be a positive integer such that*

$$k \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \ln n. \tag{2.2}$$

Then for any set V of any points in $\mathbb{R}^d$, there is a map $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all u, $v \in V$,

$$(1-\varepsilon) \parallel u - v \parallel^2 \ \leq \ \parallel f(u) - f(v) \parallel^2 \ \leq \ (1+\varepsilon) \parallel u - v \parallel^2.$$

Furthermore, this map can be found in randomised polynomial time.

This proof was further simplified by Frankl and Maehara [64], and Dasgupta and Gupta [40], also see Achlioptas [1] and Vempala [169].

We have mentioned that the optimal way to reduce dimensionality is to orthogonalise $\mathbb{R}$, but unfortunately this is computationally expensive. Vectors having random directions might be sufficiently close to orthogonal. Additionally this helps solving the problem of data sparsity in high dimensional spaces, as we will see in chapter 4.

Thus, in random projection the original d-dimensional data is projected to a k-dimensional subspace $(k << d)$, using a random $k \times d$ matrix R. Mathematically this can be described as follows:

$$X^{RP}_{k\times N} = R_{k\times d}\ X_{d\times N} \tag{2.3}$$

where $X_{d\times N}$ is the original set with d-dimensionality and N observations.

Computationally speaking random projection is simple. Forming the random matrix R and projecting the $d \times N$ data matrix X into the k dimensions is of order $O(dkN)$. If X is sparse with c nonzero entries per column, the complexity is of order $O(ckN)$.

In fact random projection can be seen as a class of hashing function. Hashing is much faster than alternative methods because it avoids the pair-wise comparisons required for partitioning and classification. This process is depicted in a Euclidean two dimensional space in Figure 2.1, where a random vector is drawn and data points projected onto it. If two points (p, q) are close, they will have a very small $\|p - q\|$ (Euclidean metric) value; and they will hash to the same value with high probability; if they are distant, they should collide with small probability.

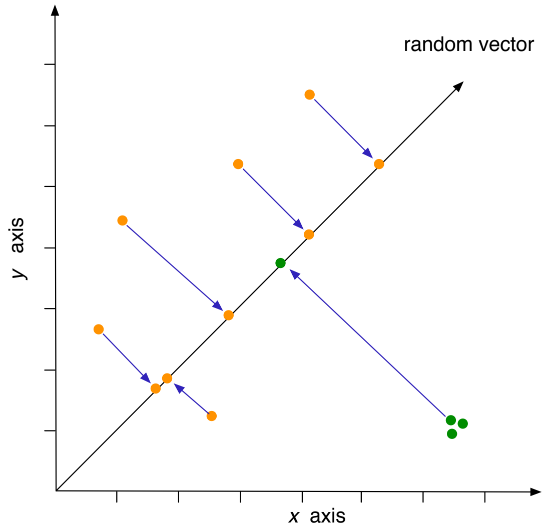


Figure 2.1: *Random projection as a hashing function.*

2.5 Clustering

Clustering is a data analysis technique that segments data into groups (subsets) called clusters, in which members of the same cluster are similar to each other. The proximity measurement (similarity) is a critical step in order to identify groups, and we will be back to it later in section 2.5.1.

When clustering we will have a number of unlabelled observations that need to be classified. We do not know anything beforehand regarding the grouping of the data. For this reason clustering is also called unsupervised classification, as opposed to supervised classification where we have a set of labels and the task is to assign the data observations to these labels (i.e. discriminant analysis).

Clustering is used for data exploration in pattern analysis, grouping, decision-making, and machine learning. Application areas include data mining, document retrieval, image segmentation, pattern classification and many others.

The literature in clustering is huge with a very active research community across many disciplines. For surveys in clustering see Cormack [34], Murtagh [120, 121], Gordon [76], Jain et al. [89], Xu and Wunsch [181], and Berkhin [16]. For books on clustering see Hartigan [84], Lorr [108], Murtagh [122], Jain and Dubes [90], Kaufman [95], Erevitt [54], Mirkin [114, 115], Xu and Wunsch [182], and Gan et al. [69].

2.5.1 Similarity measures

To group data we need a way to measure the elements and their distances relative to each other in order to decide which elements belong to a group. This is also called similarity, although on many occasions a dissimilarity measurement is also used. Note that not any arbitrary measurement is of use to us here, and in fact normally this measurement will be a metric distance, see section 2.8.1.

When working in a vector space a traditional way to measure distances is the Minkowski distances, which are defined as follows:

$$L_p(\mathbf{x}_a, \mathbf{x}_b) = (\sum_{i=1}^{n} |\mathbf{x}_{i,a} - \mathbf{x}_{i,b}|^p)^{1/p};\ \forall\ p \geq 1,\ p \in \mathbb{Z}, \tag{2.4}$$

where $\mathbb{Z}$ is the set of integers.

The Manhattan, Euclidean and Chebyshev distance (also called maximum distance) are special cases of the Minkowski distance when $p = 1,\ p = 2$ and $p \to \infty$. Thus we have the following:

- Manhattan distance:

$$L_1(\mathbf{x}_a, \mathbf{x}_b) = \sum_{i=1}^{n} |\mathbf{x}_{i,a} - \mathbf{x}_{i,b}| \tag{2.5}$$

- Euclidean distance:

$$L_2(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{\sum_{i=1}^{n} |\mathbf{x}_{i,a} - \mathbf{x}_{i,b}|^2} \tag{2.6}$$

- Chebyshev distance:

$$L_\infty(\mathbf{x}_a, \mathbf{x}_b) = max_{i=1}^{n} |\mathbf{x}_{i,a} - \mathbf{x}_{i,b}| \tag{2.7}$$

Additionally we can mention the *cosine* similarity, which gives the angle between two vectors. This is widely used in text retrieval to match vector queries to the dataset. The smaller the angle between a query vector to the document set vectors, the closer is a query to a document. The cosine similarity is defined as follows:

$$s(\mathbf{x}_a, \mathbf{x}_b) = \cos(\theta) = \frac{\mathbf{x}_a \cdot \mathbf{x}_b}{||\mathbf{x}_a||\,||\mathbf{x}_b||} \tag{2.8}$$

where $\mathbf{x}_a \cdot \mathbf{x}_b$ is the dot product and $||\cdot||$ the norm. Thus, this equation can be rewritten as follows:

$$s(\mathbf{x}_a, \mathbf{x}_b) = \cos(\theta) = \frac{\sum_{i=1}^{n} \mathbf{x}_{i,a}\ \mathbf{x}_{i,b}}{\sqrt{\sum_{i=1}^{n} \mathbf{x}_{i,a}^2\ \sum_{i=1}^{n} \mathbf{x}_{i,b}^2}} \tag{2.9}$$

Other relevant distances are the following:

- **Hellinger distance** is defined between vectors having only positive or zero elements. Let $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ and $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ be two vectors, then the Hellinger distance is defined as:

$$d_{hell}(\mathcal{X}, \mathcal{Y}) = \sqrt{\sum_{i=1}^{n} (\sqrt{x_i} - \sqrt{y_i})^2} \tag{2.10}$$

This is used to measure the similarity of two discrete or continuous probability distributions.

- **Variational distance** between two probability distributions $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ and $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$ is defined as:

$$d_{var}(\mathcal{P}, \mathcal{Q}) = \sum_{i=1}^{n} |p_i - q_i| \tag{2.11}$$

where $|\cdot|$ is absolute value.

- **Mahalanobis distance** is defined as:

$$d_{maha}(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{(\mathbf{x}_a - \mathbf{x}_b)^T S^{-1} (\mathbf{x}_a - \mathbf{x}_b)} \tag{2.12}$$

where S is any $n \times n$ positive definite matrix and $(\mathbf{x}_a - \mathbf{x}_b)^T$ is the transpose of $(\mathbf{x}_a - \mathbf{x}_b)$. The role of the matrix S is to distort the space as desired. S is the covariance matrix of the dataset.

- **Hamming distance** is defined as the number of bits which differ between two binary strings: For example the binary strings 10011011 and 10001101 have a Hamming distance of 3 bits as three is the number of dissimilar bits.

Anderberg gives a good review on measurement and metrics in [5], where their interrelationships are also discussed. Deza and Deza have produced a comprehensive list of distances in their *Encyclopaedia of Distances* [45]. Also see [69] chapter 6.

2.5.2 Clustering algorithms

Many clustering techniques and algorithms have been proposed in different application scenarios [90, 91]. Thus, it is difficult to have a strict classification of all of them. Algorithms in this area have been characterised by the probability model used, if any, or by the optimisation process used to find clusters. Other classifications are based in the type of features handled (e.g., numerical, categorical, rank data, string, graphs, etc.), or by the type of input used (e.g., pattern or similarity matrix).

One of the most popular ways to classify clustering algorithms is *hierarchical* versus *partitional* as depicted in Figure 2.2.

Within the context of this dissertation, hierarchical clustering algorithms are of special importance, since the proposed Baire distance for clustering is a hierarchical method. Thus, in section 2.5.3 we look into these techniques. Another method of importance for us is the k-means algorithm, which has been largely studied and many variations proposed. Our interest lies in the fact that we compare our methodology against the results produced by k-means.

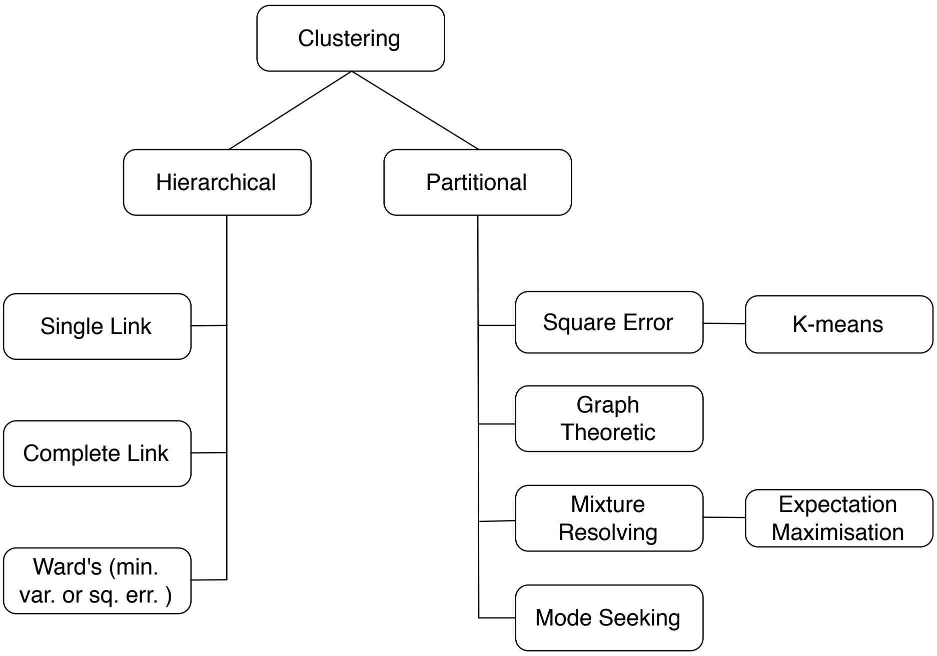


Figure 2.2: *Traditional taxonomy of clustering techniques. Two main subgroups are presented, hierarchical and partitional cluster algorithms.*

2.5.3 Hierarchical clustering algorithms

Introduction to hierarchical clustering using the single linkage agglomerative criterion

The single linkage hierarchical clustering approach outputs a set of clusters (to use graph theoretic terminology, a set of maximal connected subgraphs) at each level – or for each threshold value which produces a new partition. The following algorithm, in its general structure, is relevant for a wide range of hierarchical clustering methods which vary only in the update formula used in step 2. These methods may, for example, define a criterion of compactness in step 2 to be used instead of the connectivity criterion used here. The single linkage method with which we begin is one of the oldest methods, its origins being traced to Polish researchers in the 1950s [78]. An example is shown in Figure 2.3. Note that the dissimilarity coefficient is assumed to be symmetric, and so the clustering algorithm is implemented on half the dissimilarity matrix.

Single linkage hierarchical clustering

Input An $n(n-1)/2$ set of dissimilarities.

Step 1 Determine the smallest dissimilarity, d_{ik}.

Step 2 Agglomerate objects i and k: i.e. replace them with a new object, $i \cup k$; update dissimilarities such that, for all objects $j \neq i, k$:

$$d_{i \cup k, j} = \min \{d_{ij}, d_{kj}\}.$$

Delete dissimilarities d_{ij} and d_{kj}, for all j, as these are no longer used.

Step 3 While at least two objects remain, return to step 1.

	1	2	3	4	5
1	0	4	9	5	8
2	4	0	6	3	6
3	9	6	0	6	3
4	5	3	6	0	5
5	8	6	3	5	0

Agglomerate 2 and 4
at dissimilarity 3

	1	2U4	3	5
1	0	4	9	8
2U4	4	0	6	5
3	9	6	0	3
5	8	5	3	0

Agglomerate 3 and 5
at dissimilarity 3

	1	2U4	3U5
1	0	4	8
2U4	4	0	5
3U5	8	5	0

Agglomerate 1 and 2U4
at dissimilarity 4

	1U2U4	3U5
1U2U4	0	5
3U5	5	0

Agglomerate 1U2U4 and
3U5 at dissimilarity 5

Resulting dendrogram

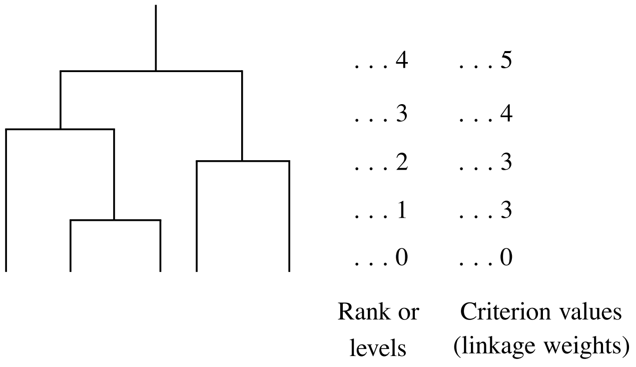


Figure 2.3: *Construction of a dendrogram by the single linkage method.*

Equal dissimilarities may be treated in an arbitrary order. There are precisely $n-1$ agglomerations in step 2 (allowing for arbitrary choices in step 1 if there are identical dissimilarities). It may be convenient to index the clusters found in step 2 by $n+1$, $n+2$, ..., $2n-1$, or an alternative practice is to index cluster $i \cup k$ by the lower of the indices of i and k.

The title *single linkage* arises since, in step 2, the interconnecting dissimilarity between two clusters ($i \cup k$ and j) or components is defined as the least interconnecting dissimilarity between a member of one and a member of the other. Other hierarchical clustering methods are characterised by other functions of the interconnecting linkage dissimilarities.

Compared to other hierarchical clustering techniques, the single linkage method can give rise to a notable disadvantage for summarising interrelationships. This is known as *chaining*. An example is to consider four subject-areas, which it will be supposed are characterised by certain attributes: computer science, statistics, probability, and measure theory. It is conceivable that "computer science" is connected to "statistics" at some threshold value, "statistics" to "probability", and "probability" to "measure theory", thereby giving rise to the fact that "computer science" and "measure theory" find themselves, undesirably, in the same cluster. This is due to the intermediaries "statistics" and "probability".

As early as the 1970s, it was held that about 75% of all published work on clustering employed hierarchical algorithms [21]. Interpretation of the information contained in a dendrogram is often of one or more of the following kinds:

- set inclusion relationships,
- partition of the object-sets, and
- significant clusters.

Much early work on hierarchical clustering was in the field of biological taxonomy, from the 1950s and more so from the 1960s onwards. The central reference in this area, the first edition of which dates from the early 1960s, is [161]. One major interpretation of hierarchies has been the evolution relationships between the organisms under study. It is hoped, in this context, that a dendrogram provides a sufficiently accurate model of underlying evolutionary progression.

The most common interpretation made of hierarchical clustering is to derive a partition: a line is drawn horizontally through the hierarchy, to yield a set of classes. These clusters are precisely the connected components in the case of the single linkage method. A line drawn just above rank 3 (or criterion value 4) on the dendrogram in Fig. 2.3 yields classes $\{1, 2, 4\}$ and $\{3, 5\}$. Generally the choice of where "to draw the line" is arrived at on the basis of large changes in the criterion value. However the changes in criterion value increase (usually) towards the final set of agglomerations, which renders the choice of best partition on this basis difficult. Since every line drawn through the dendrogram defines a partition, it may be expedient to choose a partition with convenient features (number of classes, number of objects per class).

A further type of interpretation is to dispense with the requirement that the classes chosen constitute a partition, and instead detect maximal (i.e. disjoint) clusters of interest at varying levels of the hierarchy. Such an approach is used by [146] in a clustering of colours based on semantic attributes. Lerman [101] developed an approach for finding significant clusters at varying levels of a hierarchy, which has been widely applied. See also Murtagh [128] which, based on a wavelet transform on a dendrogram, is used to find the important, i.e. best approximating, clusters.

In summary, a dendrogram provides a resume of many of the proximity and classificatory relationships in a body of data. It is a convenient representation which answers such questions as: "How many groups are in this data?", "What are the salient interrelationships present?". But it should be stressed that differing answers can feasibly be provided by a dendrogram for most of these questions, depending on the application.

2.5.4 Agglomerative hierarchical clustering algorithms

In the previous section, a general agglomerative algorithm was discussed. A wide range of these algorithms have been proposed at one time or another. Hierarchical agglomerative algorithms may be conveniently broken down into two groups of methods. The first group is that of linkage methods – the single, complete, weighted and unweighted average linkage methods. These are methods for which a graph representation can be used. [161] may be consulted for many other graph representations of the stages in the construction of hierarchical clustering.

The second group of hierarchical clustering methods are methods which allow the cluster centres to be specified (as an average or a weighted average of the member vectors of the cluster). These methods include the centroid, median and minimum variance methods.

The latter may be specified either in terms of dissimilarities, alone, or alternatively in terms of cluster centre coordinates and dissimilarities. A very convenient formulation, in dissimilarity terms, which embraces all the hierarchical methods mentioned so far, is the *Lance-Williams dissimilarity update formula*. If points (objects) i and j are agglomerated into cluster $i \cup j$, then we must simply specify the new dissimilarity between the cluster and all other points (objects or clusters). The formula is:

$$d(i \cup j, k) = \alpha_i d(i,k) + \alpha_j d(j,k) + \beta d(i,j) + \gamma \mid d(i,k) - d(j,k) \mid \qquad (2.13)$$

where α_i, α_j, β, and γ define the agglomerative criterion. Values of these are listed in the second column of Table 2.2.

In the case of the single link method, using $\alpha_i = \alpha_j = \frac{1}{2}$, $\beta = 0$, and $\gamma = -\frac{1}{2}$ gives us:

$$d(i \cup j, k) = \frac{1}{2} d(i,k) + \frac{1}{2} d(j,k) - \frac{1}{2} \mid d(i,k) - d(j,k) \mid \qquad (2.14)$$

which, it may be verified by taking a few simple examples of three points, i, j, and k, can be rewritten as:

$$d(i \cup j, k) = \min \{d(i,k), d(j,k)\} \qquad (2.15)$$

This was exactly the update formula used in the agglomerative algorithm given in section 2.5.3. Using other update formulas, as given in column 2 of Table 2.2, allows the other agglomerative methods to be implemented in a very similar way to the implementation of the single link method.

In the case of the methods which use cluster centres, we have the centre coordinates (in column 3 of Table2.2) and dissimilarities as defined between cluster centres (column 4 of Table 2.2). The Euclidean distance must be used for equivalence between the two approaches. In the case of the *median method*, for instance, we have the following (cf. Table 2.2).

Let $\mathbf{a}$ and $\mathbf{b}$ be two points (i.e. m-dimensional vectors: these are objects or cluster centres) which have been agglomerated, and let $\mathbf{c}$ be another point. From the Lance-Williams dissimilarity update formula, using squared Euclidean distances, we have:

$$\begin{aligned} d^2(a \cup b, c) &= \tfrac{d^2(a,c)}{2} + \tfrac{d^2(b,c)}{2} - \tfrac{d^2(a,b)}{4} \\ &= \tfrac{\|\mathbf{a}-\mathbf{c}\|^2}{2} + \tfrac{\|\mathbf{b}-\mathbf{c}\|^2}{2} - \tfrac{\|\mathbf{a}-\mathbf{b}\|^2}{4}. \end{aligned} \tag{2.16}$$

The new cluster centre is $(\mathbf{a}+\mathbf{b})/2$, so that its distance to point $\mathbf{c}$ is

$$\|\mathbf{c} - \frac{\mathbf{a}+\mathbf{b}}{2}\|^2. \tag{2.17}$$

That these two expressions lead to an identical outcome is readily verified. The correspondence between these two perspectives on the one agglomerative criterion is similarly proved for the centroid and minimum variance methods.

The single linkage algorithm discussed in section 2.5.3, duly modified for the use of the Lance-Williams dissimilarity update formula, is applicable for all agglomerative strategies. The update formula listed in Table 2.2 is used in step 2 of the algorithm.

For cluster centre methods, and with suitable alterations for graph methods, the following algorithm is an alternative to the general dissimilarity based algorithm. The latter may be described as a "stored dissimilarities approach" [5].

Hierarchical clustering methods (and aliases)	Lance and Williams dissimilarity update formula	Coordinates of centre of cluster, which agglomerates clusters i and j	Dissimilarity between cluster centres g_i and g_j
Single link (nearest neighbour)	$\alpha_i = 0.5$ $\beta = 0$ $\gamma = -0.5$ (More simply: $min\{d_{ik}, d_{jk}\}$)		
Complete link (diameter)	$\alpha_i = 0.5$ $\beta = 0$ $\gamma = 0.5$ (More simply: $max\{d_{ik}, d_{jk}\}$)		
Group average (average link, UPGMA)	$\alpha_i = \frac{\|i\|}{\|i\|+\|j\|}$ $\beta = 0$ $\gamma = 0$		
McQuitty's method (WPGMA)	$\alpha_i = 0.5$ $\beta = 0$ $\gamma = 0$		
Median method (Gower's, WPGMC)	$\alpha_i = 0.5$ $\beta = -0.25$ $\gamma = 0$	$\mathbf{g} = \frac{\mathbf{g}_i+\mathbf{g}_j}{2}$	$\\|\mathbf{g}_i - \mathbf{g}_j\\|^2$
Centroid (UPGMC)	$\alpha_i = \frac{\|i\|}{\|i\|+\|j\|}$ $\beta = -\frac{\|i\|\|j\|}{(\|i\|+\|j\|)^2}$ $\gamma = 0$	$\mathbf{g} = \frac{\|i\|\mathbf{g}_i+\|j\|\mathbf{g}_j}{\|i\|+\|j\|}$	$\\|\mathbf{g}_i - \mathbf{g}_j\\|^2$
Ward's method (minimum variance, error sum of squares)	$\alpha_i = \frac{\|i\|+\|k\|}{\|i\|+\|j\|+\|k\|}$ $\beta = -\frac{\|k\|}{\|i\|+\|j\|+\|k\|}$ $\gamma = 0$	$\mathbf{g} = \frac{\|i\|\mathbf{g}_i+\|j\|\mathbf{g}_j}{\|i\|+\|j\|}$	$\frac{\|i\|\|j\|}{\|i\|+\|j\|}\\|\mathbf{g}_i - \mathbf{g}_j\\|^2$

Notes: $\mid i \mid$ is the number of objects in cluster i; $\mathbf{g}_i$ is a vector in m-space (m is the set of attributes), – either an initial point or a cluster centre; $\|.\|$ is the norm in the Euclidean metric; the names UPGMA, etc. are due to [161]; finally, the Lance and Williams recurrence formula is:

$$d_{i \cup j,k} = \alpha_i d_{ik} + \alpha_j d_{jk} + \beta d_{ij} + \gamma \mid d_{ik} - d_{jk} \mid .$$

Table 2.2: *Specifications of seven hierarchical clustering methods [122].*

Stored data approach

Step 1 Examine all interpoint dissimilarities, and form cluster from two closest points.

Step 2 Replace two points clustered by representative point (centre of gravity) or by cluster fragment.

Step 3 Return to step 1, treating clusters as well as remaining objects, until all objects are in one cluster.

In steps 1 and 2, "point" refers either to objects or clusters, both of which are defined as vectors in the case of cluster centre methods. This algorithm is justified by storage considerations, since we have $O(n)$ storage required for n initial objects and $O(n)$ storage for the $n-1$ (at most) clusters. In the case of linkage methods, the term "fragment" in step 2 refers (in the terminology of graph theory) to a connected component in the case of the single link method and to a clique or complete subgraph in the case of the complete link method. The overall complexity of the above algorithm is $O(n^3)$: the repeated calculation of dissimilarities in step 1, coupled with $O(n)$ iterations through steps 1, 2 and 3. Note however that this does not take into consideration the extra processing required in a linkage method, where "closest" in step 1 is defined with respect to graph fragments.

While the stored data algorithm is instructive, it does not lend itself to efficient implementations. The reciprocal nearest neighbour and mutual nearest neighbour algorithms have to be used in practice for implementing agglomerative hierarchical clustering algorithms: see e.g. [120, 122]

2.5.5 Partition clustering algorithms

In Figure 2.2 we see that there are many partitional clustering algorithms. Our interest lies in the k-means method because we use it on a number of occasions as a basis to compare the result obtained with the Baire methodology. Now we briefly explain how the k-means algorithm works.

K-means

K-means is a major clustering algorithm technique presented in various forms, first introduced by MacQueen in 1967 [110] and further developed by Hartigan and Wong [83, 84]. This algorithm groups data by minimising the sum of the squares of distances between the data points and the cluster centroid. Suppose we have the dataset $X = \{x_1, x_2, x_3, .., x_n\}$ consisting of n observations of a d-dimensional variable X, where x_1 represents the first observation. The goal of this algorithm is to partition the set X into a number of $K > 1$ of non-overlapping clusters, where at the moment we assume the value of K is given. The algorithm has two main iterative steps; first is to update clusters according to the minimum distance rule, second is to update centroids as the centres of gravity of the clusters.

This algorithm can be expressed mathematically as follows: the jth cluster is represented by a "cluster prototype" μ_j in $\mathbb{R}^d$. The algorithm finds z_i and μ_j that minimise the function $J_{k-means}$.

$$J_{k-means} = \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|^2 \tag{2.18}$$

Equation 2.18 can be rewritten as:

$$J_{k-means} = \sum_{i=1}^{n} \sum_{j=1}^{k} I(z_i = j) \|x_i - \mu_{z_i}\|^2 \tag{2.19}$$

where $I(z_i = j)$ is the indicator function that takes different values, one if $z_i = j$ is true; and zero otherwise. To optimise equation 2.19 we assume that all μ_j are specified. Then the values of z_i that minimise the equation are given by:

$$z_i = arg\ min_j \|x_i - \mu_j\|^2 \tag{2.20}$$

Fixing z_i, the optimal μ_j can be found by differentiating $J_{k-means}$ with respect to μ_j and setting the derivatives to zero. Then we have the following:

$$\mu_j = \frac{\sum_{j=1}^{k} I(z_i = j) x_j}{\sum_{j=1}^{k} I(z_i = j)} = \frac{\sum_{i=1}^{n},\ z_i = i\ \ x_j}{No.\ of\ i\ with\ z_i = j} \tag{2.21}$$

With an initial guess on u_j (which can be random), the k-means algorithm iterates between Equations 2.20 and 2.21, which decreases the k-means objective function until the local minimum is reached. Then the resulting z_i and μ_j are the clustering solutions [100].

Some problems with k-means are related to the detection of clusters that are not spherical. It also requires a good initialisation mechanism to avoid a poor local minimum. Finally, k-means requires an initial number of clusters, that in many occasions is unknown.

2.6 Other clustering techniques

Many modern clustering techniques focus on large scale data, in [182] p. 215 these are classified as follows:

- Random sampling
- Data condensation
- Density-based approaches
- Grid-based approaches
- Divide and conquer
- Incremental learning

From the point of view of this work density and grid based approaches are of interest because they either look for data densities or split the data space into cells when looking for groups. This can be related in some of their characteristics to the Baire distance introduced later in this chapter and used throughout this work. Thus, in this section we now have a brief look at these two approaches, density and grid based.

2.6.1 Grid-based clustering algorithms

The main idea here is to use a grid like structure to split the information space, separating the dense grid regions from the less dense ones to form groups.

In general, a typical approach within this category will consist of the following steps [77]:

1. Creating a grid structure, i.e. partitioning the data space into a finite number of non-overlapping cells.
2. Calculating the cell density for each cell.
3. Sorting of the cells according to their densities.
4. Identifying cluster centres.
5. Traversal of neighbour cells.

Some of the most important algorithms within this category are the following:

- **STING:** STatistical INformation Grid-based clustering was proposed by Wang et al. in [173] divides the spatial area into rectangular cells represented by a hierarchical structure. The root is at hierarchical level 1, its children at level 2, and so on. This algorithm has a computational complexity of $O(K)$, where K is the number of cells in the bottom layer. This implies that scaling this method to higher dimensional spaces is difficult [86]. For example, if in high dimensional data space each cell has four children, then the number of cells in the second level will be 2^d, where d is the dimensionality of the database.
- **OptiGrid:** Optimal Grid-Clustering was introduced by Hinneburg and Keim [86] as an efficient algorithm to cluster high-dimensional databases with noise. It uses data partitioning based on divisive recursion by multidimensional grids, focusing on separation of clusters by hyperplanes. A cutting plane is chosen which goes through the point of minimal density, therefore splitting two dense half-spaces. This process is applied recursively with each subset of data. This algorithm is hierarchical, with time complexity of $O(N \cdot d)$ [69] pp. 210–212.
- **GRIDCLUS:** proposed by Schikute in [154] is a hierarchical algorithm for clustering very large datasets. It uses a multidimensional data grid to organise the space surrounding the data values rather than organise the data themselves. Thereafter patterns are organised into blocks, which in turn are clustered by a topological neighbour search algorithm. Five main steps

are involved in the GRIDCLUS method: (a) insertion of points into the grid structure, (b) calculation of density indices, (c) sorting the blocks with respect to their density indices, (d) identification of cluster centres, and (e) traversal of neighbour blocks.

- **WaveCluster:** this clustering technique proposed in [157] defines a uniform two dimensional grid on the data and represents the data points in each cell by the number of points. Thus the data points become a set of grey-scale points, which is treated as an image. Then the problem of looking for clusters is transformed into an image segmentation problem, where wavelets are used to take advantage of their multi-scaling and noise reduction properties. The basic algorithm is as follows: (a) create a data grid and assign each data object to a cell in the grid, (b) apply the wavelet transform to the data, (c) use the average sub-image to find connected clusters (i.e. connected pixels), and (d) map the resulting clusters back to the points in the original space. Note that the wavelet transform also has been used for clustering by other authors, see [131,132].

Additional information about grid-based clustering can be found in the following works [28,69,141,182].

2.6.2 Density-based clustering algorithms

Density-based clusters are defined as a dense region of points, which are separated by low-density regions. Therefore clusters can have an arbitrary shape and the points in the cluster may be arbitrarily distributed. An important advantage of this methodology is that only one scan of the dataset is needed and it can handle noise effectively. Furthermore the number of clusters to initialise the algorithm is not required.

Some of the most important algorithms in this category include the following:

- **DBSCAN:** Density-Based Spatial Clustering of Applications with Noise was proposed by Ester et al. [55] to discover arbitrarily shaped clusters. Since it finds clusters based on density it does not need to know the number of clusters at initialisation time. This algorithm has been widely used and counts with many variations (e.g., GDBSCAN [153], PDBSCAN [184], and DBCluC [186]).

- **BRIDGE:** proposed by Dash et al. [42] uses a hybrid approach integrating k-means to partition the dataset into k clusters, and then density-based algorithm DBSCAN is applied to each partition to find dense clusters.
- **DBCLASD:** Distribution-Based Clustering of LArge Spatial Databases [183] assumes that data points within a cluster are uniformly distributed. The produced cluster is defined in terms of the nearest neighbour distance.
- **DENCLUE:** DENsity based CLUstering aims to cluster large multimedia data. It can find arbitrarily shaped clusters and at the same time deals with noise in the data. This algorithm has two steps, first a pre-cluster map is generated, the data is divided in hypercubes where only the populated are considered. The second step takes the highly populated cubes and cubes that are connected to a highly populated cube to produce the clusters. For a detailed presentation of these steps see [87].
- **CUBN:** it has three steps. First an erosion operation is carried out to find border points. Second, the nearest neighbour method is used to cluster the border points. Finally, the nearest neighbour is used to cluster the inner points. This algorithm is capable of finding nonspherical shapes and wide variations in size. Its computational complexity is $O(n)$ with n being the size of the dataset. For a detailed presentation of this algorithm see [172].

2.7 Evaluating cluster quality

We have seen a number of methods that allow for creating clusters. A natural question that arises is how to evaluate the clusters produced. Several validity criteria have been developed in the literature. They are mainly classified as external, internal or relative criteria [90]. In the external approach, groups assembled by a clustering algorithm are compared to a previously accepted partition on the testing dataset. In the internal approach, clustering validity is evaluated using data and features contained in the dataset. The relative approach searches for the best clustering result from an algorithm and compares it with a series of predefined clustering schemes. In all cases, validity indexes are constructed to evaluate proximity among objects in a cluster or proximity among resulting clusters. For further information see Jain and Dubes [90] where chapter four is dedicated to cluster validity, [69] chapter 17, and [182] chapter 10. For relevant papers in this area see [22, 41, 81, 82]. Another interesting study is [3] where 22 indices are compared, and when adjusted for chance agreement it can be shown that many indices are similar. Also see [170] for additional information regarding correction for chance agreement.

Figure 2.4 depicts a taxonomy of the cluster validity indices. These can be separated into statistical and non-statistical. The statistical indices include the external and internal criteria and the non-statistical the relative criteria.

Table 2.3 [69, 90] shows some of the equations for cluster validation indices. Let P be a pre-specified partition of dataset X with n data points, and let C be a clustering partition from a clustering algorithm independent of P. Then by comparing C and P we obtain the evaluation of C by external criteria. Considering a pair of points x_i and x_j of X, there are four cases how x_i and x_j can be placed in C and P. We consider the following. **Case *a***: is the number of pairs of data points which are in the same clusters of C and P; **Case *b***: is the number of pairs of data points which are in the same clusters of C, but different clusters P; **Case *c***: is the number of pairs of data points which are in different clusters of C, but the same clusters P; and **case *d***, is where the number of pairs of data points which are in different clusters of C, and different clusters of P. Finally, let M be the total number of pairs of data points in the dataset, then $M = a + b + c + d = \frac{n(n-1)}{2}$.

In particular we are interested in the external criteria because we use a modified version of the Jaccard index later in this work, see section number 7.3.3.

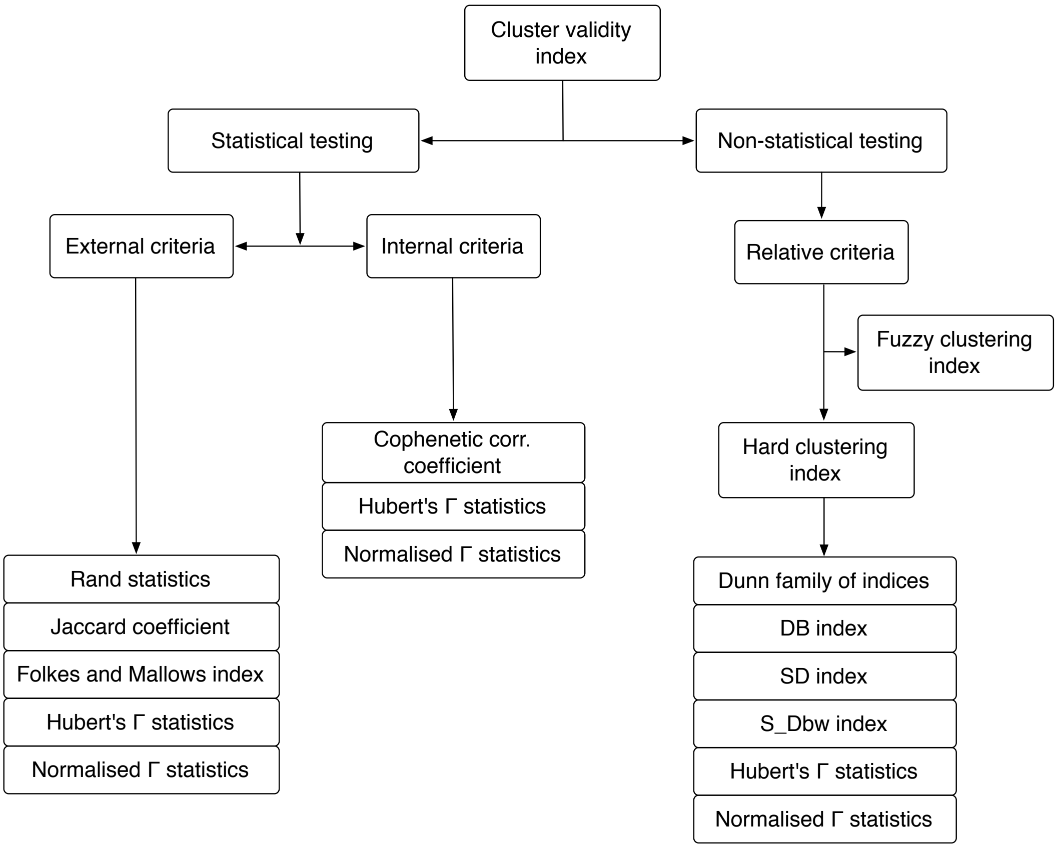


Figure 2.4: *Diagram of the cluster validity indices [69] p. 300.*

Index name	Formula
Rand statistics	$R = \frac{a+d}{M}$
Jaccard coefficient	$J = \frac{a}{a+b+c}$
Folkes and Mallows index	$FM = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$
Hubert's Γ statistics	$\Gamma = \frac{M_a - m_1 m_2}{\sqrt{m_1 m_2 (M - m_1)(M - m_2)}}$

Table 2.3: *Some external criteria indices to measure the degree of similarity between clusters. where $m_1 = (a+b)$ and $m_2 = (b+c)$; R, J, FM, and $\Gamma \in [0, 1]$.*

2.8 Baire distance or longest common prefix

2.8.1 Metric and ultrametric spaces

Our purpose consists of mapping data into an ultrametric space, searching for an ultrametric embedding, or ultrametrisation [168]. Actually, inherent ultrametricity leads to an identical result with most commonly used agglomerative criteria [122]. Furthermore, data coding can help greatly finding how inherently ultrametric data is [123].

A metric space (X, d) consists of a set X on which is defined a *distance function* d which assigns to each pair of points of X a distance between them, and satisfies the following four axioms for any triplet of points x, y, z:

1. $\forall x, y \in X, d(x, y) \geq 0$ (positiveness);
2. $\forall x, y \in X, d(x, y) = 0$ *iff* $x = y$ (reflexivity);
3. $\forall x, y \in X, d(x, y) = d(y, x)$ (symmetry);
4. $\forall x, y, z \in X, d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

When talking about an *ultrametric space* we need to consider the "strong triangular inequality" or *ultrametric inequality* defined as:

$$d(x,z) \leq max\ \{d(x,y),\ d(y,z)\},$$

this in addition to the positivity, reflexivity and symmetry properties for any triple of point $x, y, z \in X$.

Furthermore, ultrametric space implies respect for a range of stringent properties. For example, the triangle formed by any triplet is necessarily isosceles, with the two large sides equal; or is equilateral. Every point of a circle in an ultrametric space is a centre of the circle. Two circles of the same radius, that are not disjoint, are overlapping [101]. Additionally, an ultrametric is a distance that is defined strictly on a tree, which is a property that is very useful in classification.

2.8.2 Ultrametric Baire space and distance

A Baire space consists of countably infinite sequences with a metric defined in terms of the longest common prefix: the longer the common prefix, the closer a pair of sequences. What is of interest to us here is this longest common prefix metric, which we call the Baire distance [33,116,130].

The longest common prefixes at issue here are those of precision of any value. For example, let us consider two such values, x_i and y_j, which, when the context easily allows it, we will call x and y. We take x and y to be bounded by 0 and 1. Each are of some precision, and we take the integer $|K|$ to be the maximum precision.

Thus we consider ordered sets x_k and y_k for $k \in K$. In line with our notation, we can write x_k and y_k for these numbers, with the set K now ordered. So, $k = 1$ is the first decimal place of precision; $k = 2$ is the second decimal place; . . . ; $k = |K|$ is the $|K|$ th decimal place. The cardinality of the set K is the precision with which a number, x_k , is measured.

Consider as examples $x_3 = 0.478$; and $y_3 = 0.472$. Start from the first decimal position. For $k = 1$, we find $x_1 = y_1 = 4$. For $k = 2$, $x_2 = y_2 = 7$. But for $k = 3$, $x_3 \neq y_3$.

We now introduce the following distance (case of vectors x and y, with 1 attribute, hence unidimensional):

$$d_{\mathcal{B}}(x_K, y_K) = \begin{cases} 1 & \text{if } \; x_1 \neq y_1 \\ \inf \; 2^{-n} & \quad x_n = y_n, \quad 1 \leq n \leq |K| \end{cases} \tag{2.22}$$

We call this $d_{\mathcal{B}}$ value Baire distance, which can be shown to be an ultrametric [123–125,127,130] distance.

Note that the base 2 is given for convenience. When dealing with binary data 2 is the chosen base. When working with real numbers the base can be redefined to 10 if needed.

The longest common prefix can be implemented following the Java pseudocode presented in Algorithm 1.

Algorithm 1 Computes the longest common prefix for two strings.

```
$x \leftarrow String\ A$
$y \leftarrow String\ B$
$N$ = Math.min( x.length(), y.length() )
for $i = 0; i < N; i$++ do
    if ( x.chart($i$) != y.charAt($i$) ) then
        return x.substring( 0, $i$ )
    end if
    return x.substring( 0, $N$ )
end for
```

This works when the first character of the two strings are different, when one or both strings lengths is 0, and when one is a prefix of the other. A Java implementation of this algorithm and other related Java methods are shown in Appendix A.

Note that the distance defined by means of the longest common prefix between two strings will also give us a way to assess how close these two strings are. In particular when working with numbers it can be seen that the Baire distance is embedded in a 10-way tree (in the case of decimal numbers) which presents many advantages when classifying data. Namely data can be organised, stored and accessed very efficiently in a tree.

2.9 A note on working with very small decimal numbers

Arising out of section 2.8.2 we may want to normalise our data and then proceed to the analysis of very small decimal numbers. In doing so we must be careful for reasons which we will now describe.

Let us take the case of the Java programming language. This language supports two primitive floating point types: *float* and *double*, which are implementations of the IEEE 754 standard. This standard represents floating point numbers as base 2 decimal numbers in scientific notation. The problem arises since floating point arithmetic is not exact.

For example, some numbers can be represented exactly as a binary. This is the case for 0.5 that equals 2^{-1} and 0.75 that equals $2^{-1} + 2^{-2}$.

Other numbers can not be represented exactly in base 2, such as 0.1. In this case we have that 0.1 equals $2^{-4} + 2^{-5} + 2^{-8} + ...$, which is a never ending series. Therefore the value 0.1 is not exactly representable in IEEE 754 floating point arithmetic.

Let us see the rounding problem presented in Listing 2.1. That simple calculation should be 2.9, but in fact the result is 2.900000000000001.

```java

double s = 0;

for (int i = 0; i < 29; i++){
    s += 0.1;
}

System.out.println(s);
```

Listing 2.1: *Java floating point rounding problem – addition [75].*

Another example can be seen in Listing 2.2, the result from the multiplication in line "2" is $29 * 0.01 = 0.29$. When casting from floating point to integer errors can be very serious. For example in line "6" the result should be 29 but in fact we get 28.

```java
1
2    double d = 29.0 * 0.01;
3
4    System.out.println(d);
5
6    System.out.println((int) (d * 100));
```

Listing 2.2: *Java floating point rounding problem [75].*

This kind of error seems trivial, but the consequences are far-reaching. Overon in [139] provides a nice description of floating point arithmetic, where he also presents errors that had serious outcomes. In particular it can be mentioned that in 1991 during the Gulf War a Patriot defence system failed because of time conversion that involved floating point arithmetic.

Perhaps even more dramatic was the self-destruction of the European rocket Ariane 5 on the morning of June 4, 1996, which was produced by an error in the floating point calculations. In this case the two inertial reference systems ceased to work because of an operand error when converting a 64-bits floating point number to a 16-bit signed integer [158].

In this Thesis a number of times we work with very small numbers, in particular when normalising data and in the random projection processes. These operations involve multiplying matrices of small numbers. In the Java programming language case the *BigDecimal* class is used to avoid the foregoing problems.

It is important to highlight that not only the Java programming language suffers from the floating point arithmetic problem. In fact this standard is in use by most programming languages. For additional information see [93,118,144].

2.10 Summary

There is a huge amount of data in today's world and increasing, in fact currently, we are in a situation where there is not enough storage to collect all the data produced. Not only increase in data volume is a problem but also the increase in features of this data. Thus, the big challenge to the data analysis community is to introduce new algorithms that deal with these problems with ease.

This work is about a new distance that can be used to produce a hierarchy and in turn for matching, searching and clustering. Thus in this chapter we have looked at various clustering techniques with particular attention given to hierarchical methods. Additionally, we presented some techniques that are based on density, that look for low-density regions to separate groups; also, grid-based algorithms were described, which split the data space into a finite number of non-overlapping cells to identify the more dense regions of the data for later forming of groups.

In this chapter we have introduced the Baire distance, or longest common prefix distance. This distance is an ultrametric that can be seen as a hierarchical classification method that only needs one pass over the data to produce groups. This is very advantageous computationally speaking, consequently very well suited for clustering large datasets.

Often the clustering process takes place after reducing the number of features in the original data, in other words, after mapping the dataset to a lower dimensionality. This has many advantages, for the most part related with volume of data to process. Here we have shown why this process is necessary, not only mentioning how Principal Component Analysis works, but also introducing the random projection method.

Part I

Applications to Science

Introduction to Part I

In **Chapter** 2 we have introduced the Baire (ultra)metric algorithm for hierarchical clustering.

In this part we deal with the application of the Baire (ultra)metric to science data. First in **Chapter** 3 we look into mapping photometric redshift to spectrometric redshift. We use the Baire distance to obtain clusters that are compared against the clusters obtained with the *k*-means clustering algorithm.

In **Chapter** 4 we cluster chemical compounds databases on multidimensional vectors. In this case we first use the random projection technique to reduce the data dimensionality. Comparisons are made against clusters obtained with the *k*-means algorithm.

In **Chapter** 5 we look into genomic data using a modified Baire distance for clustering, namely the longest common substring. In this chapter our aim is to analyse Giardia hypothetical proteins in order to obtain clusters that can be used as targets for drug discovery. Thus we start by explaining the Giardia parasite, then the embedding of the genomic information into the 20 most common amino acids is described. This is followed by an explanation about hypothetical proteins and their characterisation in Giardia's genome. We follow by introducing suffix trees and their use in genomic string analysis. The final section is dedicated to explaining the clusters obtained using the longest common substring and the single-linkage clustering algorithm.

Chapter 3

Application to Astronomy: SDSS Redshift Calibration

3.1 Introduction

IN this chapter we applied the Baire distance to spectrometric and photometric redshifts from the Sloan Digital Sky Survey. We look specifically into four parameters: right ascension (RA), declination (DEC), spectrometric redshift (z_{spec}) and photometric redshift (z_{phot}). RA and DEC give the position of an astronomical object in the sky. Spectrometric and photometric parameters are two different values obtained from measuring the redshifts. On the one hand we have the spectrometric technique that uses the spectrum of electromagnetic radiation (including visible light) which radiates from stars and other celestial objects. On the other hand we have the photometric technique that uses a faster and economical way of measuring redshifts.

We follow by introducing a cluster-wise mapping from $z_{spec} \to z_{phot}$. Finally, clusters of the individual signals are built and presented.

3.2 Astronomical data and information

Astronomy has a long history in observing, registering, and analysing large quantities of data. Driven by technical advances in sensors, telescopes, satellites, and computer technology the rate at which data is acquired is increasing tremendously.

For a better understanding of the challenges presented by the massive amount of data that astronomers need to manage, it is important to understand the sources of this data. Different kinds of "telescopes" are needed to gather the information emitted by celestial bodies, information that is collected through observation on the electromagnetic spectrum as shown in Figure 3.1.

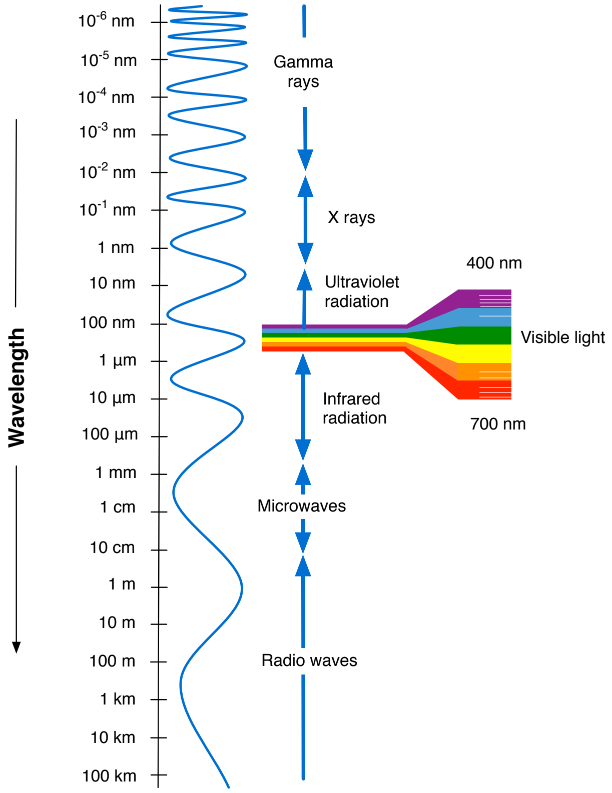


Figure 3.1: *Electromagnetic spectrum in different wavelengths [66].*

The following list includes a brief description of the kind of instrument used to collect this astronomical data in different wavelengths [112].

- **Radio waves:** this is the longest wavelength, detectable by large radio dishes like the Very Large Array [171] in New Mexico, the Arecibo radio telescope [7] in Puerto Rico, and the Parkes Observatory in Australia, just to name a few. The radio sky is dominated mainly by gas clouds.
- **Submillimeter radiation:** instruments that study submillimeter radiation are either satellites or located on the earth's driest and highest places, like the submillimeter array in Mauna Kea, Hawaii. In this band complex molecules in dark clouds are studied.
- **Far infrared light:** can only be seen from space observatories like the Spitzer Space Telescope [163]. Sources of infrared light are embedded in dense regions of gas and dust.
- **Visible and near infrared light:** this is from where "traditional" astronomy obtains its data. The instruments used to obtain information for this wavelength are the traditional astronomical observatories (e.g. Palomar Observatory [140]) and the Hubble Space Telescope.
- **Ultraviolet light:** this light is too blue for humans to see. The atmosphere blocks most ultraviolet radiation, therefore observations must take place mainly in space with instruments such as the NASA GALEX [68] satellite.
- **X–Rays:** this is the light beyond ultraviolet in the spectrum. Space telescopes like NASA's Chandra X–Ray [27] Observatory and the European Space Agency's XMM–Newton [180], can detect black holes and the X–ray galaxy chemical composition in this wavelength.
- **Gamma rays:** are studied with ground-based telescopes, satellites, and balloons. Gamma bursts, which result from some of the most violent explosions in the universe, can be observed through instruments like the recently launched Swift gamma–ray burst satellite [165].

Many digital surveys and archives already exist, and integration of these data sources is well advanced under the umbrella of the International Virtual Observatory Alliance (IVOA) [88]. Brunner et al. [24] discuss massive datasets in astronomy and some of the installations to acquire astronomical data, also see Reshetnikov [147]. A comprehensive list of observatories and telescopes can be found at [9, 11], which is classified into ground-based [10] and space-borne telescopes [12]. Additionally, a list of astronomical survey projects is available at [8].

3.3 Doppler effect and redshift

Light from moving objects will appear to have different wavelengths depending on the relative motion of the source and the observer. On the one hand we have that if an object is moving towards an observer, the light waves will be compressed from the observer viewpoint, then the light will be shifted to a shorter wavelength or it will appear to be blue shifted. On the other hand if the object is moving away from the observer, the light wavelength will be expanding, thus red shifted. This is also called Doppler effect (or Doppler shift) named after the Austrian physicist Christian Doppler, who first described this phenomenon in 1845.

A very important piece of information obtained in cosmology from the Doppler shift is to know if an object is moving towards or away from us, and the speed at which this is happening.

Spectrometric measurement of redshift: under certain conditions all atoms can be made to emit light, doing so at particular wavelengths, which can be measured accurately. Chemical compounds are a combination of different atoms working together. Thus, when measuring the precise wavelength at which a particular chemical radiates we are effectively obtaining a signature of this chemical. These emissions are seen as lines (emission or absorption) in the electromagnetic spectrum. For example, hydrogen is the simplest chemical element with atomic number 1, and also is the most abundant chemical in the universe. Hydrogen has emission lines at 6562.8 Å, 4861.3 Å, 4340 Å, 4102.8 Å, 4102.8 Å, 3888.7 Å, 3834.7 Å and 3798.6 Å (where Å is an Angstrom equal to 10^{-10}m). If the spectrum of a celestial body has emission lines in these wavelengths we can conclude that hydrogen is present there.

Photometric measurement of redshift: sometimes obtaining spectrometric measurements can be very difficult due to the large number of objects to observe or because the signal is too weak for the current spectrometric techniques. A redshift estimate can be obtained using large/medium band photometry instrumentation instead of spectrometric. This technique is based on the identification of strong spectral features. This is much faster than spectrometric measurement but also of lesser quality and precision [58].

Hence the context of our clustering work is to see how well the more easily obtained photometric redshifts can be used as estimates for the spectrometric redshifts that are obtained with greater cost. We limit our work here to the fast finding of clusters of associated photometric and spectrometric redshifts. In doing so, we find some interesting new ways of finding good quality mappings from photometric to spectrometric redshifts with high confidence.

3.4 The Sloan Digital Sky Survey

The Sloan Digital Sky Survey (SDSS) [155] is systematically mapping the sky producing a detailed image of it and determining the positions and absolute brightnesses of more than 100 million celestial objects. It is also measuring the distance to a million of the nearest galaxies and to a hundred thousand quasars. The acquired data has been openly given to the scientific community.

Figure 3.2 depicts the SDSS data release 5 for imaging and spectral data. For every object a large number of attributes and measurements are acquired, see [2] for a description of the data available in this catalogue.

In particular we use the data made available to us by Prof. Giuseppe Longo's group [107], and used intensively by Raffaele D'Abrusco [36–38].

3.5 Applying Baire ultrametric to astronomy data (SDSS)

The aim here is to build a mapping from $z_{spec} \rightarrow z_{phot}$ to help calibrating the redshifts, based on the z_{spec} observed values. Traditionally we could map $f : z_{phot} \longrightarrow z_{spec}$ based on trained data. The mapping f could be linear (e.g. linear regression) or non-linear (e.g. multilayer perceptron) as used by D'Abrusco [38]. These techniques are global. Here our interest is to develop a locally adaptive

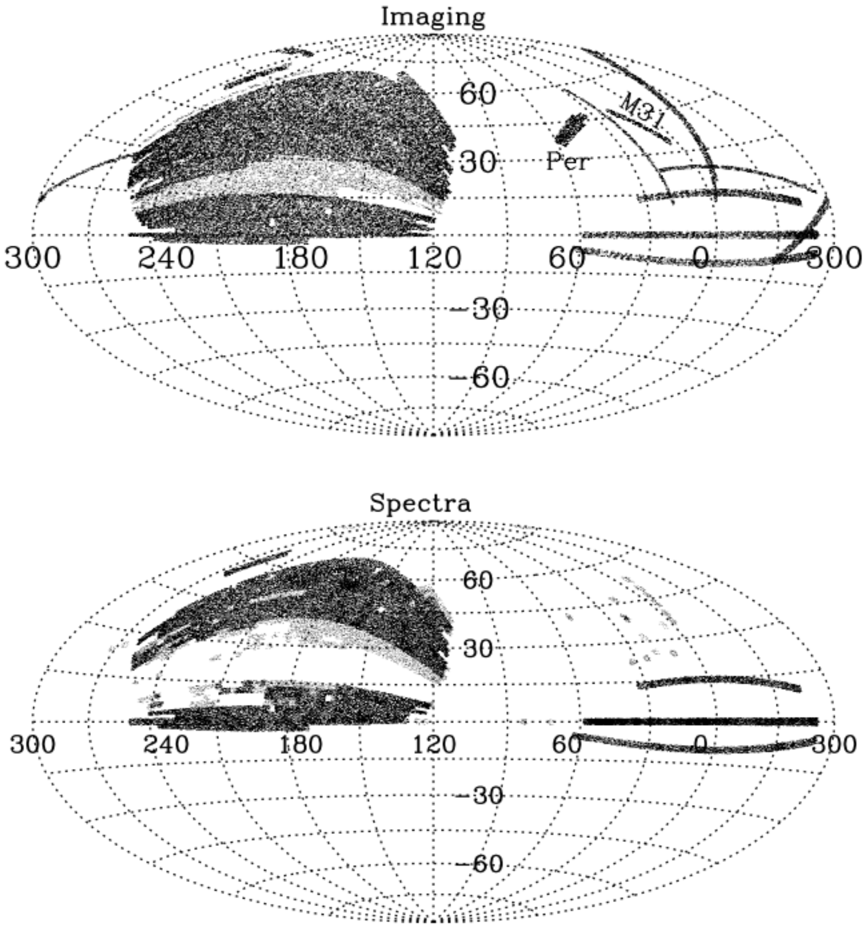


Figure 3.2: *Distribution in the sky of the SDSS data release 5 [2].*

approach based on numerical precision. That is the great benefit of the Baire distance.

We look specifically into four parameters: right ascension (RA), declination (DEC), spectrometric (z_{spec}) and photometric (z_{phot}) redshift. Table 3.1 shows a small subset of the data used for experimentation and analysis. The IDL software was used to read the data which was in Flexible Image Transport System format [60] (FITS), which is traditionally used in astronomy.

As already noted the spectrometric technique uses the spectrum of electromagnetic radiation (including visible light) which radiates from stars and other celestial objects. The photometric technique uses a faster and economical way of measuring the redshifts.

RA	DEC	Spec	Phot
145.4339	0.56416792	0.14611299	0.15175095
145.42139	0.53370196	0.145909	0.17476539
145.6607	0.63385916	0.46691701	0.41157582
145.64568	0.50961215	0.15610801	0.18679948

Table 3.1: *Data format for right ascension, declination, z_{Spec} and z_{Phot}.*

3.5.1 Clustering SDSS data based on a Baire distance

In order to perform the clustering process based on the Baire distance presented in equation 2.22 and described in section 2.8.2, we compare every z_{Spec} and z_{Phot} data point, searching for common prefixes based on algorithm 1 (see section 2.8.2). Thereafter, the data points that have digit coincidences are grouped together to form clusters.

Data characterisation is presented in Figure 3.3. The left panel shows the z_{spec} and z_{phot} sky coordinates of the data currently used by us to cluster redshifts. This section of the sky presents approximately 0.5 million object coordinate points with the current data. As can be observed, various sections of the sky are represented in the data. We find this useful since preliminary data exploration has shown that correlation between z_{spec} and z_{phot} is consistent in different parts of the sky. For example, when taking correspondences between z_{spec} and z_{phot} as shown in Figures 3.5 and 3.6, and plotting them in RA and DEC space (i.e. astronomical coordinate space) we have the same shape as presented in Figure 3.3.

This leads us to conclude that digit coincidences of the redshift measures are distributed approximately uniformly in the sky and are not concentrated spatially. The same occurs for all the other clusters. We will concentrate on the very near astronomical objects, represented by redshifts between 0 and 0.6. When we plot z_{spec} vs. z_{phot} we obtain a highly correlated signal as shown in Figure 3.3, right panel. The number of observations that we therefore analyse is 443,014.

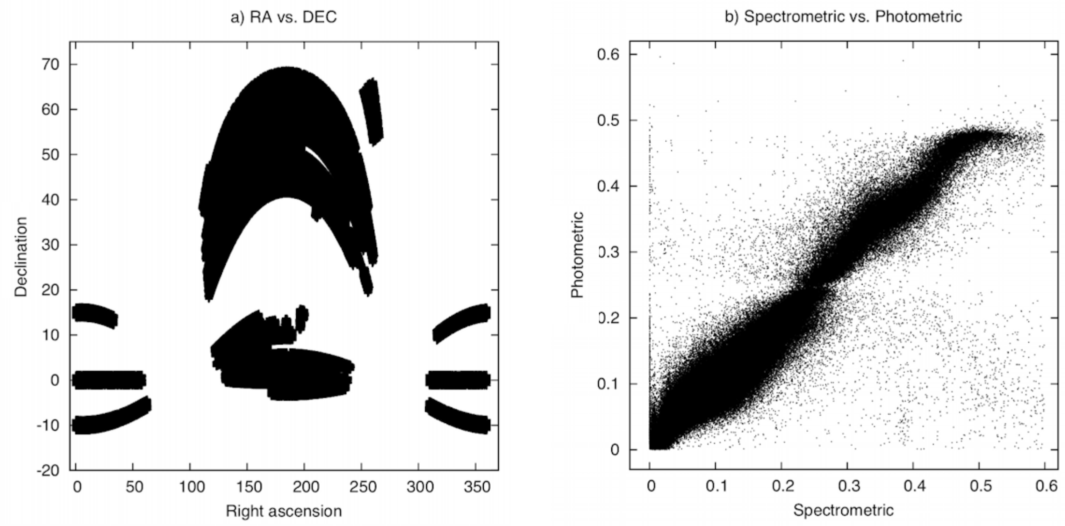


Figure 3.3: *Left: right ascension (RA) vs. declination (DEC); Right: z_{spec} vs. z_{phot}. SDSS data selection used for redshift analysis.*

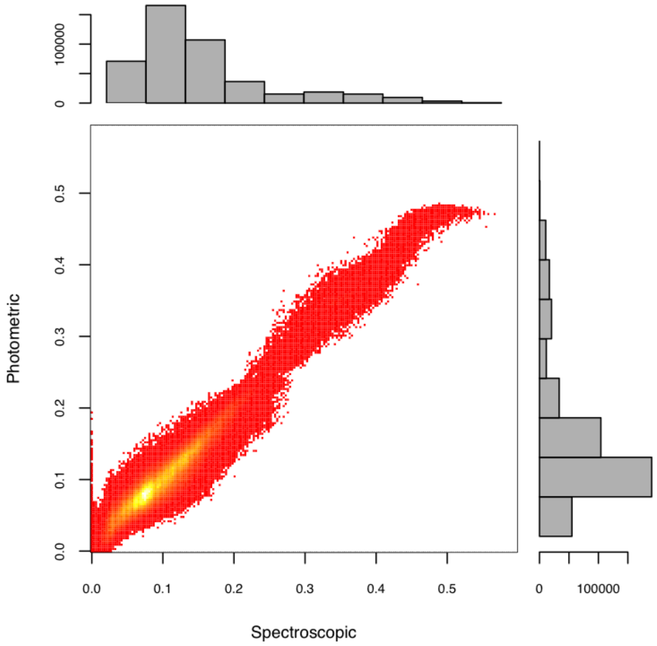


Figure 3.4: *Heat plot and histogram for z_{spec} vs. z_{phot}. Histogram at the top shows the z_{spec} frequencies, histogram at the bottom shows z_{phot} frequencies.*

Looking into Figure 3.4 it can be seen clearly that most data points fall in the range between 0 and 0.2. Here the histogram on the top shows the z_{phot} data points distribution, and the histogram on the right the z_{spec} data points distribution. The heat plot also highlights the area where data points are concentrated, where the yellow colour (white region in monochrome print) shows the major density.

Consequently, now we know that most cluster data values will fall within this range (0 and 0.2) if common prefixes of digits in the redshift values, each value taken as a string, are found.

Figures 3.5 and 3.6 show graphically how z_{spec} and z_{phot} correspondences look at different levels of decimal precision. On one hand we find that values of z_{spec} and z_{phot} that have equal precision up to the 3rd decimal digit are highly correlated. On the other hand when z_{spec} and z_{phot} have only the first digit in common, correlation is weak. For example, let's consider the following situations for plots 3.5 and 3.6:

- Figure 3.5 left: let us take the values of $z_{spec} = 0.437$ and $z_{phot} = 0.437$. We have that they share the first digit, the first decimal digit, the second decimal digit, and the third decimal digit. Thus, we have a highly correlated signal of the data points that share only up to the third decimal digit.
- Figure 3.5 right: let us take the values of $z_{spec} = 0.437$ and $z_{phot} = 0.439$. We have that they share the first digit, the first decimal digit, and the second decimal digit. Therefore, the plot shows data points that share only up to the second decimal digit.
- Figure 3.6 left: let us take the values of $z_{spec} = 0.437$ and $z_{phot} = 0.474$. We have that they share the first digit, and the first decimal digit. Thus, the plot shows data points that share only up to the first decimal digit.
- Figure 3.6 right: let us take the values of $z_{spec} = 0.437$ and $z_{phot} = 0.571$. We have that they share only the integer part of the value, and that alone. Furthermore, this implies redshifts that do not match in succession of decimal digits. For example, if we take the values 0.437 and 0.577, the fact that the third digit is 7 in each case is not of use.

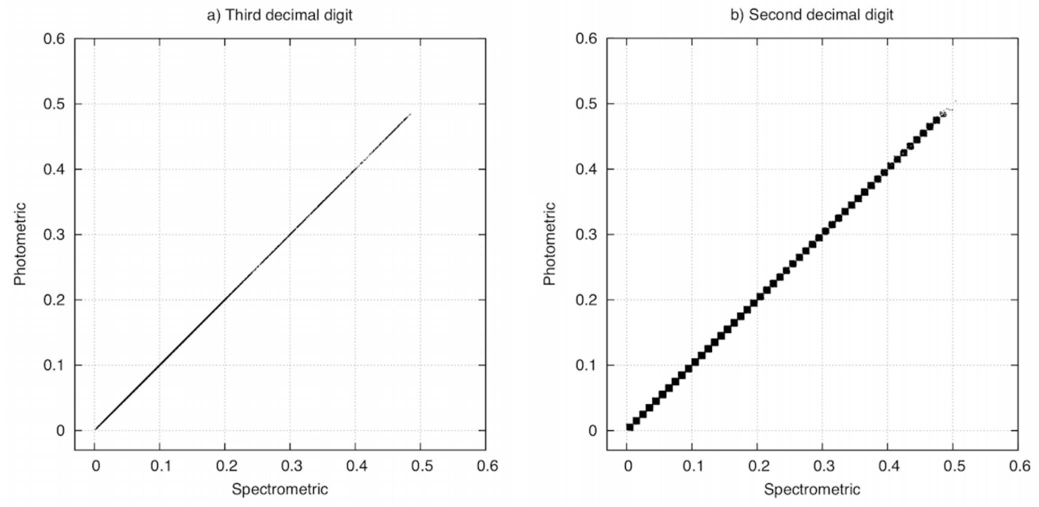


Figure 3.5: *Prefix-wise clustering frequencies depicting only the 3rd decimal digit coincidences (left panel), and only two decimal digit coincidences (right panel).*

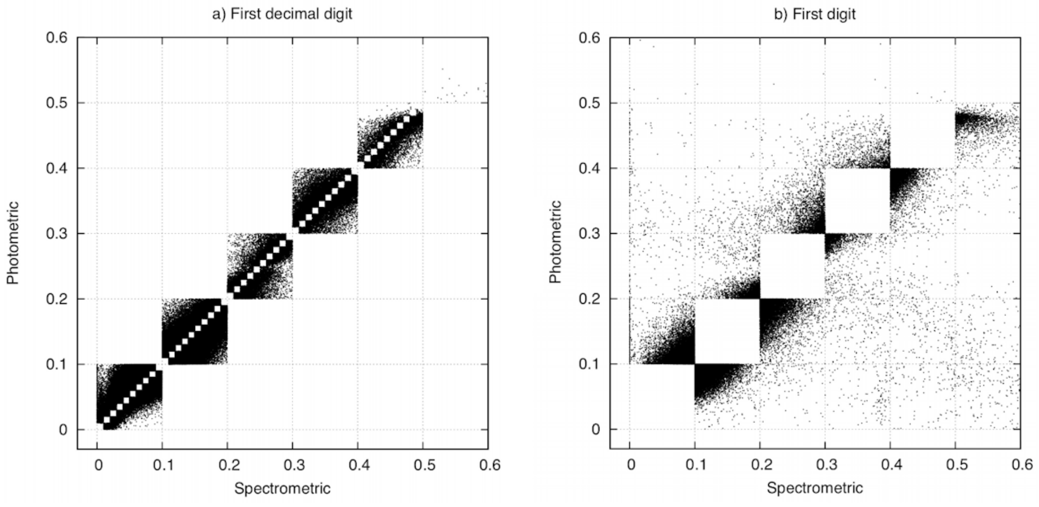


Figure 3.6: *Prefix-wise clustering frequencies depicting only the 1st decimal digit coincidences (left panel), and only first digit coincidences (right panel).*

Table 3.2 shows the clusters found for all different levels of precision. In other words this table allows us to define empirically the confidence levels for mapping of z_{phot} and z_{spec}. For example, we can expect that 82.8% of values for z_{spec} and z_{phot} have at least two common prefix digits. This percentage of confidence is derived as follows: the data points that share six, five, four, three, two, and one decimal digit (i.e., $4 + 90 + 912 + 8,982 + 85,999 + 270,920 = 366,907$ data points. Therefore 82.8% of the data). Additionally we observe that around a fifth of the observations share at least 3 digits in common. Namely, $4 + 90 + 912 + 8,982 + 85,999 = 95,987$ data points, that equals 21.7% of the data.

Digit	No.	%
1	76,187	17.19
Decimal digit	No.	%
1	270,920	61.14
2	85,999	19.40
3	8,982	2.07
4	912	0.20
5	90	0.02
6	4	—
	443,094	100

Table 3.2: *Data points based on the longest common prefix for different levels of precision. This includes the integer part of a data point (first digit) and the decimal digits of a data point (first to sixth digit).*

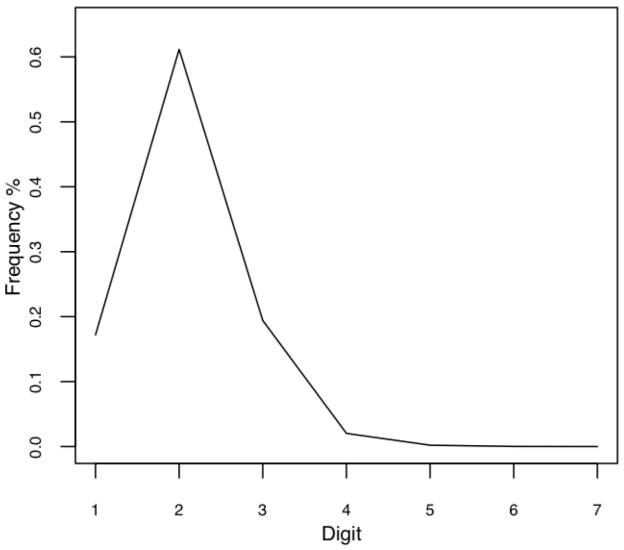


Figure 3.7: *Frequency distribution for Table 3.2. The abscissa shows the digit positions, where 1 is the first digit, 2 the first decimal digit, 3 the second decimal digit and so on.*

In the following section we take this notion of clusters even further and compare it to results obtained with the *k*-means clustering algorithm.

3.5.2 Baire and *k*-means cluster comparison

In order to establish how "good" the Baire clusters are we can compare them with clusters resulting from the *k*-means algorithm. Let us recall that our data values are in the interval $[0, 0.6[$ (i.e. including zero values but excluding 0.6). Additionally, we have seen that the Baire distance is an ultrametric that is strictly defined in a tree. Thus, when building the Baire based clusters we will have a root node "0" that includes all the observations (every single data point analysed starts with 0). For the Baire distance equal to two we have six nodes (or clusters) with indices "00, 01, 02, 03, 04, 05". For the Baire distance of three we have 60 clusters with indices "000, 001, 002, 003, 004,...,059" (i.e. ten children for each node 00,..,05).

We carried out a number of comparisons for the Baire distance of two and three. For example, by design we have that for $d_{\mathcal{B}} = 2$ there are six clusters. Thus we took our data set and applied *k*-means with six centroids based on an implementation from the Hartigan and Wong [83] algorithm. The results can be seen in Table 3.3, where the columns are the k-means clusters and the rows are the Baire clusters. From the Baire perspective we see that the node 00 has 97084 data points contained within the first k-means cluster and 64950 observations in the fifth. Looking at node 04, all members belong to the third cluster of k-means. We can see that the Baire clusters are closely related to the clusters produced by k-means at a given level of resolution.

—	1	5	4	6	2	3
00	97084	64950	0	0	0	0
01	0	28382	101433	14878	0	0
02	0	0	0	18184	4459	0
03	0	0	0	0	25309	1132
04	0	0	0	0	0	11116
05	0	0	0	0	0	21

Table 3.3: *Cluster comparison based on $d_{\mathcal{B}} = 2$. Columns show the k-means clusters, and the rows show the Baire clusters. The cells present the number of data points for a given cluster.*

We can take this procedure further and compare the clusters for $d_{\mathcal{B}}$ defined from 3 digits of precision, and k-means with $k = 60$ centroids as observed in Figure 3.8.

Looking at the results from the Baire perspective we find that 27 clusters are overlapping, 9 clusters are empty, and 24 Baire clusters are completely within the boundaries of the ones produced by k-means as presented in Table 3.5. This last result is better seen in Table 3.4, which is the subset of Table 3.5 where complete matches are shown. These tables have been row and column permuted in order to clearly appreciate the correspondences.

It is seen that the match is consistent even if there are differences due to the different clustering criteria at issue. We have presented results in such a way as to show both consistency and difference.

3.5.3 Baire and k-means clustering time comparison

In order to compare the time performances of the Baire and k-means algorithms we took $d_{\mathcal{B}} = 3$ as a basis for the test. Let us remember that for $d_{\mathcal{B}} = 3$ we have potentially 60 clusters for the data in the range $[0, 0.6[$. Looking at the classification from the hierarchical tree viewpoint we have: one cluster for first level (i.e., the root node or first digit); six clusters for the second level (i.e., first decimal digit or 0, 1, 2, 3, 4, and 5); and ten clusters for the third level or second decimal digit. To obtain the potential number of clusters we multiply the potential nodes for the first, second and third levels of the tree. That is $1 \cdot 6 \cdot 10 = 60$ clusters.

Therefore for the time comparison we have $d_{\mathcal{B}} = 3$ of 60 clusters, which is the parameter given to k-means as initial number of centroids. The other parameter needed is the number of iterations. For k-means we are interested in the average time over many runs. Thus, we use average time over 50 executions for each iteration of 1, 5, 10, 15, 20, 28, 30, 35, and 38.

—	21	1	6	38	25	58	32	20	15	13	14	37	17	2	51	4
015	3733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004	0	3495	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018	0	0	2161	0	0	0	0	0	0	0	0	0	0	0	0	0
020	0	0	0	1370	0	0	0	0	0	0	0	0	0	0	0	0
001	0	0	0	0	968	0	0	0	0	0	0	0	0	0	0	0
000	0	0	0	0	515	0	0	0	0	0	0	0	0	0	0	0
022	0	0	0	0	0	896	0	0	0	0	0	0	0	0	0	0
034	0	0	0	0	0	0	764	0	0	0	0	0	0	0	0	0
036	0	0	0	0	0	0	0	652	0	0	0	0	0	0	0	0
037	0	0	0	0	0	0	0	508	0	0	0	0	0	0	0	0
026	0	0	0	0	0	0	0	0	555	0	0	0	0	0	0	0
027	0	0	0	0	0	0	0	0	464	0	0	0	0	0	0	0
032	0	0	0	0	0	0	0	0	0	484	0	0	0	0	0	0
030	0	0	0	0	0	0	0	0	0	0	430	0	0	0	0	0
045	0	0	0	0	0	0	0	0	0	0	0	398	0	0	0	0
044	0	0	0	0	0	0	0	0	0	0	0	295	0	0	0	0
039	0	0	0	0	0	0	0	0	0	0	0	0	278	0	0	0
024	0	0	0	0	0	0	0	0	0	0	0	0	0	260	0	0
041	0	0	0	0	0	0	0	0	0	0	0	0	0	0	231	0
042	0	0	0	0	0	0	0	0	0	0	0	0	0	0	225	0
047	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	350
048	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	57
049	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 3.4: *Subset of cluster comparison based on $d_{\mathcal{B}} = 3$; columns show the k-means clusters ($k = 60$); rows show Baire nodes.*

—	21	1	6	38	25	58	32	20	15	13	14	37	4	17	2	51	30	16	28	44	59	46	23	48	33	60	40	35	50	42	26	31	27	56
015	3733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004	0	3495	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018	0	0	2161	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020	0	0	0	1370	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
001	0	0	0	0	968	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
000	0	0	0	0	515	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
022	0	0	0	0	0	896	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
034	0	0	0	0	0	0	764	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
036	0	0	0	0	0	0	0	652	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
037	0	0	0	0	0	0	0	508	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
026	0	0	0	0	0	0	0	0	555	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
027	0	0	0	0	0	0	0	0	464	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
032	0	0	0	0	0	0	0	0	0	484	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030	0	0	0	0	0	0	0	0	0	0	430	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
045	0	0	0	0	0	0	0	0	0	0	0	398	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044	0	0	0	0	0	0	0	0	0	0	0	295	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
047	0	0	0	0	0	0	0	0	0	0	0	0	350	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048	0	0	0	0	0	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
049	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
050	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
039	0	0	0	0	0	0	0	0	0	0	0	0	0	278	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024	0	0	0	0	0	0	0	0	0	0	0	0	0	0	260	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
041	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	231	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
042	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	225	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
002	0	0	0	247	0	0	0	0	0	0	0	0	0	0	0	0	1870	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
003	0	523	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2320	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
005	0	118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1720	2392	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
006	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	389	6024	0	0	0	0	0	0	0	0	0	0	0	0	0	0
007	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1417	825	5989	0	0	0	0	0	0	0	0	0	0	0	0
008	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	559	7001	0	0	0	0	0	0	0	0	0	0	0
009	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7904	59	5042	0	0	0	0	0	0	0	0	0
010	0	0	0	0	0	0	1613	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	710	3148	0	0	0	0	0	0	0
011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	213	5437	0	0	0	0	0	0
012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1784	239	3244	0	0	0	0
013	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4392	861	0	0	0
014	517	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3099	927	0	0
016	697	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	259	2153
017	0	0	116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2302
019	0	0	644	1187	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
021	0	0	0	245	0	787	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
023	0	0	0	0	0	294	287	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
035	0	0	0	0	0	0	603	239	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
033	0	0	0	0	0	0	129	0	0	463	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038	0	0	0	0	0	0	0	72	0	0	0	0	0	317	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
025	0	0	0	0	0	0	0	0	20	0	0	0	0	0	334	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028	0	0	0	0	0	0	0	0	204	0	275	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
031	0	0	0	0	0	0	0	0	0	462	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
029	0	0	0	0	0	0	0	0	0	0	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
043	0	0	0	0	0	0	0	0	0	0	0	161	0	0	0	76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
046	0	0	0	0	0	0	0	0	0	0	0	150	213	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040	0	0	0	0	0	0	0	0	0	0	0	0	0	88	0	127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.5: *Cluster comparison based on $d_{\mathcal{B}} = 3$. Column: k-means clusters; Rows: Baire clusters. The array has been row and column permuted in order to highlight the good correspondence.*

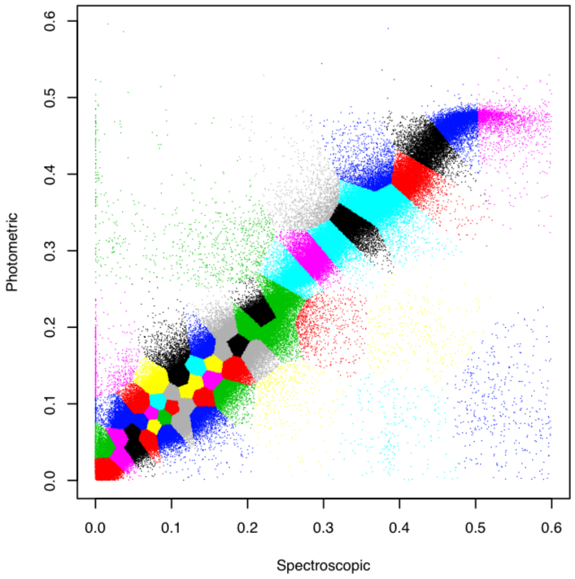


Figure 3.8: *K-means clustering for k* $= 60$ *after 38 iterations. Note that non-contiguous groups may be coloured the same.*

The results can be observed in Figure 3.9. It is clear that the time in *k*-means is linear with respect to the number of iterations (this is well understood in the *k*-means literature). In this particular case the algorithm converges around the iteration number 38. Note that these executions are based on different random initialisations. The times for the *k*-means algorithm were obtained with the R statistical software. These times were faster than the times obtained by the algorithm implemented with Java.

The Baire method only needs one pass over the data to produce the clusters. Regarding the time needed, we tested a Java implementation of the Baire algorithm. We ran 50 experiments over the SDSS data. It took on average 2.9 seconds. Compare this to Table 3.6.

We recall that this happens because of the large number of iterations involved in the case of *k*-means. Even in the case when just one iteration is considered for *k*-means (note that the algorithm does not converge in that case) the time taken is more than double when compared with the Baire (6.8 seconds vs. 2.9 seconds).

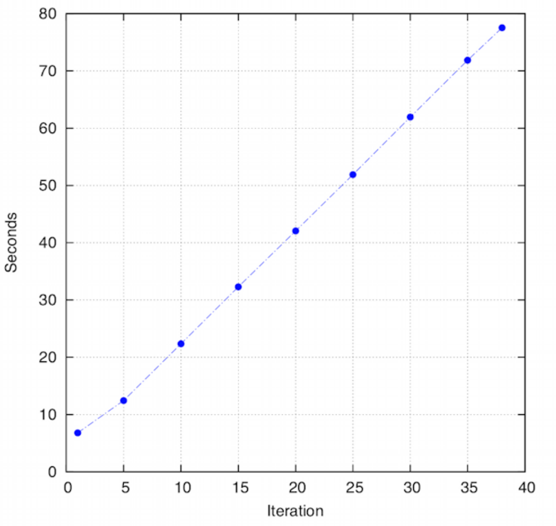


Figure 3.9: *K-means average processing time in seconds for $k = 60$. Averages are obtained for 9 examples with 50 executions each.*

3.6 Spectrometric and photometric digit distribution

We have seen that the Baire ultrametric produces a strict hierarchical classification. In the case of z_{spec} and z_{phot} this can be seen as follows. Let us take any observed measurement of either case of $z_{spec} = z_{phot}$. Let us say $z_{spec} = z_{phot} = 0.1257$. Here we have that for $K = 4$, $z_{spec} = z_{phot}$. Hierarchically speaking we have that the root node is 0, for the first level where there potentially exist 6 nodes (i.e. 0,1,...,5);

Iteration	Average time
1	6.81
5	12.44
10	22.35
15	32.30
20	42.07
25	51.90
30	61.94
35	71.85
38	77.53

Table 3.6: *Time average for k-means algorithm over 50 executions for each total iteration count.*

for the second level potentially there are 60 nodes; and so on until $K = 4$, and $z_{spec} = z_{phot}$, where potentially there are $6 \cdot 10 \cdot 10 \cdot 10 = 6,000$ nodes.

Of course not all nodes will be populated, in fact we can expect that a large number of these potential nodes will be empty if the number of observations n is lower than the potential number of nodes for a certain precision K (i.e. $n \leq K^{10}$). Note that this points to a big storage cost, but in practice the tree is very sparsely populated and K small, see section 7.3.4.

A particular interpretation can be given in the case of an observed data point. Following up the above example if we take $z_{spec} = z_{phot} = 0.1257$, a tree can be produced to store all observed data that falls within this node. Doing this has many advantages from the viewpoint of storing. Access and retrieval, for example, is very fast and it is easy to retrieve all the observations that fall within a given node and its children.

With this tree it is a trivial task to build bins for data distribution. Figure 3.10 depicts the frequency distribution for a given digit and precision. There are 100 data points that have been convolved with a Gaussian kernel to produce surface planes in order to assemble three-dimensional plots.

This helps to build a cluster-wise mapping of the data. Following the Figure 3.10 top panel we observe that for the first decimal digit most data observations are concentrated in the digits 0, 1, 2, and 3. Then the rest of decimal precision data is uniformly distributed, gradually going towards zero when the level of precision increases. There is the exception of two peaks, for precision equal to 8. This turns out to be useful because when comparing the z_{spec} and z_{phot} digit distribution we don't find the same peaks in z_{phot}. This is very useful because now we can discriminate which observations are more reliable in z_{phot} through different characteristics of the data associated with the peaks.

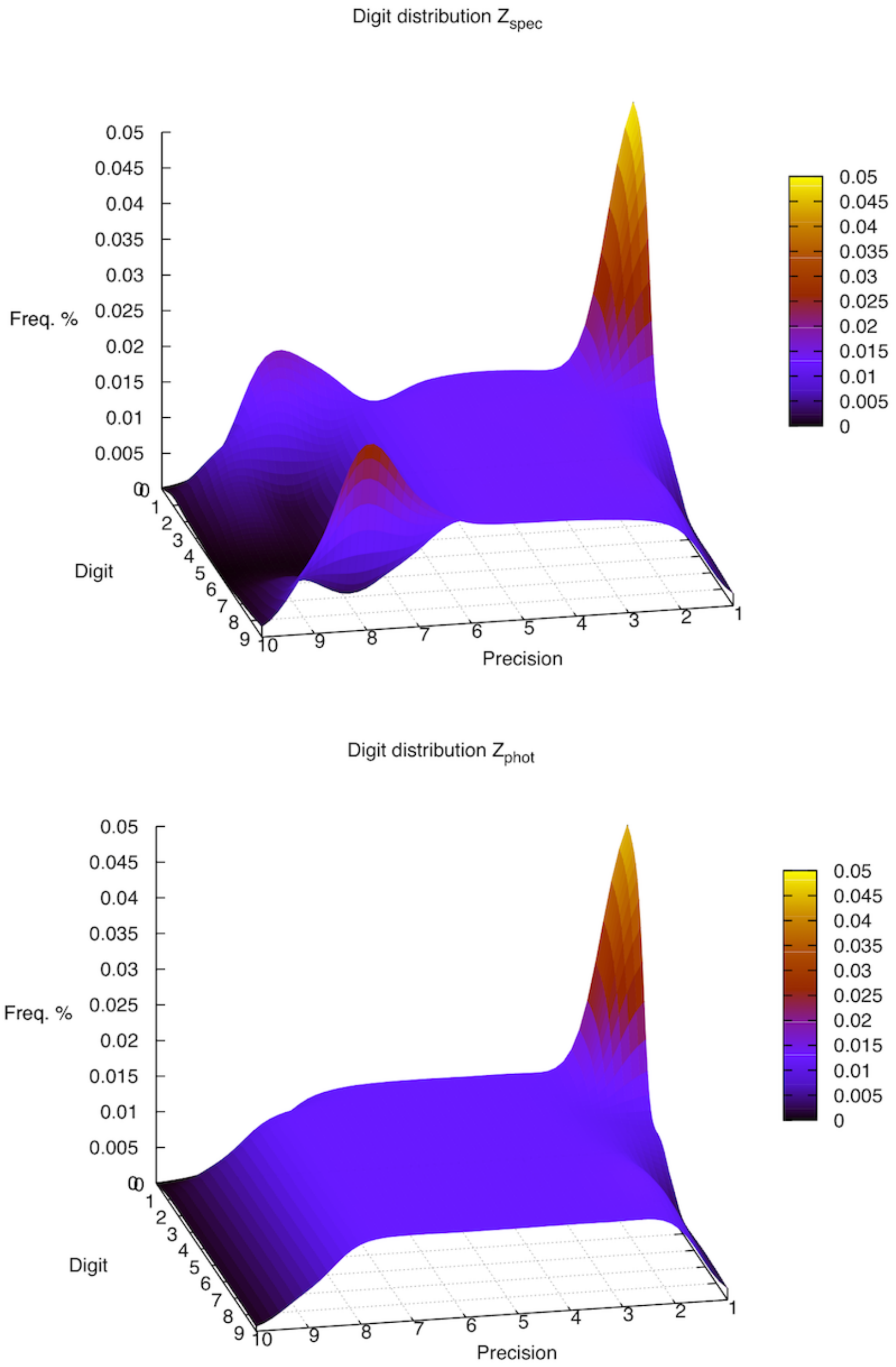


Figure 3.10: *Digit distribution for z_{spec} and z_{phot}; Top: Spectrometric digit distribution; Bottom: Photometric digit distribution. Note that digit distribution for z_{spec} has three peaks, but z_{phot} only one.*

3.7 Summary

In this work a novel distance called the Baire distance is presented. We show how this distance can be used to generate clusters in a way that is computationally inexpensive when compared with more traditional techniques. This approach therefore is a good candidate for exploratory data analysis when data sets are very big. In addition to the advantage of speed, this distance is an ultrametric which can easily be seen as a hierarchy.

In the astronomy case clusters generated with the Baire distance can be useful when calibrating redshifts. In general, applying the Baire method to cases where digit precision is important can be of relevance, specifically to highlight data "bins" and some of their properties.

Note that when two numbers share 3 prefix digits we have a $d_{\mathcal{B}} = 3$ which is a Baire distance of $2^{-3} = 0.125$. We did not need to define the actual (ultra)metric values in this chapter. It was, in fact, more convenient to work on the hierarchy, with its different levels.

In section 3.5.1 we showed how we could derive that 82.8% of values for z_{spec} and z_{phot} have at least two common prefix digits. This is a powerful result in practice when we recall that we can find very efficiently where these 82.8% of the astronomical objects are.

Using the Baire distance we showed in section 3.6 that z_{spec} and z_{phot} signals can be stored in a tree like structure. This is advantageous when measuring the digit distribution for each signal. When comparing these distributions, it can easily be seen where the differences arise.

Chapter 4

Application to Chemistry: Clustering Chemical Compounds

4.1 Introduction

CHEMISTRY in many ways suffers from the same problems described in section 2.2. In this area large databases containing millions of chemical compounds are available. Furthermore, new clustering and data analysis techniques are needed in order to identify meaningful relationships within this data.

In this chapter we describe some of the challenges presented when analysing large databases in the chemistry area. A large number of chemical structure-key fingerprints annotated in binary form are used for analysis.

First we describe and characterise the dataset available to us. Then the chemistry data space is embedded within a Baire ultrametric. To achieve this, the data is normalised and then randomly projected to a unidimensional vector. Then the projections are sorted, allowing the Baire distance to be calculated, and applied to form clusters. Resulting clusters are compared to clusters obtained from using the *k*-means algorithm which in turn uses the Baire clusters as centroids. The last section is dedicated to a discussion on random projection and the computational complexity of the method proposed here.

4.2 Problem description and data characterisation

One of the most common problems in mining large chemical libraries is classifying the compounds into different classes. Different classes could represent different levels of activity within compounds.

The application of clustering techniques to chemical databases started in the early 1980s [49]. This coincided with the increase of chemical compound collections in huge databases and advances made by the information retrieval community to analyse large datasets.

In the 1990s, the Ward minimum variance hierarchical clustering method became the method of choice due to its hierarchical nature and the quality of the clusters produced. Unfortunately the method reached its limits once the pharmaceutical companies tried processing datasets of more than 500,000 compounds due to: the $O(n^2)$ processing requirements of the reciprocal nearest neighbour algorithm; the requirement to hold all chemical structures in memory to enable random access; and the requirement that parallel implementation use a shared-memory architecture.

Currently technology allows for coding chemical structures into different schemes or representations that are easier to analyse with computer technology. However, due to the molecules' complicated structure, their computer representation is not unique. For example, in chemoinformatics one of the most used molecular representation is the topological (2D chemical structure), but also other methods such as: adjacency matrix, connection graph, line notation, and molecular geometries are available. See [23] for a description of these.

Substructure searching and matching in chemoinformatics is an NP-complete problem [74]. Thus, querying large molecular databases is very time consuming. For this reason and in order to filter out those molecules that do not contain the substructures of interest the *substructure screening* method was developed. This is based on a vector representation where mainly two methods are used: hash-key fingerprints and structure-key fingerprints. The former is of interest to us here (for a description of hash-key fingerprints see [23]).

Structure-key (also called dictionary-based) fingerprints use a dictionary of defined substructures to produce a fixed length binary string. Basically, when encoding a molecule if the substructural key is present it will be annotated as 1 in the binary string or 0 in the case of absence. To carry out this encoding a number of dictionaries are available such as the Elsevier MDL [52]. Figure 4.1 shows a structure-key like binary representation of a typical chemical structure.

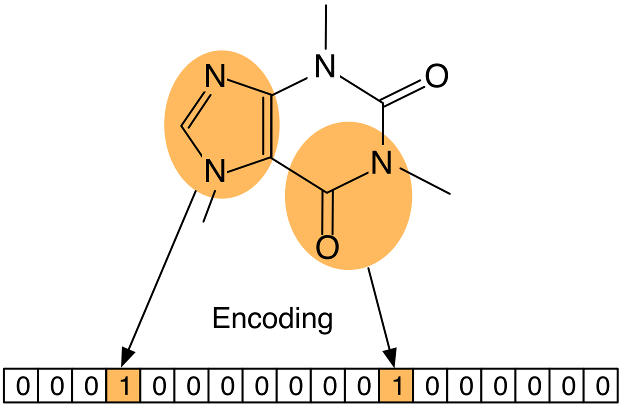


Figure 4.1: *An example of a molecule encoding as a structure-key fingerprint using a fragment dictionary. A defined fragment is assigned to a single bit position on the string [23].*

4.2.1 Notation used and data normalisation

We will use the notation x for the data matrix to be analysed, and x_i denotes any particular row. A chemical structure (or chemical) i is represented by a row, and the set of chemical structures, or rows, is denoted I. We work with just over 1.2 million chemicals, $i \in I$. Similarly the column codes or attributes, 1052 in number, are denoted by set J. Needless to say, our chemicals × codes view of the data, used here for convenience of exposition, is fully compatible with a more appropriate form of storage.

We will take the notation a little further (as in [126]) by writing x_{IJ} for the given data, and a row of this matrix is denoted x_{iJ} (so we are indicating row i and the column set, J). The sum of the columns gives the vector (marginal)

x_J. We normalise the data by dividing each matrix value by its column sum, and the resulting normalised matrix is denoted x_{IJ}^J. Here we are saying: the presence of a code j in chemical i must take into account whether that code is rare, implying importance of the presence property; or common, implying a lower value of presence. Given our notation, a tensor product allows us to reconstruct our original data: $x_{IJ}^J \circ x_J = x_{IJ}$ ($\circ$ denotes tensor product; see [126] for use of this notation). Normalisation can be very important, to homogenise the effects of the coding identifiers (set J) that are used: see Figure 4.2.

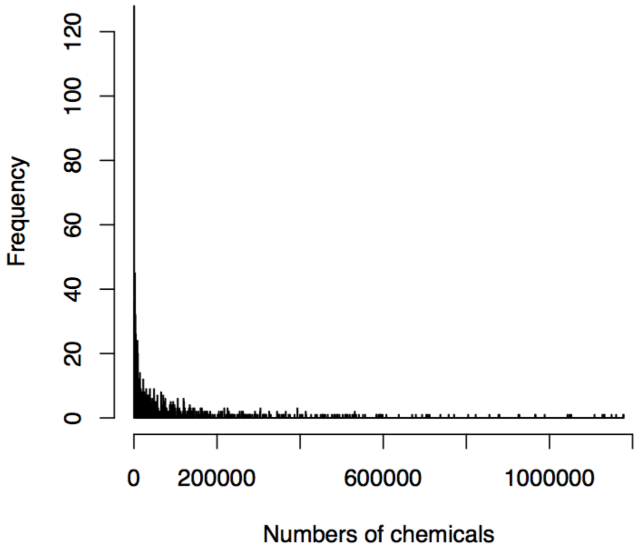


Figure 4.2: *Histogram of column sums, denoted x_J in the text.*

4.2.2 Data distribution and properties

We use a set of 1,219,553 chemical structures coded through 1052 presence/absence values, using the Digital Chemistry bci1052 dictionary of fragments [178]. Our experimental work is based on a matrix of binary-valued vectors: in some instances it would be more efficient to work directly on the small set of code off-sets rather than a 1052-vector. The binary-valued matrix is sparse: occupancy is 8.6347%.

A power law (see [117]) is a distribution (e.g. of frequency of occurrence) in the general form $x^{-\alpha}$ where constant $\alpha > 0$; and an exponential law is of the form e^{-x}. For a power law, $P(x > x_0) \sim cx^{-\alpha}$, $c, \alpha > 0$. A power law has heavier tails than an exponential distribution. In practice $0 \leq \alpha \leq 3$. For such values, x has infinite (i.e. arbitrarily large) variance; and if $\alpha \leq 1$ then the mean of x is infinite. The density function of a power law is $f(x) = \alpha c x^{-\alpha-1}$, and so $\ln f(x) = -\alpha \ln x + C$, where C is a constant offset. Hence a log-log plot shows a power law as linear. Power laws have been of great importance for modelling networks and other complex data sets.

Figure 4.3 shows a log-log plot based on the 1052 presence/absence attributes, using all 1.2 million chemicals. In a very similar way to the power law properties of large networks (or file sizes, etc.) we find an approximately linear regime, ending (at the lower right) in a large fan-out region.

The slope of the linear region characterises the power law. For this data, we find that the probability of having more than n chemicals per attribute to be approximately $c/n^{1.23}$ for large n.

The histogram of attributes per chemical, on the other hand, does not show a pronounced power law: see Figure 4.4. In fact, it it close to a Gaussian.

4.3 Ultrametric from longest common prefixes

4.3.1 From boolean data to normalised, real-valued data

In this section we use (i) a random projection of vectors into a 1-dimensional space (so each chemical structure is mapped onto a scalar value, by design ≥ 0 and ≤ 1) followed by (ii) implicit use of a prefix tree constructed on the digits of the set of scalar values. First we will look at this procedure. Then we will return to discuss its properties.

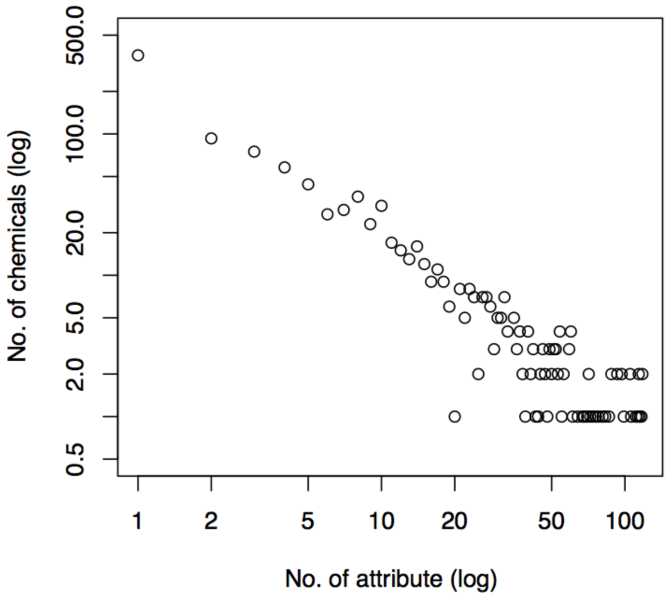


Figure 4.3: *Log-log plot of number of chemicals per attribute, based on the whole data set of 1.2 million chemicals.*

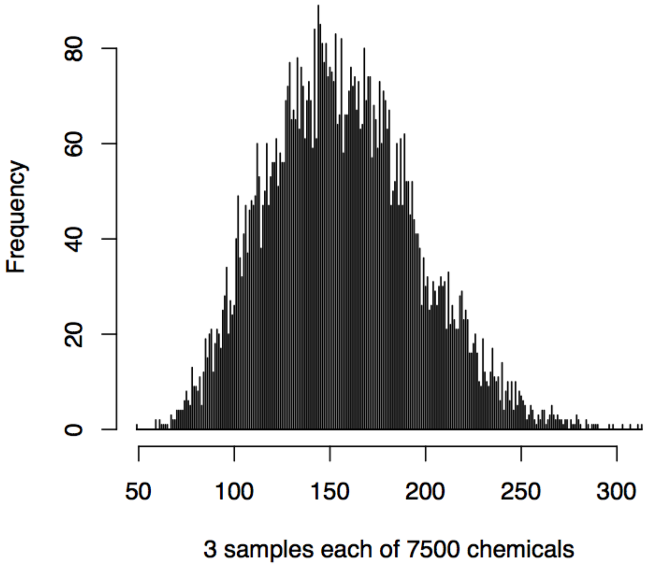


Figure 4.4: *Histogram of numbers of attribute presences for the set of chemicals.*

4.3.2 Ultrametrisation through Baire space embedding

To achieve a simple clustering hierarchy, we applied the following steps:

- Matrix normalisation by column sums.
- For precision k, k = 1, 2, ..., $|K|$. For attribute set, J. Determine random projections of all chemical vectors into a 1-dimensional space.
- Sort projected values; determine identical values, to define cluster, grouping chemicals that are projected into the same value.

Our objective is not the same as fitting a hierarchical structure, as is traditionally used in multivariate data analysis. A mainstream approach over at least four decades of data analysis has been to fit a tree structure well to a data set, with quality of fit presupposing a clustering (mostly agglomerative, but possibly divisive). Instead we seek inherent ultrametricity in a data set, and so the more ultrametricity we find in our data the better.

Table 4.1 shows results for three different data sets, each consisting of 7500 chemicals. The first column presents the significant digit and the second column presents the number of clusters obtained, where the number of significant decimal digits is 4 (more precise, and hence more different clusters found), 3, 2, and 1 (lowest precision in terms of significant digits).

Therefore for 7500 chemicals we found: approximately 140 clusters at precision (number of digits) 1; approximately 2,550 clusters at precision 2; and approximately 6,400 clusters at precision 3.

We seek all i, i' such that

1. for all $j \in J$,
2. $x_{ijK} = x_{i'jK}$
3. to fixed precision K

Recall that K is an ordered set. We impose a user specified upper limit on precision, $|K|$.

Now rather than $|J|$ separate tests for equality of projected chemicals, a *sufficient condition* is that $\sum_j w_j x_{ijK} = \sum_j w_j x_{i'jK}$ for a set of weights w_j. What helps in making this sufficient condition for equality work well in practice is that many of the x_{iJK} values are 0: cf. the approximate 8% matrix occupancy rate that holds here. We experimented with such possibilities as $w_j = j$ (i.e., $\{1, 2, \ldots, |J|\}$ and $w_j = |J| + 1 - j$ (i.e., $\{|J|, |J| - 1, \ldots, 3, 2, 1\}$. A first principal component would allow for the definition of the least squares optimal linear fit of the projections. The best choice of w_j values we found for uniformly distributed values in $(0, 1)$: for each j, $w_j \sim U(0, 1)$.

Table 4.1 shows, in immediate succession, results for the same set of three data sets used previously. The normalising column sums were calculated and applied independently to each of the three data sets. Insofar as x_J is directly proportional, whether calculated on 7500 chemical structures or 1.2 million, leads to a constant of proportionality, only, between the two cases.

Sig. dig. c	No. clusters
4	6591
4	6507
4	5735
3	6481
3	6402
3	5360
2	2519
2	2576
2	2135
1	138
1	148
1	167

Table 4.1: *Results for the three different data sets, each consisting of 7500 chemicals, are shown in immediate succession. The number of significant decimal digits is 4 (more precise, and hence more different clusters found), 3, 2, and 1 (lowest precision in terms of significant digits).*

A random projection was used. Finally, identical projected values were read off, to determine clusters.

Let us look closer at one outcome here, the 4-digit precision set of 6591 clusters found for the first of the three data sets used. We may ask whether these clusters are "balanced" or if, in fact, one massive cluster accounts for most of the chemical structures. Figure 4.5 shows a histogram, indicating clearly the "balance" in cluster cardinalities.

For a smaller precision, however, such as 1-digit, we find that one very large cluster dominates in terms of cardinality (cf. discussion of k-means results in section 4.3.3 below).

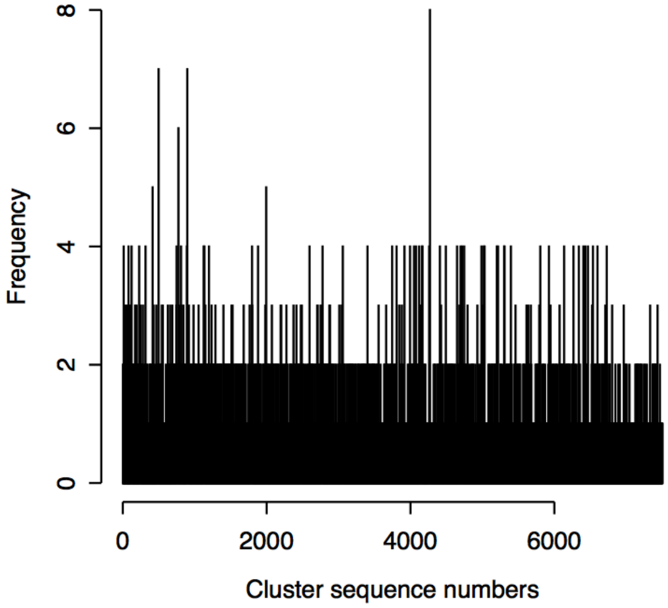


Figure 4.5: *Histogram of cluster sizes.*

4.3.3 Comparison with k-means clustering algorithm

In Table 4.2 we show the results of k-means for the same three data sets each relating to 7500 chemical structures, with 1052 descriptors."Sig. dig.": number of significant digits used. "No. clusters": number of clusters in the data set of 7500 chemical structures, associated with the number of significant digits used in the Baire scheme. "Largest cluster": cardinality. "No. discrep.": number of discrepancies found in k-means clustering outcome.

We look at k-means, using as input the cluster centres provided by the 1-significant digit Baire approach. Relatively very few changes were found. We note that the partitions in each case are dominated by a very large cluster, which is a direct consequence of the data used. In cases that do not give rise to such "imbalanced" cluster cardinalities, our Baire-related approach should perform even better, in that it will give rise to more equal cardinality clusters.

As a conclusion we can say that these results show that Baire and k-means algorithms can present a similar outcome when used for chemical compound clustering.

Sig. dig.	No. clusters	Largest cluster	No. discrep.
1	138	7037	3
1	148	7034	1
1	167	6923	9

Table 4.2: *Results of k-means for three different datasets samples related to 7500 chemical structures with 1052 descriptors."Sig. dig": number of significant digits used. "No. clusters": number of the clusters in the dataset of 7500 chemicals structures. "Largest cluster"; cardinality. "No. discrep.": number of discrepancies found in k-means cluster outcome.*

4.4 Discussion on random projection and hashing

Random projection is the finding of a low dimensional embedding of a point set – dimension equals 1, or a line or axis, in this work – such that the distortion of any pair of points is bounded by a function of the lower dimensionality [169]. While random projection *per se* will not guarantee a bijection of best match in original and in lower dimensional spaces, our use of projection here is effectively a hashing method ([113] uses MD5 for nearest neighbour search), in order to deliberately find hash collisions – thereby providing a sufficient condition for the mapped vectors to be identical.

Collision of identically valued vectors is guaranteed, but what of collision of non-identically valued vectors, which we want to avoid?

To prove such a result may require an assumption of what distribution our original data follow. A general class is referred to as a stable distribution [6]: this is a distribution such that a limited number of weighted sums of the variables is also itself of the same distribution. Examples include both Gaussian and long-tailed or power law distributions.

Interestingly, however, very high dimensional (or equivalently, very low sample size or low n) data sets, by virtue of high relative dimensionality alone, have points mostly lying at the vertices of a regular simplex or polygon [123]. This intriguing aspect is one reason, perhaps, why we have found random projection to work well. Another reason is the following: if we work on normalised data, then the values on any two attributes j will be small. Hence x_j and x'_j are small. Now if the random weight for this attribute is w_j, then the random projections are, respectively, $\sum_j w_j x_j$ and $\sum_j w_j x'_j$. But these terms are dominated by the random weights. We can expect near equal x_j and x'_j terms, for all j, to be mapped onto fairly close resultant scalar values.

How close are these values that are mapped? In the following section, we look in more detail at how good this mapping is.

4.4.1 Random projection and digit distribution

A very important question is to know if distances are kept in the lower dimensional space after applying random projection. If distances are maintained the clusters obtained using a lower dimensional signal will be similar to the ones obtained using the whole dataset. For the dataset that we are using, we can test empirically if this holds.

Let us recall what we have discussed in section 4.4. Following equation 2.3 and the notation introduced in section 4.2.1 we have the following:

$$x^{RP}_{I \times k} = x^{J}_{I \times J} \times R^{k}_{J' \times k}$$

where x is the original dataset with I observations and J-dimensions, if normalised it becomes $x^{J}_{I \times J}$; $R_{J' \times k}$ is a random matrix where $J' = J$ and k rows, if normalised by column sums it becomes $R^{k}_{J' \times k}$. Here we normalise R in order to obtain values between 0 and 1 for x^{RP}; and x^{RP} is the resulting sub-space projection.

To test if the distances hold in the projected sub-space we have selected a sample from the original data matrix with 20,000 observations and keeping the 1052 dimensions corresponding to the binary fingerprints of chemical descriptors.

Thus, we want to project the data matrix x onto a unidimensional vector. Therefore, we obtain a random generated matrix R with one dimension and with k = 20,000.

Algebraically we have the following:

$$x^{RP}_{20{,}000 \times 1} = x_{20{,}000 \times 1052} \; R_{1052 \times 1} \tag{4.1}$$

Note that the binary (presence/absence) data matrix x has been normalised by column sums. The same process is applied to the randomly generated matrix R. We normalise values in order to obtain a random projection matrix x^{RP} with values between 0 and 1.

If the random vectors are orthogonal, the random projected matrix x^{RP} will preserve the original distances exactly [94]. This is referred to as Parseval invariance in the signal processing literature. Therefore, orthogonality in the random matrix is the ideal case. However, orthogonalisation is very costly computationally speaking. Hecht-Nielsen [85] noted that in a high dimensional space there are many more quasi-orthogonal vectors than orthogonal ones. This means that we can approximate orthogonality by simply choosing random directions in the high-dimensional space [151].

For this chemical dataset we can test empirically if the behaviour observed by Hecht-Nielsen holds. Thus, we carry out several random projections and calculated the digit distribution for each projection in order to verify the digit distributions. The idea behind this is the following: by counting the digit occurrences for a given precision we are measuring the resulting projected distribution. If different projections produce a similar mapping, we can conclude that the distances are kept in the lower dimensionality.

Figure 4.6 depicts the digit distribution for 8 different random projected vectors (we carry out 50 projections but for reasons of space we only include 8 here). It can be observed that the digit distributions are almost identical and practically indistinguishable from each other. Therefore, it can be concluded that for this dataset the random projection method for dimensionality reduction preserves the distances from the original data space.

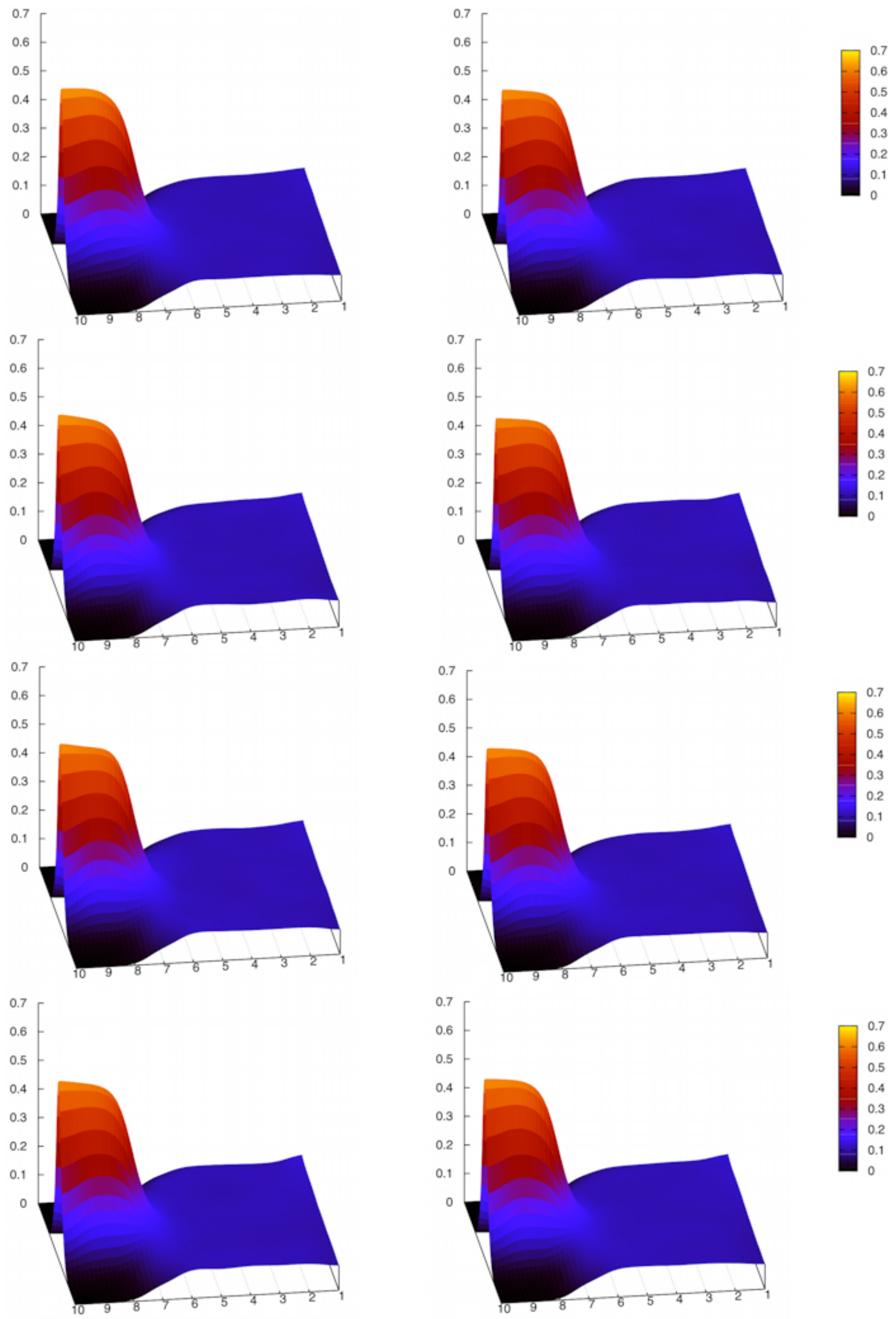


Figure 4.6: *Digit distributions from eight different random projections, where: the x axis shows the digit decimal position; the y axis shows the numeric digits from 0 to 9; and the z axis shows the normalised frequency of digit occurrences in percentage.*

4.4.2 Computational time complexity

The computational time complexity is as follows. As usual, let the number of chemicals be denoted $n = |I|$; the number of attributes is $|J|$; and the total number of digits precision is $|K|$. Consider a particular number of digits precision, k_0, where $1 \leq k_0 \leq |K|$. Then the random projection takes $n \cdot k_0 \cdot |J|$ operations. A sort follows, requiring $O(n \log n)$ operations. Then clusters are read off with $O(n)$ operations. Overall, the computational effort is bounded by $c_1 \cdot |I| \cdot |J| \cdot |K| + c_2 \cdot |I| \cdot \log |I| + c_3 |I|$ (where c_1, c_2, c_3 are constants), which is equal to $O(|I| \log |I|)$ or $O(n \log n)$.

4.5 Summary

In this chapter we have shown that a Baire ultrametric embedding can be effectively used to cluster chemical fingerprints. In this particular case the data space is very sparsely populated, a characteristic that helps when reducing the dimensionality through random projection.

Additionally we show that reducing dimensionality by means of random projection does not significantly change the results, even when different random vectors are used. This is a direct consequence of the phenomenon noted by Hecht-Nielsen, which states that in high dimensional spaces there many more quasi-orthogonal vectors than orthogonal ones. The quasi-orthogonal vectors may be close enough to orthogonal, thus close to the optimal solution for reducing dimensionality.

It follows that the resulting clusters are comparable to the ones obtained using the k-means clustering algorithm.

The advantage in using the method proposed here is based on the number of calculations needed and the lower computational complexity when compared with other clustering algorithms.

Chapter 5

Application to Biology: Protein Clustering

5.1 Introduction

GIARDIA intestinalis causes giardiasis, which is a notorious health problem throughout the world. Giardia is a member of the diplomonads, often described as an ancient protist group, whose primitive nature is suggested by the lack of typical eukaryotic organelles. Recent completion of the Giardia lamblia genome sequence gives a very good opportunity to have a better understanding of this parasite. In particular we are targeting mitosomes, an organelle that resembles a mithocondria, which provides a very good starting point to locate proteins that could provide a target to develop new antigiardiasis drugs.

For mitosomes we seek markers analysing 2,542 hypothetical proteins obtained from http://giardiadb.org/giardiadb. Within this setting our goal is to identify a target group within these hypothetical proteins. Subsequently our work involves that we pass the results on to the Laboratório Biologia Molecular de Parasitas e Vetores, FIOCRUZ, Brazil, for biological analysis.

5.2 Giardia lamblia

Giardia lamblia 5.1 (also known as Lamblia intestinalis and Giardia duodenalis) is a protozoan parasite of humans. It was first observed by Antony Van Leeuwenhoek (best known for his work on the improvement of the microscope) in 1681 [62]. It was Vliém Dušan Lambl who described the parasite in great detail in 1859, but it was only by the mid–1980s that Koch's postulates (these are four criteria to establish causal relationship between a causative microbe and a disease) to establish the pathogenicity in humans were formally described [62].

Figure 5.1 shows the Giardia cyst, which is approximately 7–10 μm in length and oval in shape. They are environmentally resistant, and the cyst remains viable for several months in cool and moist conditions. The cyst is able to survive standard concentrations of chlorine used in water purification systems [73].

Giardiasis prevalence in patients with diarrhoea is about 20% (range 5–43%) in developing countries. In developed countries giardiasis is prevalent among hikers and campers, people who swim in public pools and children who attend day care, with percentages varying from 3% to 7% [176].

Giardiasis is spread via the fecal-oral route, frequently through the ingestion of contaminated water or food. Symptoms vary from person to person, but the infection generally produces diarrhoea for more than 10 days, abdominal pain, flatulence, bloating, vomiting, and weight loss.

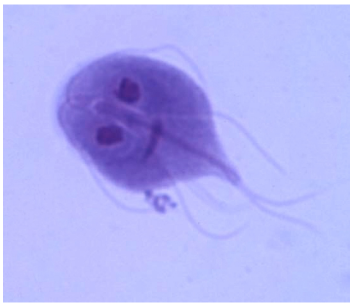


Figure 5.1: *Giardia lamblia, the binucleate structure of the cell and its appendages are clearly visible. Retrieved from the National Institute of Infectious Diseases of Tokyo, Japan.* http://www.nih.go.jp/niid/para/atlas/japanese/lambl.html.

The literature about Giardia is abundant, for additional references regarding Giardia see [4,56,160], for the biology of giardiasis see [149], for a diagnosis review see [61], and for treatments see [72,185].

Giardia genetic code has been recently sequenced and published (2007). The sequence is approximately 12 million base pairs and contains approximately 5,000 protein coding genes [119]. Regarding genomics the main information source is *GiardiaDB* (`http://giardiadb.org/giardiadb`) which is the central point of reference for the community that studies this parasite.

In the next section we briefly introduce how the genes are encoded.

5.3 DNA and proteins encoding

In very simple terms the deoxyribonucleic acid (DNA) is a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms. The main role of the DNA molecules is the long-term storage of information. DNA is stored inside the cell, organised in complex structures called chromosomes. During the cell division process (e.g. mitosis or binary fission process) chromosomes are duplicated to be ready for the new cells. This complex process is called DNA replication.

Inside the nucleus, the DNA molecules are composed of four different bases (Guanine, Cytosine, Adenine, Thymine), called nucleotides, and the order of these bases encodes the information carried in DNA. These four base pairs follow certain rules, called Watson–Crick base-pairing, and create two related strands. Watson and Crick discovered that the Adenine (A) paired with Thymine (T) and Cytosine (C) paired with Guanine (G) therefore create two complementary base sequences. These strands are oriented and running in opposite directions, creating a double helix [174].

In genes we can find sequences that carry the information necessary to encode a protein. The creation of the proteins and their related functions starts from sequences of DNA.

Figure 5.2 depicts the twenty standard amino acid translations and their three-letters and one-letter abbreviations. Proteins are necessary to living organisms because they process chemical reactions (working like enzymes). They are molecules made of amino acids arranged in a chain and folded into a spherical shape. This chain of amino acids is defined by the sequence of a gene. The process to create a protein from the DNA is called protein synthesis. Proteins can work together or alone to achieve their goals or to help the cell works (e.g. cell signalling, immune responses, cell adhesion, and the cell life cycle).

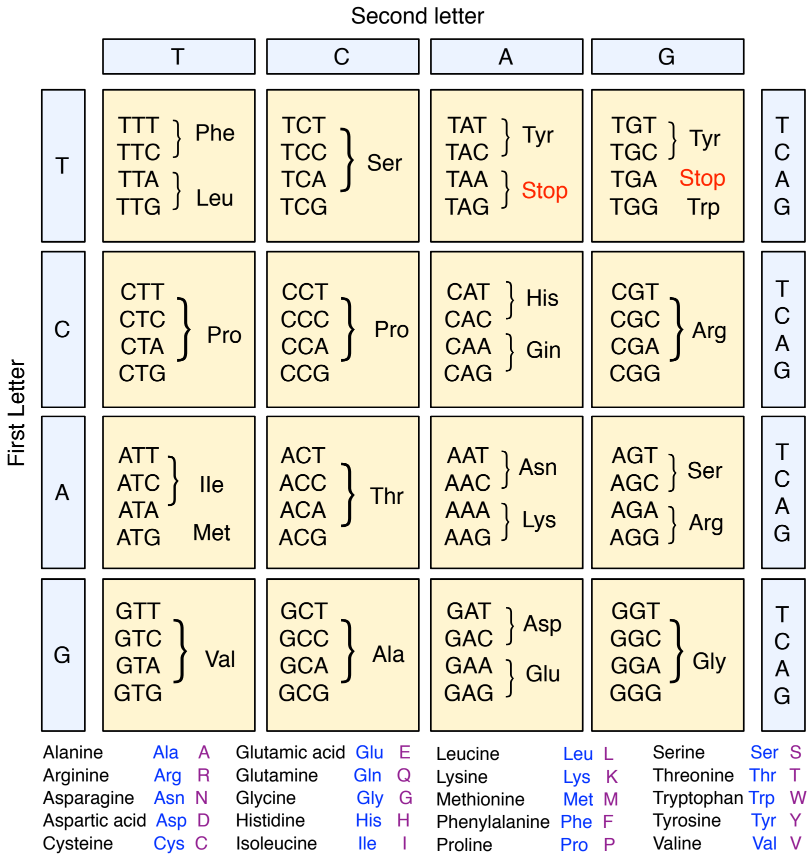
Literature in genomics and cell biology is very extensive. For additional information regarding the DNA and cell biology see [106,174]. For an introduction to bioinformatics see [30,102,179,191]. For protein interaction network see [187], and for microarray image analysis see [65].

5.3.1 Hypothetical proteins

We have seen in the previous section that proteins play a very important role in the living organisms, and they govern the traffic in the numerous metabolic pathways, which form the scaffold for numerous cellular structures. Many amino acid sequences forming proteins are well known and understood. But there is a very big number of proteins that have been annotated as unknown due to absence of high homology with other proteins included in genomic databases, although they exist and can be validated by the various biochemical, biophysical and genetic techniques. Such proteins are called "hypothetical".

The hypothetical proteins present a very good opportunity for the creation of drugs. The hypothetical protein may be specific to the particular organism in study. Therefore if the protein's function is studied and described, let us say in the case of a parasite, a drug can be produced in order to kill this organism without affecting the host.

Thus, in the Giardia case the benefit of studying and describing the hypothetical proteins is critical to produce a cure.



Alanine	Ala	A	Glutamic acid	Glu	E	Leucine	Leu	L	Serine	Ser	S
Arginine	Arg	R	Glutamine	Gln	Q	Lysine	Lys	K	Threonine	Thr	T
Asparagine	Asn	N	Glycine	Gly	G	Methionine	Met	M	Tryptophan	Trp	W
Aspartic acid	Asp	D	Histidine	His	H	Phenylalanine	Phe	F	Tyrosine	Tyr	Y
Cysteine	Cys	C	Isoleucine	Ile	I	Proline	Pro	P	Valine	Val	V

Figure 5.2: *The genetic code and the twenty standard amino acid translations, which are listed with their three-letters and one-letter abbreviations.*

5.4 Genome databases and file formats

There are many attempts to collect all known genes into a single database for analysis, and also there have been many efforts to create automatic tools to extract and analyse genes by means of specific queries.

Database structures can be classified mainly in four different types:

- Flat file databases
- Relational databases
- Object oriented databases
- Other databases

In particular we are interested in flat files because it is the format used by *GiardiaDB*, thus the format in which we have our files for analysis. Within this category the most common formats are FASTA, EMBL, GenBank, UniProt, and XML. Specifically we work with files in the FASTA format [142, 145] (also called Pearson format) that look like the following:

```
>gb|GL50803_10013 | organism=Giardia_lamblia_ATCC_50803
| product=Hypothetical protein | location=CH991782:800122-800784(+)
| length=220
MDDSGNDIGEQLRTKINSLRTQLATLKDSARSAIQDDDIARLALLQGKVTEISGALTSLK
AEFKALTLSDNLTRELDQLLVFLETGVNSLKSELRQADRGPVVTTVDVPTVVTREVLQTL
DTKKDNEILKLADNVKIIAEVNNRINEKLDEGAEVMEDVDTEIVAAQEKIDQAVGRMKLF
QSYMKKTKVPASVCILTFIFLIIVWSSKAFCSWGFTWQCP
```

Here the text is broken into more than one line for display purposes. The first line starts with the symbol “>” and contains the protein’s name, description and necessary encoding for localisation. The second part contains the amino acids.

There are many tools available to work with these formats. For convenience we built a simple parser that stores the FASTA data structure in two vectors. The first vector contains the protein header, and the second vector contains the amino acid. In this way it is easier to walk through the proteins in order to do comparisons within our code. For example, one of our first tasks was to retrieve only the hypothetical proteins from the Giardia genome sequence, with the sequences stored in vectors, and this was very quickly done.

5.5 Data characterisation

In order to understand the data our first task was to analyse Giardia’s amino acid frequencies of the hypothetical and non-hypothetical proteins. This information is useful for the biologist because amino acids have different functions. For example, a biochemical profile of hypothetical versus non-hypothetical proteins can be identified.

Tables 5.2 and 5.1 show the statistics for Giardia hypothetical and non-hypothetical proteins.

	Hypothetical	Non-Hypothetical
Proteins	2542	2347
Min. length	33	40
Max. length	1861	7449
Amino acids	1218227	1444928
Average	479.24	615.651
Std. dev.	701.62	545.01

Table 5.1: *Statistics for hypothetical and non-hypothetical proteins.*

	Hypothetical		Non-Hypothetical	
A.A.	Freq.	Freq. %	Freq.	Freq. %
G	57140	0.046904	87831	0.060786
A	93956	0.077125	120180	0.083174
P	60202	0.049418	61155	0.042324
V	72717	0.059691	87301	0.060419
I	70406	0.057794	81711	0.05655
L	130912	0.107461	144289	0.099859
F	44796	0.036771	47121	0.032611
M	27337	0.02244	33855	0.02343
S	117050	0.096082	121292	0.083943
C	22733	0.018661	43785	0.030303
T	77863	0.063915	91388	0.063247
N	48703	0.039979	60486	0.041861
Q	50887	0.041771	54855	0.037964
H	30085	0.024696	33098	0.022906
Y	41444	0.03402	47598	0.032941
W	7589	0.00623	9989	0.006913
D	65479	0.053749	80522	0.055727
E	70762	0.058086	87122	0.060295
K	60753	0.04987	77553	0.053673
R	67413	0.055337	73797	0.051073

Table 5.2: *Amino Acid frequency of occurrences for hypothetical and non-hypothetical proteins, where A.A.: amino acid; Freq.: amino acid frequency; Freq. %: normalised amino acid frequency.*

Figure 5.3 depicts the normalised frequencies of all amino acids found in Giardia. Figure 5.4 shows the Giardia lamblia hypothetical protein lengths arranged in descending order.

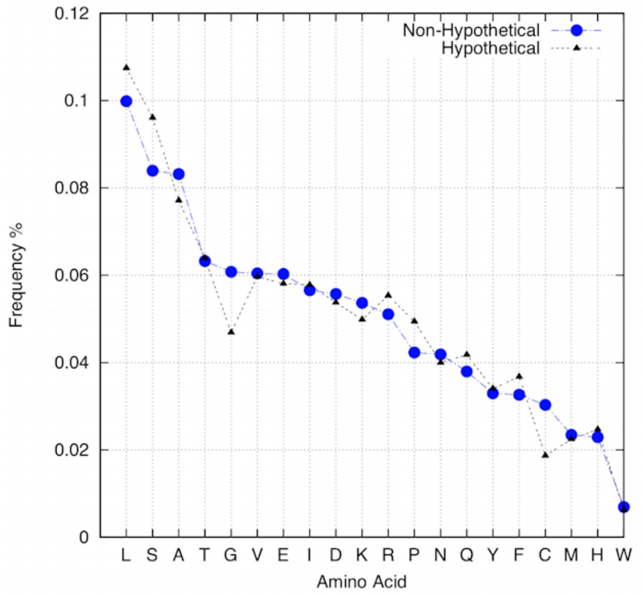


Figure 5.3: *Giardia lamblia protein frequency distribution in percentages.*

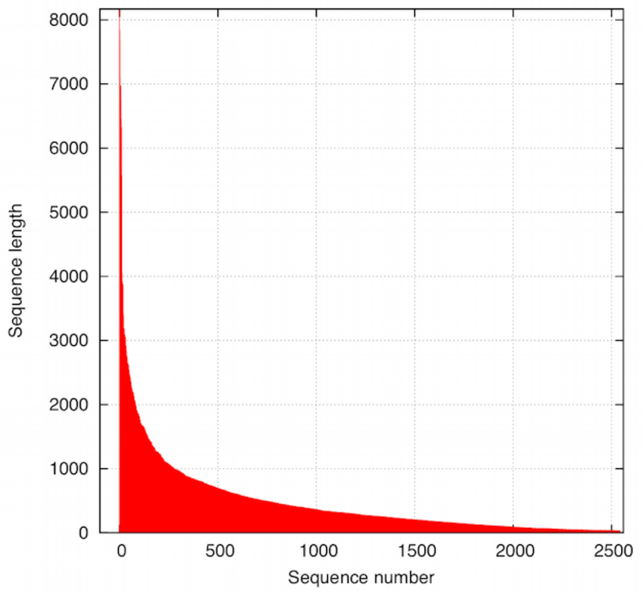


Figure 5.4: *Giardia lamblia hypothetical proteins lengths arranged in descending order.*

5.6 Clustering on strings

Working with strings is critical for clustering not only in genomic but in many other areas. In this section we introduce suffix arrays for storing strings, which later on is used to create clusters based on the longest common substring.

5.6.1 Suffix array for searching and matching

A suffix tree, or trie (which is abbreviated from "Retrieval"), is a data structure for storing strings in such a way that look-up is easy and fast. This data structure can be used as a list of keywords or a dictionary in its simplest form. Furthermore, by associating each string with an object it can be used as an alternative to a hash map.

Suffix trees can be used to solve the exact string matching problem in linear time, but the real advantage of using this data structure is its ability to solve more complex problems than the exact match.

Suffix trees and their applications are very well documented. A very good exposition about this topic is given by Gusfield [80]. In addition and as a starting point the reader interested in algorithms on strings can refer to Crochemore et al. [35], Charras and Lecroq [29], and Navarro and Raffinot [137]. For relevant papers see Navarro [134], Navarro and Mäkinen [136].

Table 5.3 shows how the suffix tree for the string "mississippi" is built. The first step is to allocate the string S in a data structure, a common selection is an array. Then prefixes of the string S are subtracted one by one as shown in Table 5.3 first column. This operation is repeated until the string S is empty. Now what we effectively have is a decomposed string S of length m into m substrings. In the case of the string "mississippi" of length 11 it has been decomposed into T_{11} substrings. Also we can see in the second column of Table 5.3 that this array can be lexicographically ordered for fast access and retrieval.

Figure 5.5 shows the tree representation for the string "mississippi". Here a suffix tree T is built for a string $S[1...m]$. The tree is rooted and directed with m leaves, which are numbered from 1 to m. Each edge is labelled with a non-empty

Position	Text	Sorted pos.	Text
$T_1 =$	mississippi	$T_{11} =$	i
$T_2 =$	ississippi	$T_8 =$	ippi
$T_3 =$	ssissippi	$T_5 =$	issippi
$T_4 =$	sissippi	$T_2 =$	ississippi
$T_5 =$	issippi	$T_1 =$	mississippi
$T_6 =$	ssippi	$T_{10} =$	pi
$T_7 =$	sippi	$T_9 =$	ppi
$T_8 =$	ippi	$T_7 =$	sippi
$T_9 =$	ppi	$T_4 =$	sissippi
$T_{10} =$	pi	$T_6 =$	ssippi
$T_{11} =$	i	$T_3 =$	ssissippi
$T_{12} =$	(empty)		

Table 5.3: *List of suffixes in the string "mississippi": Left hand side shows the degeneration of the string into substrings; Right hand side shows the suffix strings sorted lexicographically.*

substring of S. The internal nodes of the tree (not including the root) have at least two outgoing edges, and the labels of all outgoing edges are labelled with different characters. By following the path from the root to the leaf i and concatenating the edges labels, the suffix $S[1...m]$ is obtained.

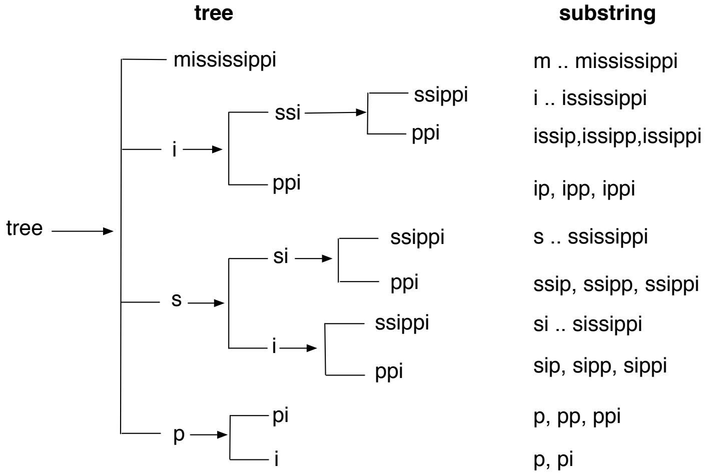


Figure 5.5: *Suffix tree representation for string "mississippi".*

When there is more than one string to be searched a *generalised suffix tree* (GST) can be built. In this case each leaf node contains the string suffix and an index or identifier to the string. For example, in the GST for "cagca" and "gagcga", the suffix “agcga” belongs to the string 2 and the suffix “agca” belongs to the string 1 [103].

Once the suffix tree has been constructed, finding all the occurrences of any pattern $P[1...n]$ in the string S takes time $O(n+k)$, where k is the number of times that the string S appears in the text.

Note that the “longest common substring” problem is different to the longest “common subsequence” problem which is closely related to the “edit–distance” problem: An instance of a subsequence can have gaps where it appears in string 1 and in string 2, but an instance of a substring cannot have gaps.

5.6.2 A note on DNA and protein embedding in ultrametric spaces

The Baire metric as defined in chapter 2 only considers the longest common prefix distance. In the current setting this was applied to protein sequences, which did not give any significant result. From the biological viewpoint it makes sense to split a protein into amino and carboxy terminal. Thus, we also applied the Baire (ultra)metric definition to find a hierarchical clustering when proteins are separated into these regions, which however gave rise to poor results. We concluded that the longest common prefix is not the right distance to apply within the specific context of the Giardia genome. Although slightly modifying this definition we can obtain clusters not based on the longest common prefix, but on the longest common substring. This is very closely related to the traditional way in which genome alignments are carried out.

Therefore, a very reasonable question to ask is if proteins are ultrametric? Clearly the Giardia genome is not when following the traditional genome sequence encoding. Encoding is very important when clustering in ultrametric spaces. Murtagh in [129] explains how data recoding can affect ultrametric clustering.

Khrennikov and Kozyrev [98] have proposed a p-adic number representation and ultrametric topology to investigate the genetic code. In particular Khrennikov and Kozyrev show that degeneracy of the genetic code can be described by local constancy of some map defined in an ultrametric space.

In the same line Dragovich and Dragovich [50] show that a 5-adic number representation is appropriate for DNA sequence embedding. It follows that this 5-adic distance is also suitable for representing genetic code degeneracy, which is related to the p-adic distance between codoms. This is a result that is consistent with the Khrennikov and Kozyrev findings.

An example comparing their two p-adic systems is as follows.

In the Khrennikov and Kozyrev work, for nucleotides A,T,G,C, in DNA, use the binary encoding $A = 00, G = 01, T = 10, C = 11$. Therefore each nucleotide has the code $\sum_i c_i p^i$ for $p = 2$. Hence for G we have: $0 \times 2^0 + 1 \times 2^1$.

In the Dragovich and Dragovich work, the coding of nucleotides for RNA is: $C = 1, A = 2, U = 3, G = 4$. These are encoded as digits in a p-adic system where $p = 5$. An example for the mitochondrial code UCG is as follows: Digits 3,1,4 encode as: $\sum_i c_i p^i$ for $p = 5$, $c = \{3, 1, 4\}$. Hence we have the code for the UCG target: $3 \times 5^0 + 1 \times 5^1 + 4 \times 5^2$.

In both these p-adic number representations, we are dealing with genetic codes that are expressed as p-adic numbers.

In a more recent work Khrennikov and Kozyrev [97] apply 2-adic numbers to the structure of the PAM (Point Accepted Mutation) matrix, showing that a PAM matrix A allows for expansion into the sum of two matrices $A = A^2 + A^\infty$, where the matrix A^2 is 2-adically regular, and the matrix A^∞ is sparse.

PAM matrices are used for sequence alignment in the same way that BLOSUM (BLOcks of Amino Acid SUbstitution Matrix) matrices are used (PAM matrices are built using a Markov Chain model). PAM and BLOSUM matrices are score matrices based on the observed frequencies of amino acids occurrences. While PAM matrices are based on global alignments of closely related proteins, BLOSUM matrices are based on local alignments.

5.6.3 Results from the longest common substring

In order to cluster the Giardia hypothetical proteins based on the longest common substring, a suffix array was built to compare pairs of all the 2,542 hypothetical sequences. As a result a symmetric squared matrix was obtained, where the rows and columns represent the protein sequences. For example, the first row presents the first hypothetical protein and each column its distance with the other proteins until the 2542*th* column.

This matrix of distances is too large for easily seeing what are the largest subsequences of strings. Thus we assigned to each cell in the resulting matrix a colour for easy visualisation. This was done with the R statistical software with a customised script based on Seidel's [156] script. Thus, a colour map with a gradual colour scaling is created, where blue shows distances that are closest to a lower bound value, and yellow distances that are closest to upper bound value.

Results are depicted in Figure 5.6, where the top panel shows the distance between the values of 0 and 10. Note that resulting distances are mainly very small, even for sequences with a large number of amino acids. For this reason we assigned a scale from 0 to 10, which is the range for most distances. Furthermore the diagonal of this matrix is given by the length of the sequence. For instance, if the first protein string is of length 100, then we have that the longest common subsequence is 100 when compared with itself. If the second protein string is of length 140, then the longest common substring in cell 2,2 is 140. And so on for all sequences.

Note that in Figure 5.6 top panel the diagonal is not shown. This is because the distances are well above the 0 to 10 range shown in this image.

In order to understand better the obtained result the original matrix of distances was rearranged. Largest distances were sorted towards the diagonal as depicted in Figure 5.6 bottom panel, where patterns become clearer.

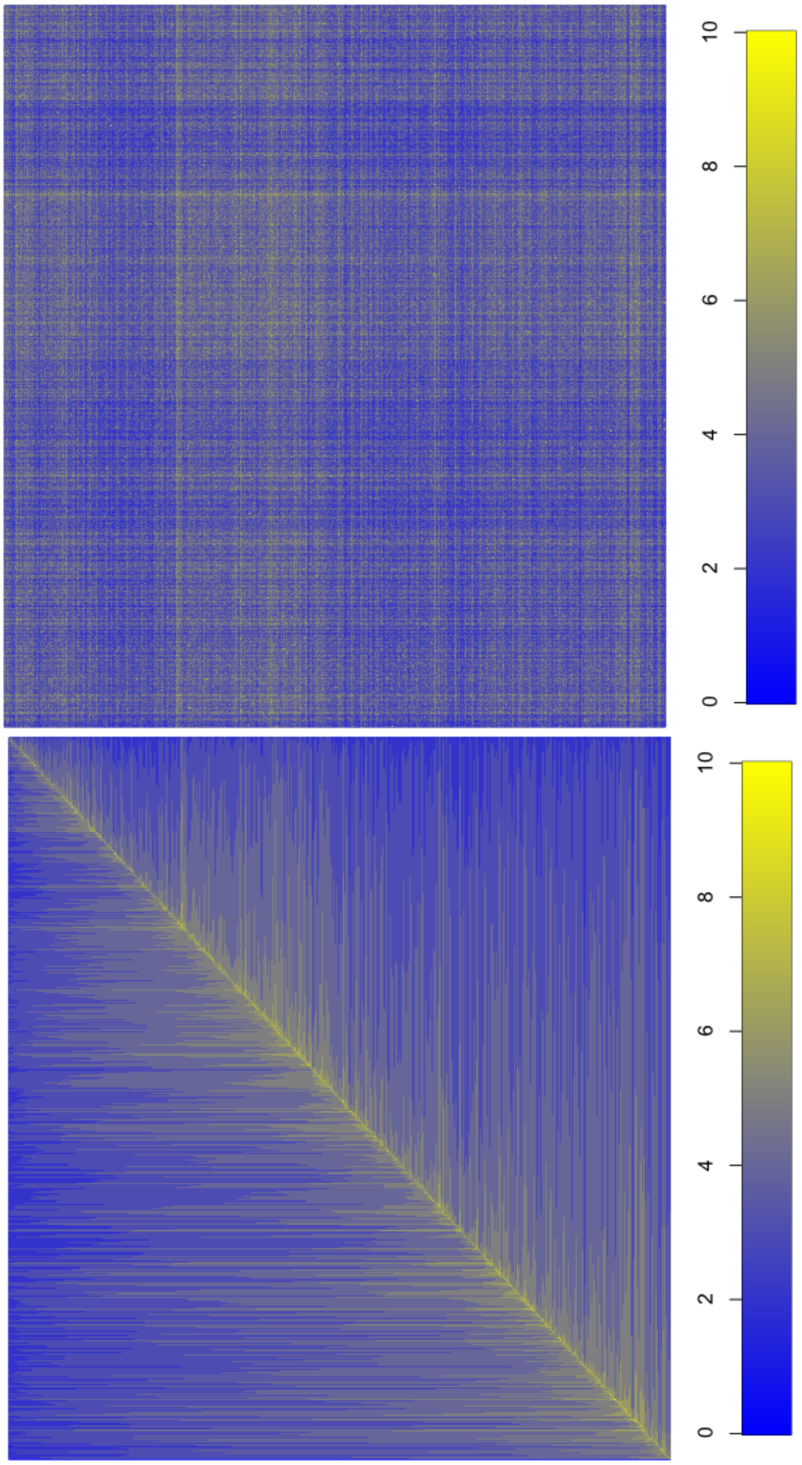


Figure 5.6: *Longest common substring images: Top panel shows the unsorted distances from 0 to 10 for the Giardia 2,542 hypothetical proteins; Bottom panel shows the sorted distances from 0 to 10 agglomerated towards the diagonal.*

The 10 longest common substrings from the similarity matrix are shown as follows:

Hypothetical protein results for the 10 biggest distances in longest common substring matrix.

```
>gb|GL50803_112017: 218   >gb|GL50803_114014: 304  length 315
>gb|GL50803_112893: 255   >gb|GL50803_112937: 258  length 310
>gb|GL50803_113470: 281   >gb|GL50803_135870: 631  length 655
>gb|GL50803_113740: 291   >gb|GL50803_114779: 328  length 286
>gb|GL50803_115328: 341   >gb|GL50803_115337: 342  length 329
>gb|GL50803_115439: 351   >gb|GL50803_115468: 352  length 300
>gb|GL50803_115469: 353   >gb|GL50803_115478: 354  length 739
>gb|GL50803_11696 : 411   >gb|GL50803_116967: 412  length 290
>gb|GL50803_117164: 423   >gb|GL50803_117191: 424  length 317
>gb|GL50803_9099   : 2372  >gb|GL50803_9283   : 2406 length 254
```

where the first column is the protein name, the second column is its location in the similarity matrix, the third column is the protein name, the forth column is its location in the similarity matrix, and the last column is the length of shared amino acids. For example, looking at the third row we have that the protein sequence 281 and 631 in the similarity matrix (i.e. proteins $gb|GL50803\_113470$ and $gb|GL50803\_135870$) share 655 amino acids in common. Thus, a preliminary conclusion is that for this set of proteins there is a high probability of common biological function.

Additional results show that when filtering the longest common substring distances matrix: 21 pairs of sequences have at least 200 amino acids in common; 32 pairs of sequences have at least 150 amino acids in common; 60 pairs of sequences have at least 100 amino acids in common; and 131 pairs of sequences have at least 50 amino acids in common.

Taking the last subset of 131 protein pairs, where at least 50 amino acids are shared by the hypothetical proteins, we have the following results:

Cluster results using the longest common substring.

1st cluster, 5 proteins

```
>gb|GL50803_112557 gb|GL50803_112893 gb|GL50803_112937 gb|GL50803_11305
gb|GL50803_113137
245 255 length: 64
245 258 length: 108
245 263 length: 70
245 266 length: 53
```

2nd cluster, 4 proteins

```
>gb|GL50803_112893 gb|GL50803_112937 gb|GL50803_11305 gb|GL50803_113137
255 258 length: 310
255 263 length: 59
255 266 length: 53
```

3rd cluster, 5 proteins

```
>gb|GL50803_11305 gb|GL50803_11196 gb|GL50803_112557 gb|GL50803_112893
gb|GL50803_112937
263 213 length: 55
263 245 length: 70
263 255 length: 59
263 258 length: 75
```

4th cluster, 6 proteins

```
>gb|GL50803_113137 GL50803_10256 gb|GL50803_112557 gb|GL50803_112893
gb|GL50803_112937 gb|GL50803_11305
266  62 length: 55
266 245 length: 53
266 255 length: 53
266 258 length: 53
266 263 length: 118
```

5th cluster, 4 proteins

```
>gb|GL50803_16803 gb|GL50803_10423 gb|GL50803_112059 gb|GL50803_114698
1089 105 length: 52
1089 220 length: 127
1089 326 length: 104
```

where coordinates show the position in the distance matrix and length is the common substring length of the sequences.

Note that sharing the same group does not necessarily mean sharing the same substring. For instance, if we take the 5th cluster results (where protein sequence number 1089 shares 52 amino acids with the sequence number 105, 127 amino acids with sequence 220, and 104 amino acids with sequence 326) the shared substring does not mean that they are the same amino acids. What we imply here is that sequence number 1089 shares a good number of amino acids with sequences 105, 220, and 326.

This kind of relationship is difficult to visualise clearly with the method shown in Figure 5.6. The size of the matrix is $2,542 \times 2,542$ which gives 6,461,764 cells. For example taking the 131 pairs of sequences that share at least 50 amino acids will only depict 131 bright yellow data points in the 6,461,764 space.

5.6.4 Results using the single-linkage clustering algorithm

In order to compare the clusters resulting from the longest common substring to other methods, experiments were carried out to obtain clusters from unaligned FASTA sequences using the single-linkage clustering method with the BLASTclust [48] program from the Bioinformatics Toolkit at the Max-Planck Institute for Developmental Biology [19] website (`http://toolkit.tuebingen.mpg.de/sections/classification`). A number of experiments were carried out with different parameters.

BLASTClust uses the BLOSUM62 [53] matrix for proteins. This involves gap opening cost 11, gap extension cost 1, and no low-complexity filtering. Additionally, the e-value parameter threshold is set to 1e-6 by default. For each pair of sequences the top-scoring alignment is evaluated according to the following criteria [47]:

```
              x1                   x2
              |                    |
sequence X ---======================-----
               \\|||||||||||||||||//
sequence Y ----====================------
               |                  |
               y1                 y2
```

where, high-scoring segment pair (HSP) length on sequence X: $H_x = x_2 - x_1 + 1$; gaps in sequence X: G_x; sequence X: L_x; basic local alignment search tool (BLAST) score: S; number of identical residues: N; sequence Y length: L_y; gaps in sequence Y : G_y; HSP length on sequence Y : $H_y = y_2 - y_1 + 1$; coverage of sequence X: $C_x = H_x/L_x$; coverage of sequence Y: $C_y = H_y/L_y$; coverage: $max(C_x, C_y)$ or $min(C_x, C_y)$, depending on the value of -b option; alignment length $Al = H_x + G_x = H_y + G_y$; score density: $S/min(H_x, H_y)$ or $N/Al * 100\%$.

Then, if the coverage is above a certain threshold and the score density is above a certain threshold, these two sequences are considered to be neighbours. The neighbour relationships determined in this way are considered symmetric and provide the basis for clustering by a single-linkage method (which associates a sequence with a cluster if the sequence is a neighbour of at least one sequence in the cluster).

Results are the following:

Using as parameters: lvalue = 100.

```
-------------------------------------------
```

1st cluster, 1 group of 7 proteins

```
>gb|GL50803_102575 gb|GL50803_111973 gb|GL50803_112630 gb|GL50803_112914
gb|GL50803_112938 gb|GL50803_113165 gb|GL50803_113130
```

2nd cluster, 1 group of 4 proteins

```
>gb|GL50803_116393 gb|GL50803_116394 gb|GL50803_120651 gb|GL50803_120652
```

3rd cluster, 10 groups of 3 proteins

```
>gb|GL50803_17342 gb|GL50803_112063 gb|GL50803_114777
>gb|GL50803_101451 gb|GL50803_112341 gb|GL50803_103785
>gb|GL50803_117391 gb|GL50803_117392 gb|GL50803_117393
>gb|GL50803_125104 gb|GL50803_125105 gb|GL50803_125106
>gb|GL50803_118540 gb|GL50803_118541 gb|GL50803_118542
>gb|GL50803_123978 gb|GL50803_123979 gb|GL50803_123980
>gb|GL50803_134501 gb|GL50803_134502 gb|GL50803_88320
>gb|GL50803_119853 gb|GL50803_119854 gb|GL50803_1370
>gb|GL50803_115670 gb|GL50803_115671 gb|GL50803_37793
>gb|GL50803_123278 gb|GL50803_123279 gb|GL50803_123280
```

Using as parameters: lvalue = 95.

```
-------------------------------------------
```

1st cluster, 1 group of 8 proteins

```
> gb|GL50803_16293 gb|GL50803_102575 gb|GL50803_111973 gb|GL50803_112630
gb|GL50803_112914 gb|GL50803_112938 gb|GL50803_113165 gb|GL50803_113130
```

2nd cluster, two groups of 6 protein

```
>gb|GL50803_117191 gb|GL50803_117192 gb|GL50803_10241 gb|GL50803_116967
gb|GL50803_116968 gb|GL50803_11149
>gb|GL50803_12830 gb|GL50803_20020 gb|GL50803_5206 gb|GL50803_101451
gb|GL50803_112341 gb|GL50803_103785
```

3rd cluster, 1 group of 4 proteins

```
gb|GL50803_116393 gb|GL50803_116394 gb|GL50803_120651 gb|GL50803_120652
```

4th cluster, 11 groups of 3 proteins

```
>gb|GL50803_90236  gb|GL50803_15404  gb|GL50803_10238
>gb|GL50803_17342  gb|GL50803_112063 gb|GL50803_114777
>gb|GL50803_117391 gb|GL50803_117392 gb|GL50803_117393
>gb|GL50803_119191 gb|GL50803_117436 gb|GL50803_117437
>gb|GL50803_125104 gb|GL50803_125105 gb|GL50803_125106
>gb|GL50803_118540 gb|GL50803_118541 gb|GL50803_118542
>gb|GL50803_123978 gb|GL50803_123979 gb|GL50803_123980
>gb|GL50803_134501 gb|GL50803_134502 gb|GL50803_88320
>gb|GL50803_119853 gb|GL50803_119854 gb|GL50803_1370
>gb|GL50803_115670 gb|GL50803_115671 gb|GL50803_37793
>gb|GL50803_123278 gb|GL50803_123279 gb|GL50803_123280
```

where *l* value indicates the minimum length coverage.

Comparing the result obtained from using the longest common subsequence and the single-linkage clustering method it can be observed that resulting clusterings are not the same. This is due to the way the single-linkage method groups clusters, and to the fact that we do not use amino acid normalisation scoring matrices when using the longest common substring (BLOSUM or PAM). The main reason we do not use scoring matrices is based on the fact that hypothetical proteins are not known, thus assigning any predefined schema for scoring (which in fact can be seen as a normalisation process) can distort the resulting groups. Furthermore many scoring methods are available which inevitably will lead to different results.

5.7 Summary

The Baire (ultra)metric proposed in chapter 2 is based on the longest common prefix. In the case of biological data, this approach cannot be applied directly to the Giardia genome sequence case because proteins do not behave in a hierarchical manner when embedded in the traditional 20 amino acid letters space. This distance can be modified from the longest common prefix to the longest common substring. This results in a symmetric distance matrix where all proteins are compared against each other. Based on this distance matrix we obtain clusters based on the longest common substrings.

We find that using the longest common prefix as defined in chapter 2 does not perform well due to the very nature of amino acid sequences. In the Giardia case proteins are not similar from the beginning of the sequence when compared with each other. We modified the longest common prefix distance to the longest common substring distance. In this case we find many similarities, which means that amino acid subsequences are common throughout the Giardia hypothetical proteins.

Additionally we use the BLASTclust program to produce clusters based on the single-linkage clustering algorithm. Comparing the result obtained from using the longest common subsequence and the single-linkage clusterings method it can be observed that resulting clustering are not the same. This is due to the way the single-linkage method groups clusters, and to the fact that we do not use amino acid normalisation when using the longest common substring (BLOSUM or PAM scoring matrices).

Part II

Application to Information Retrieval

Introduction to Part II

IN this part we deal with the application of the Baire (ultra)metric to information retrieval and clustering of text.

Based on **Part** I of this thesis we show how our algorithm can be applied to data from different scientific disciplines. In this part our work illustrates well:

- The unity of the mathematical and computational underpinning of our method applied to the enterprise.
- The ability to take algorithms that have been developed and validated in one field, e.g. astronomy, and use them successfully in other fields, e.g. business.

In **Chapter** 6 we give a general overview of the processes involved in implementing a search solution. Furthermore, we describe the context of our work in the enterprise setting.

In **Chapter** 7 we deploy our algorithm in the enterprise setting, and in the heritage searching context. An important part of this chapter relates to the specific nature of enterprise search, and moreover how it differs from, say, Web search.

Chapter 6

Supporting Massive Best Match Search and Retrieval

6.1 Introduction

IN this chapter we look into the main processes to build a search engine from the information retrieval perspective. In general we have that the main steps for such a system are the same in different scenarios, namely a search engine for the Internet, the enterprise, email or any other context. However there are particularities that are very important in different cases, which will effectively determine if the system is finally usable.

For example, when searching the Web all HTML files are publicly available (with exception of protected directories) and crawled by robots (here again exception policies can be implemented) such as by Yahoo! and Google. This is not necessarily true in the case of *enterprise search*, because different level of security are needed for different users, and also results may have to be ranked with criteria that are different to the Web case.

6.2 Structuring and searching text in a massive dataset

Figure 6.1 shows a schematic representation of the system that we are aiming to address. Typically a user working on his/her computer would like to backup information contained within a specific directory (we refer only to one directory for the sake of simplicity). This directory may contain documents in several formats such as MS Word, MS Excel, MS PowerPoint, e-mails, PDF, HTML, text, etc. A commercial application needs to handle all these different formats to be successful. Producing parsers for each of these formats falls well out of the scope of this work. Therefore, we are to concentrate on plain text, since this is the simplest of cases and it is well suited to perform experiments (also we avoid the hassle of dealing with proprietary formats such as MS Office). Our interest is to design a system that if successful with plain text, later can be applied to other formats.

Coming back to Figure 6.1 we can see that a **local collector** takes care of storing parsed tokens. These tokens are to be compressed and encrypted before being sent to a centralised repository for backup purposes. Note that also the documents are sent to the data centre to be stored.

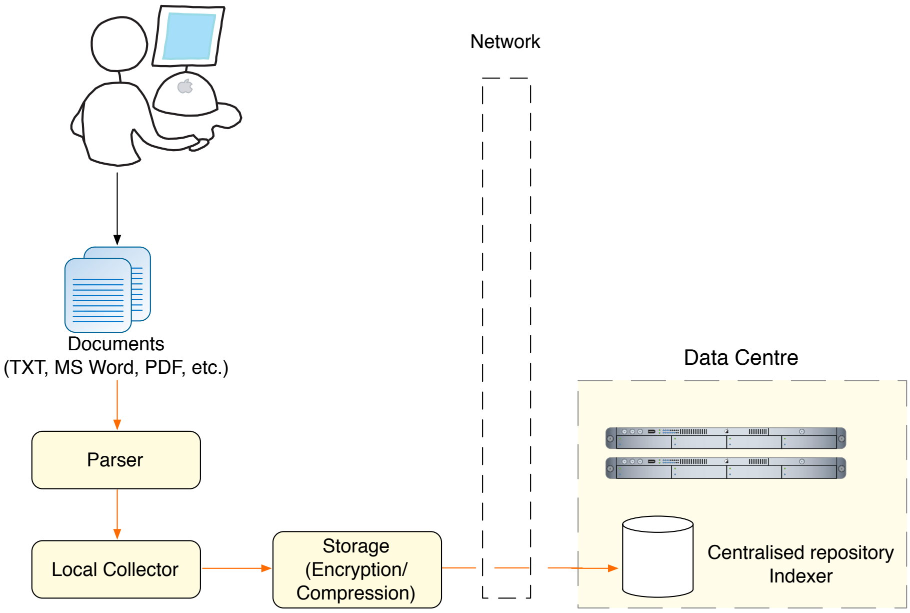


Figure 6.1: *Architecture overview. In general a document source is selected and parsed, the local collector stores resulting key words. In turn these are compressed and encrypted, then sent to a central repository for storage.*

The scenario presented above is a simplification of a real life situation, but gives us a good base as a starting point. A real life scenario would include a situation where there are different units (e.g. sections, groups, departments, etc.) within a corporation backup information. Then collectors for these different units may interact in order to send information to the central repository (indexer).

A situation closer to reality would be the one presented in Figure 6.2. Here we can see that different collectors feed an **indexer process** that in turn stores information regarding tokens and document locations embedded in the database (DB). This DB can later be queried to match tokens (or words) against documents (at this point we are not considering compression nor encryption).

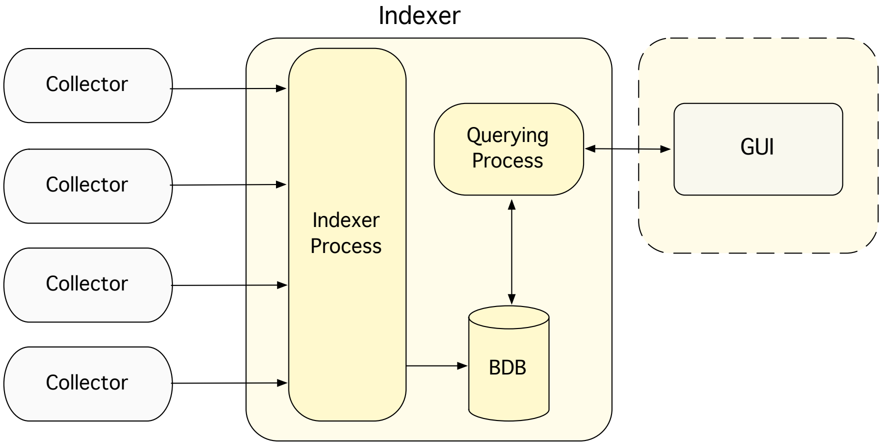


Figure 6.2: *Indexing process overview. Each collector process stores parsed documents (key words) that are sent to an indexing process, which is in charge of merging collectors into a database ready to be queried.*

6.2.1 On searching

In this section we approach the question of what kind of searching system we aim to build. Searching and retrieving the right information depends on many factors. A similar query means different things to different people. Thus searching and retrieving the "right" information depends greatly on context and meaning (i.e. semantics), and how the query system handles these factors. For example, a system that searches documents on the Web is different from a system that searches images and videos. Searching in a single purpose system (let us call this "specialist search system") such as fingerprints, astronomical, financial or economical databases require different techniques and approaches if compared with the Web. Also a deeper knowledge is required from the user. To complicate things further user expertise is relevant when retrieving information, e.g. two users working on the same search system and trying to retrieve the very same information may have different approaches when searching due to their experience and expertise.

When starting to build a retrieval system some good questions to start to think about are the following:

– Who are the users?
– What are the users looking for?
– What kind of information will the system be handling (text, numeric, etc.)
– Information context and meaning.

We will be dealing with a specialist search system, that is to say that very few people will have access to it and the users would be looking for forensic types of information. By forensic information we mean that the user will be doing exhaustive queries to answer questions such as: a) retrieve all documents that contain a particular person's name, b) retrieve all documents related to certain projects, c) retrieve documents produced on a particular date. By exhaustive search we mean that all matches should be retrieved because all matches are relevant.

A realistic scenario would be something like this: a firm has been sued and needs to retrieve and destroy all documents that mention a person's name, if not the company will be liable to pay compensation. The search engine should be able to pinpoint all such documents and produce valid locations for them.

6.2.2 On indexing

In this section we explain in detail the collector component on the client side of the system. Additionally some restrictions regarding development tools and programming language are introduced.

Figure 6.3 shows the collector components: Parser and CollectorStorage Engine. The parser is in charge of taking information (tokens) from files that are contained within a directory (and sub-directories), then extracting and filtering valid tokens from these files. Note that directories should be walked through, since a data source is considered to be a directory and all its dependencies.

Tokens are defined as words, therefore at this stage any token that is not a word will not be included (e.g. symbols +, –, &, $ and numbers).

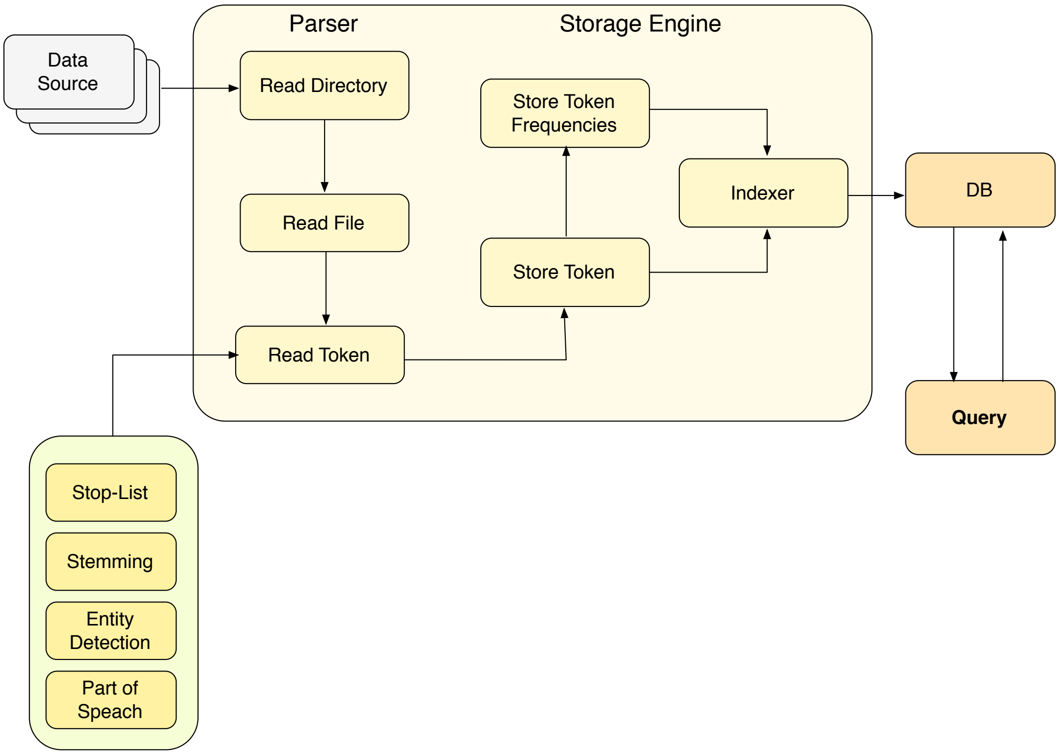


Figure 6.3: *Collector overview. First a data source is defined, in this case a directory is specified. All files within are read and tokens identified. At this point a stop-list, stemming, entity detection and part of speech software can be used to help with the parsing. Afterwards tokens are stored, and frequencies relative to a document identified. Finally, this information is sent to the central indexer to be merged and stored in a DB.*

Additionally the parser excludes words that are considered not relevant (e.g. a, the, and, etc.). These are included in a stop-list file, and thus this file is a further filter in the parsing process. Note that stop-words are language dependent and there should be one stop-list per language.

When a valid token has been identified – this is a token that is neither a symbol, number nor a word in the stop-list – it should be passed to the CollectorStorage, which is in charge of storing tokens and their frequencies for a given document.

The CollectorRecord is effectively an intermediate form of metadata that is to be stored in the final index. The important point is that no ranking or placement within the actual index has taken place on the computer running the Indexer.

By using an intermediate form of the search metadata, the process of creating a search index can be distributed. The other advantage is that computational impact of indexing data does not adversely affect the server, which provides key services during the course of the business day.

The Collector component incorporates the parser module. One key part of this system is devising the storage of the search metadata. Any compression techniques that can be applied to the metadata will be advantageous to the implementation. Ultimately this will reduce the amount of data Collector will have to transmit up to the Indexer. For compression the zlib [67] compression/decompression library is to be used. Optionally the parser generated metadata needs to be stored in an encrypted form.

Dealing with documents the Collector needs to be able to handle the following document scenarios.

- Document Additions – A new document is added to the corpus
- Document Updates – A document in the corpus has been updated
- Document Moves – The location of the file has changed
- Document Deletions – The document no longer exists.

The fact that a document has been renamed will be handled implicitly since it will be seen as an addition of a new document and a deletion of the old file.

The CollectorRecord has two forms, a local form for use by the Collector and another form for the Indexer, IndexerRecord. Inheritance can be used to relate the basic form to the specialised form.

Some of the information that the CollectorRecord should store is the following:

1. DocumentID – The local document identifier

2. Timestamp – The last time the record was updated
3. Filename – The filesystem name for the file
4. Location – The path to the file
5. MD5 hash – The MD5 hash of the content of the file
6. Token list – Including the token and frequency

The Timestamp is the time at which the Indexer created the CollectorRecord, the time-stamp format is YYYYMMDDHHMMSSTTT. Other formats are also valid, the important point is to have a format that does not change. This in order to keep the system simple.

The location and filename are stored separately because they are not exclusive, since a file can have the same name, but reside at different locations. A CollectorRecord can be matched to a document if the Filename and Location match a document being processed.

The Message-Digest algorithm 5 [148] (MD5 Checksum) of the content is needed for detecting file changes (important for document updates). In this case the indexing process should be re-applied to the document in question, causing the conversion and parsing process to re-occur. This is not needed in the IndexerRecord.

In addition the IndexRecord incorporates the Media Access Control address (MAC address) of the computer running the Collector. The simplest combination is to concatenate the 2 numbers together into a single string. This is needed so that the Indexer can identify the record's source.

e.g.

- MAC Address
- Sequence Number
- DocumentID

6.2.3 On retrieval

An important factor to consider is how the information will be retrieved; how the queries are built; and how they are sent to the servers. Query representation in a retrieval system is important because the effectiveness and efficiency will depend on a good design and implementation. In the following chapter, chapter 7, we look into queries in more detail. Next we provide a brief review of different query strategies.

6.3 Supporting massive best match search and retrieval

A retrieval strategy is based on an algorithm that takes a query Q and a set of document D (where $d_{i1}, d_{i2}, d_{i3}, .., d_{it}$ are individual documents) of length t, identifying a *similarity coefficient* used to specify distances between documents (or between documents and queries). Retrieval strategies can be classified as follows [79]:

- **Vector space model:** this presents the query and documents (terms) as vectors, then a similarity measure is applied between vectors.
- **Probabilistic retrieval:** this is a computer probability based on the likelihood that a term will appear in a relevant document.
- **Language models:** a language model is built for each document, and the likelihood that the document will generate the query is computed.
- **Inference network:** here a Bayesian network is used to infer the relevance of a document to a query. The strength of this inference is used to compute the similarity coefficient.
- **Boolean indexing:** a score is assigned such that an initial boolean query results in a ranking. This is done by associating a weight with each query term so that this weight is used to compute a similarity coefficient.
- **Latent semantic indexing:** terms in a document are represented with a term-document matrix. This matrix is reduced via Singular Value Decomposition (SVD). Then documents that have the same semantics should be located closer together.

- **Neural networks:** a neural network can be trained by adjusting the neuron's weights.
- **Genetic algorithms:** an optimal query to find documents can be produced by evolution. An initial query is used with an estimated weight. New queries are generated by modifying these weights. Then a new query survives by being close to known relevant documents and queries with less strength or "fitness" are removed.
- **Fuzzy set retrieval:** boolean queries are mapped into fuzzy set intersections and union. These associations are used as "strengths" in the same way as a similarity coefficient.

6.4 Building a search engine

6.4.1 Vector space model

A document set D composed of m documents indexed by n terms can be represented as an $m \times n$ *document-by-term matrix* A. Thus, the matrix element a_{ij} is the weighted frequency at which term i occurs in the document j [17]. It follows that this document representation is very important when querying documents. Then this vector space represents the columns of A denoting terms vectors, and rows denoting document vectors.

Several different ways exist of comparing a query vector with a document. The reader can refer to chapter 2 section 2.5.1 for more detail regarding commonly used similarity measurements. The most common of these in the text retrieval context is the cosine, where the cosine of the angle between the query and document vector is given by:

$$d(\mathbf{x}_a, \mathbf{x}_b) = \cos(\theta) = \frac{\sum_{i=1}^{n} \mathbf{x}_{i,a}\ \mathbf{x}_{i,b}}{\sqrt{\sum_{i=1}^{n} \mathbf{x}_{i,a}^2\ \sum_{i=1}^{n} \mathbf{x}_{i,b}^2}}$$

Other coefficients are the Dice and Jaccard, presented below. These are often used for boolean or "binary" data, representing presences or absences. The Dice coefficient is defined by the following:

$$D_c = \frac{2|X \cap Y|}{|X \cup Y| + |X \cap Y|} = \frac{2|X \cap Y|}{|X| + |Y|}$$

where $\cap$ depicts set intersection, $|\cdot|$ denotes set cardinality of the documents that we have taken as sets of terms that are present in the document.

In vector terms,

$$D_c = \frac{2 \cdot XY}{\|X\| + \|Y\|}$$

where (XY) is the inner product of X and Y, and $\|X\|$ is the Euclidean norm of X.

The Jaccard coefficient is defined as:

$$J_s = \frac{XY}{\|X\|^2 + \|Y\|^2 - (XY)}$$

6.4.2 Document querying

Figure 6.4 describes the three-tier [18] process generally involved when transforming a user's query into a search engine query. In the first phase we have a user using a GUI (graphical user interface) to formulate a query in a search engine, in the second level the query is transformed into tokens, and in the last stage the search engine must use the information given by the token to look for the relevant information within a database to retrieve the desired documents.

Generally we can identify the following types of queries:

- **Boolean:** AND, OR, NOT words are included in the search in order to include/exclude results.
- **Natural Language Queries:** here the query is formulated as a question or a statement.
- **Thesaurus Queries:** The user selects the term from a previous term-set provided by the information retrieval (IR) system.
- **Fuzzy Queries:** the threshold of relevance is expanded to include additional documents.
- **Term Searches:** is based on a few words or phrases provided by the user.

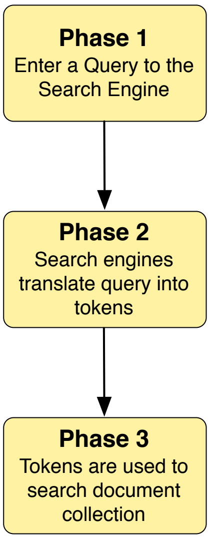

Phase 1
Enter a Query to the Search Engine
Phase 2
Search engines translate query into tokens
Phase 3
Tokens are used to search document collection

Figure 6.4: *Typical querying retrieval systems [18]. In phase one a query is written into the search engine, then in phase two this query is translated into tokens, which in phase three are converted to the vector space for search.*

- **Probabilistic Queries:** IR systems based on a computed probability to retrieve documents.

6.4.3 Term weighting

For text collections presenting many different contexts such as newspapers and encyclopaedias, the number of terms (n) is typically bigger than the number of documents (m), i.e. $n \geqslant m$. This is not necessarily true for every collection of text, on the Web for example the situation is the opposite. In order to improve retrieval performance *term weighting* can be used, meaning that relevant information about *word frequency* can be incorporated in an automatic indexing system.

Following the formulas presented in Tables 6.1, 6.3 and 6.2 a simplified weighting schema can be introduced [17, 18, 79, 152, 175].

Symbol	Name	Formula
B	Binary [152]	$\chi(f_{ij}) = \begin{cases} 1 & \text{if } (f_{ij}) > 0 \\ 0 & \text{if } (f_{ij}) = 0 \end{cases}$
L	Logarithmic	$log(1 + f_{ij})$
N	Augmented normalised term freq.	$\frac{\chi(f_{ij}) + \left(\frac{f_{ij}}{max_k f_{ki}}\right)}{2}$
T	Term frequency [152]	f_{ij}

Table 6.1: *Formulas for local term weights l_{ij} [18].*

Here, f_{ij} is the number of times (frequency) that a term i appears in document j.

Symbol	Name	Formula
X	none	1
E	Entropy	$1 + \left(\frac{\sum_i (p_{ij} \log(p_{ij})}{\log n}\right)$
F	Inv. doc. freq. (IDF)	$log\left(n / \sum_i \chi(f_{ij})\right)$
G	Global freq. IDF	$\frac{\sum_i f_{ij}}{\sum_i \chi(f_{ij})}$
N	Normal	$\frac{1}{\sqrt{\sum_i f_{ij}^2}}$
P	Probabilistic inverse	$\log\left(\frac{n - \sum_i \chi(f_{ij})}{\sum_i \chi(f_{ij})}\right)$

Table 6.2: *Formulas for global term weights [18, 51, 152] g_i.*

Here $p_{ij} = f_{ij} / \sum_j f_{ij}$.

Symbol	Name	Formula
X	none	1
C	Cosine	$(\sum_i (G_i L_{ij})^2)^{-1/2}$

Table 6.3: *Formulas for document normalisation [152] d_j.*

For example, with reference to Tables 6.1, 6.3 and 6.2, a simple notation for specifying a term-weighting scheme is to use the three-letter string associated with particular local, global, and normalisation factors. For instance the *L*-*F*-*C* weighting schema a_{ij} is defined by the following formula: $L_{ij}F_{ij}/C_{ij}$.

This can be rewritten as:

$$a_{ij} = \frac{log(1+f_{ij})\log\left(\frac{n}{\sum_i \chi(f_{ij})}\right)}{\sqrt{\sum_j \left(log(1+f_{ij})\log\left(\frac{n}{\sum_i \chi(f_{ij})}\right)\right)^2}}$$

In general the term weighting schema presented here is available in many traditional retrieval systems and is considered the basis from which further extensions and customisation can be applied.

6.4.4 Compressed inverted list indexes

In this section we deal with the document indexing problem. We assume that data preparation and extraction from a corpus has been carried out. Normally this will include at least word extraction and recording of frequencies of occurrence, but certainly additional steps can be taken in order to improve search results such as including a stop-list (i.e., terms frequently used but not needed in the index e.g. articles, prepositions and conjunctions), word normalisation (lemmatisation) see [143], entity detection (e.g. a person's name [14]) and part-of-speech tagging, to name a few. We call this process parsing.

Once a document has been parsed and a token (e.g., a word) identified this will be stored in an inverted indexed list [190]. This is an alphabetically ordered word list typically stored in a vector, where each word occurrence has a pointer to the document where it appears. Searching in this structure is reduced to looking for coincident words (string matching) and then retrieving the document occurrences.

Building inverted text indices for a large document corpus mainly involves three steps:

1. Build in-memory index,
2. Store memory index on disk,
3. Merge temporary indexes to a database.

Indexing text has expanded beyond recording word occurrences and merging indexes. It follows that a key objective of inverted index files is to produce algorithms to reduce I/O (input/output) bandwidth and storage overhead [26,59,136, 177].

Modern compression techniques are very useful to reduce search time and index size. Here we are interested in lossless compression techniques applied to inverted index files. For a lossy approach see [25].

Compression performance depends on the type of data that is handled. For example obtaining good compression rates in English text with a particular method may not be replicated when working with images or numerical data. Also it is important to highlight that generally speaking compression has a trade off with memory used: the better the compression, the slower the program execution. In other words more memory is needed.

Most compression techniques are based on the idea of sorting the inverted list in ascending order [79]. For example in [177], pp. 114-143, a number of compression techniques are used and evaluated over an inverted file index. In general for practical purposes local Bernoulli method using Golomb coding [167] and [177], p. 115, is the technique to use.

Certainly, compression techniques applied to inverted indexes are not limited to the above. The so called Block-addressing compressed indexes [135] reduce the index size by means of indexing blocks of text with fixed size. Other techniques include caching and early query termination.

From the Baire distance perspective an important compression method is Lempel-Ziv [188,189], because of its use of the longest common substring [159] to compress data, which is closely related to the Baire metric.

6.5 Enterprise search

In this section we describe the current technologies used in the enterprise search industry. Section 6.5 presents the main differences between the Web and enterprise search. This section also deals with the criteria to consider when evaluating and selecting current search implementations. Section 6.5.1 shows the main components in an information retrieval system. Finally section 6.5.8 presents a list of commercial enterprise search providers as well as the main solutions today offered by the open source community.

Differences between enterprise and Web search are very significant when looking into areas such as compliance, security and other important topics. In [46] ten issues that must be considered when building a retrieval system within an organisation are considered.

- **Security and privacy**: implementing a search system within an organisation should consider what information is to be indexed. This will prevent sensitive information (e.g. employee data, intellectual property documents, etc.) to be exposed unnecessarily.

- **Policy and compliance**: a well-designed enterprise search engine should follow closely the company's policies.
- **Access control**: access to information is limited by authentication, not all users are allowed to access the same information and controls must be placed to ensure the system is secure [135]
- **Comprehensiveness**: if a search is to be effective it needs to be comprehensive. Thus, access to different repositories, email systems, and business applications (structured and unstructured data) are necessary.
- **Relevant results**: internet search engines generally rank relevance based on document linkage. This is not necessarily true for enterprise applications where a relevance algorithm may be configured to look for entities within documents such as titles, names, dates, locations, etc.
- **Federation**: rather than re-indexing content across different retrieval systems, index integration (or federation) may be better in order to improve performance.
- **Personalisation**: enterprise search results can be improved by incorporating user information into the result, e.g. by relating a specific department (e.g. marketing, finances) for which the final result can benefit. Also search history may be an important factor to consider.
- **Search as a service**: designing a search system as a service (i.e. Web service) helps in order to present results on a variety of platforms. This is relevant for a call centre, a portal, an intranet site, etc.
- **Enterprise scale and scope**: growth must be always taken into consideration. Also multilingual support may be needed.
- **Support**: finally, support is crucial to maintain and update a search system.

Additionally to traditional search technology several aspects need consideration. In the ThinkingSAFE case the following list includes some of the most important characteristics to consider when implementing the search technology:

- **Document synchronisation**: the backup system has to handle document sync based on delta records (i.e. the differences between the current and previous versions). Then each record is processed to determine if a change (i.e. add, delete, or modify operation) has been made.

- **Query categorisation**: when querying the system exhaustive searching will be carried out, meaning a possibly large outcome. Then a mechanism to categorise or rank this outcome is needed.
- **System integration**: it is a must to consider currently used systems, e.g. the current ThinkingSAFE backup system. Any search capability will interact with such a system, and therefore integration issues may arise.
- **Performance**: indexation processes should have a small footprint and high performance. For the query system high performance also is needed. Performance measures and benchmarks have to be produced for these software components.
- **Usability**: graphical user interfaces are required to be intuitive, easy to use, and simple.
- **Compression and encryption**: perhaps the biggest challenge is to take advantage of compression techniques to maximise performance and to minimise storage. Encryption support is a tough question since a by-pass technique is needed to query data that has been encrypted. In principle this can be answered through keeping the index database in a secure location with restricted access. The documents can be encrypted.

6.5.1 Anatomy of a search engine

This section describes the main processes within a search technology. We separate this into six main areas: preprocessing, parsing, indexing, storing, querying and system administration.

6.5.2 Preprocessing

This involves converting raw documents into a suitable structure to be processed. By raw document we mean any kind of input that needs to be indexed. This process recognises the kind of document and transforms its content into a stream of text. Additionally processes such as language recognition can be implemented here and this can be passed together with the text stream to the next process. A great number of documents are used within an organisation so the preprocessing stage should be implemented in a modular way (i.e. plug-ins), allowing for inclusion of new formats such as:

- MS Office (mainly Word, Excel, in their different versions),
- PDF,
- e-mail repositories,
- HTML,
- RTF,
- Plain text,
- XML,
- Many others.

6.5.3 Parsing

We assume that data preparation and extraction from a corpus has been carried out. Normally this includes at least word extraction, but language identification and recording of word frequencies could have taken place. This data can be separated into two main sections, metadata (language, frequencies, etc.), and the text stream.

Parsing is the process carried out to improve the understanding of the text stream. Steps can be taken in order to improve search results such as including a stop-list.

6.5.4 Indexing

Once a document has been parsed and a token (e.g., a word) identified this will be stored in an inverted indexed list [190]. See section 6.5.3.

6.5.5 Storing

Storing deals with the indexing system and the way words or phrases are stored and retrieved. The main tasks to consider here are the following:

- Index merging,
- Index updating,
- Index deletion,
- Index compression,
- Index performance,
- Storage.

6.5.6 Querying

The three-tier process is generally involved when transforming a user's query into a search engine query. As seen in section 6.5.6 in the first phase we have a user using a graphical user interface to formulate a query in a search engine, in the second level the query is transformed into tokens, in the last stage the search engine must use the information given by the tokens to look for the relevant information within a database (i.e. index of some sort) to retrieve the desired documents.

Generally we can identify the following types of queries:

- Boolean: AND, OR, NOT words are included in the search in order to include/exclude results.
- Natural Language Queries: the query is formulated as a question or a statement.
- Thesaurus Queries: the user selects the term from a previous term-set provided by the IR system.
- Fuzzy Queries: the threshold of relevance is expanded flexibly to include additional documents.
- Term Searches: this is based on a few words or phrases provided by the user.

- Probabilistic Queries: here the information retrieval system is based on a computed probability to retrieve documents.

Other important issues to consider are the following. These were discussed in section 6.5, describing enterprise search:

- User permissions and access control
- Term weighting for retrieval
- Graphical user interface

6.5.7 System administration

A complete retrieval system should include processes for maintenance, feedback and control to detect atypical behaviour. This can be characterised as an administration system that carries out tasks such as: log file analysis and usage tracking; statistics reports; system monitoring; system diagnosis; system auditing and GUI management.

Figure 6.5 shows the preprocessing, parsing and indexing steps described in this section. Figure 6.6 presents the storing, querying and the system administration processes.

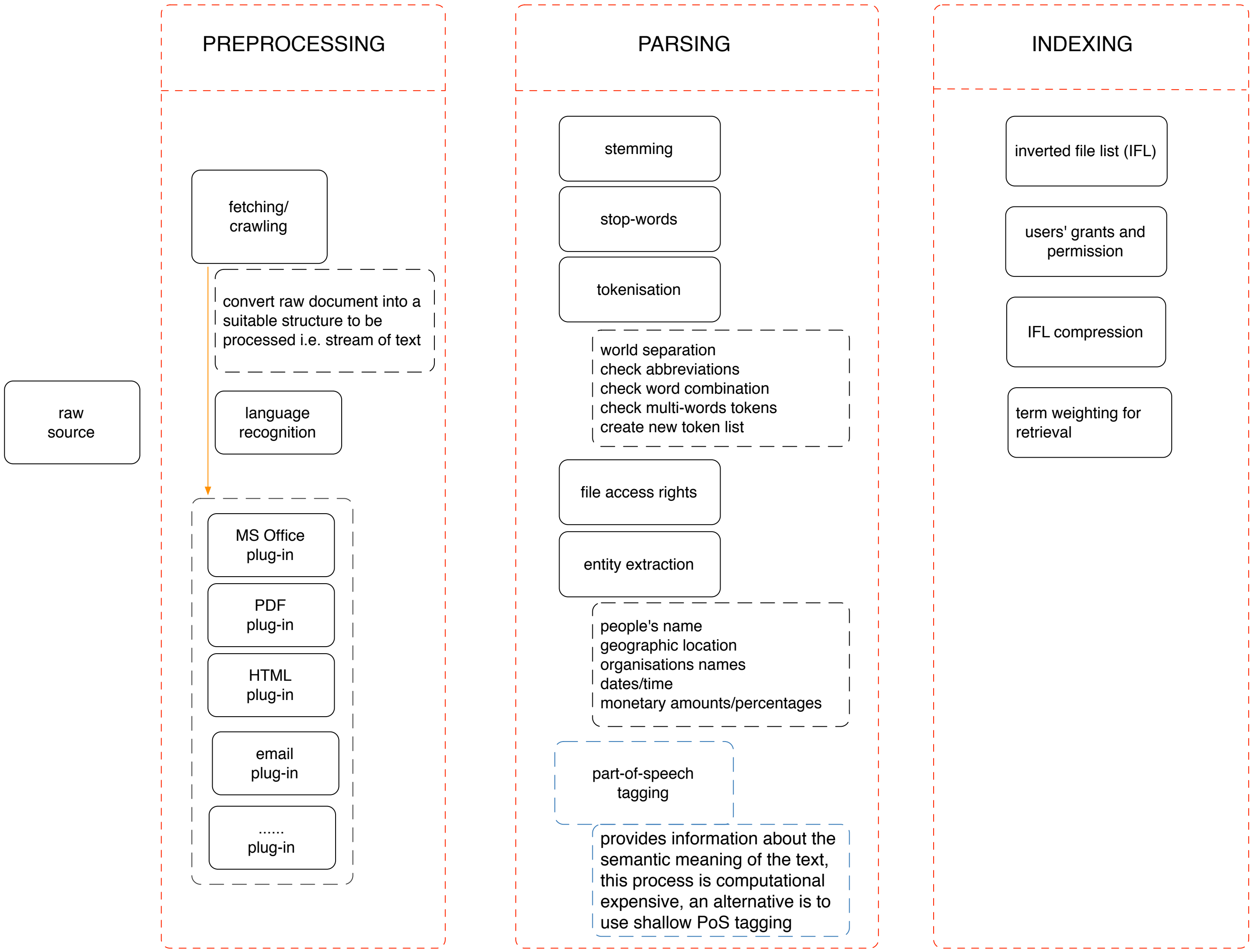


Figure 6.5: *Search engine processes (first part). Three main processes are described here: preprocessing, parsing and indexing.*

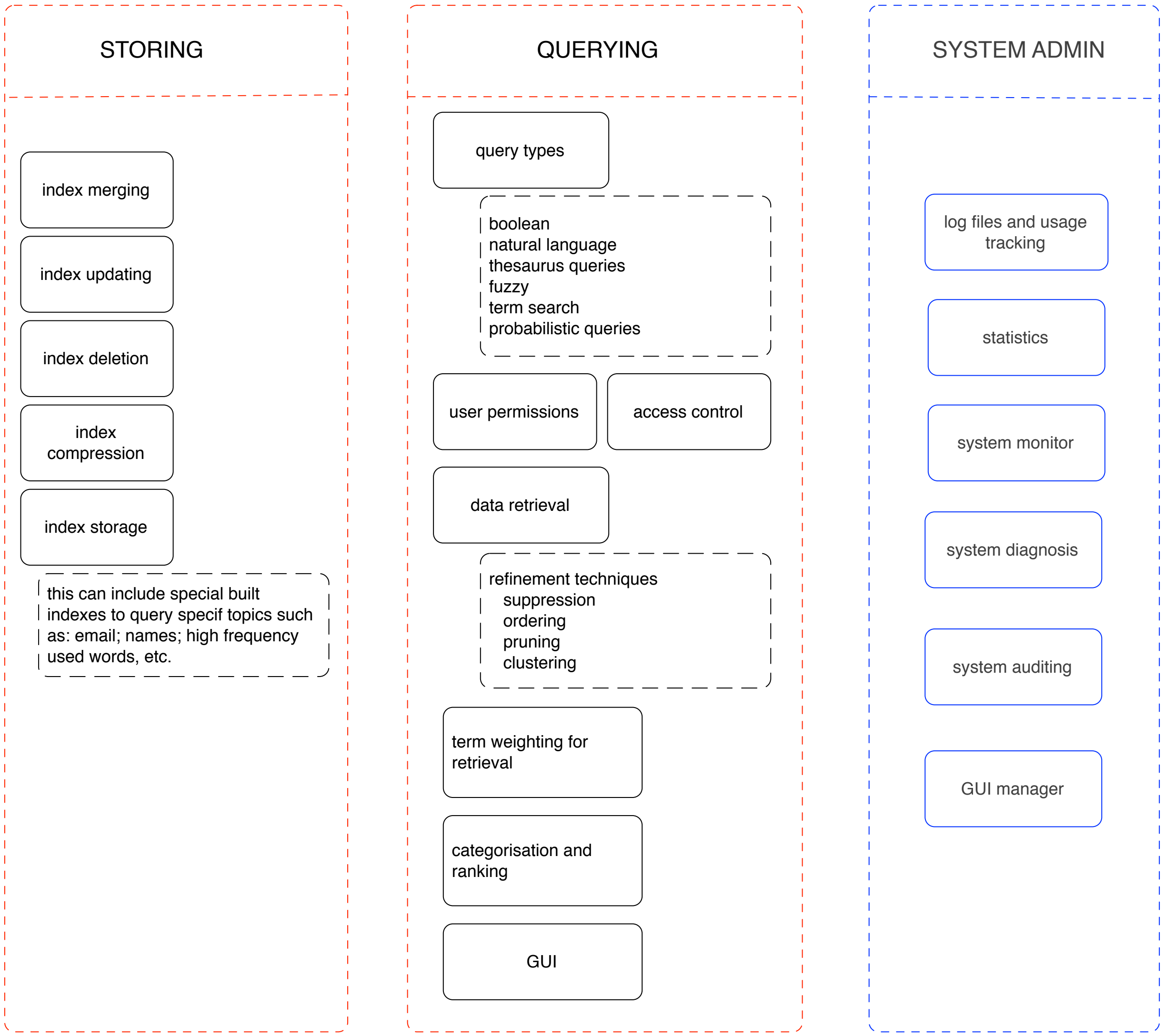


Figure 6.6: *Search engine processes (second part). Three processes are described here: storing, querying and system administration.*

6.5.8 Evaluating enterprise search engines

Building a search solution from scratch is a hard process. Many times a desirable solution is to adapt an already available product. Therefore if an in-house built engine is not an option, the problem becomes how to evaluate and select one of the currently offered technologies.

The following list includes the most relevant selection criteria when comparing existing search technology. If an evaluation of different technologies is needed these criteria can be used for such evaluation. This should be complemented with a measurable scale to quantify each output.

- Ease of use
- Features
- Ability to integrate with other applications
- Cost
- Scalability
- Speed of search application
- Vendor reputation
- Ease of installation
- Upgradability
- Support

6.5.9 Enterprise search providers

In the enterprise search industry there are a great number of solutions based on information retrieval technology. Common characteristics are shared since they aim to build search engines capable of extracting information from different sources and make it readily available to users. They differ in the way the technology is implemented and the methodologies used.

The following list includes a number of relevant commercial tools that have been selected by their importance in the market place.

6.5.10 Commercial solutions

- Attivio, http://www.attivio.com
- Autonomy Corporation, http://www.autonomy.com (also see http://www.ultraseek.com)
- Concept Searching Limited, http://www.conceptsearching.com
- Coveo, http://www.coveo.com
- Dieselpoint, Inc., http://www.dieselpoint.com
- dtSearch Corp., http://www.dtsearch.com
- Endeca Technologies Inc., http://www.endeca.com
- Exalead, http://www.endeca.com
- Expert System S.p.A., http://www.expertsystem.net
- Fast Search and Transfer, http://www.fastsearch.com
- Funnelback, http://funnelback.com
- ISYS Search Software, http://www.isys-search.com
- Northern Light, http://www.northernlight.com/engine.html
- Open Text Corporation, http://www.opentext.com
- Queplix Universal Search Appliance, http://www.queplix.com
- SLI_Systems, http://www.sli-systems.com
- TeraText, http://www.teratext.com
- Ultraseek, http://www.ultraseek.com
- Vivísimo, http://vivisimo.com
- X1 Technologies, Inc., http://www.x1.com
- ZyLAB Technologies, http://www.zylab.com

Companies that provide additional enterprise search technology

- IBM (OmniFind)
- Oracle (Oracle Secure Enterprise Search)
- SAP (Netweaver)
- Microsoft (Fast)
- Google Search Appliance, http://www.google.com/enterprise

An extensive list of currently offered solutions can be found in *Appendix* B.

6.5.11 Open source solutions

The following list includes some of the most widely used open source search engines. Most of them are mainly built to index Web pages with support for MS Office, PDF, RTF and other well established document formats. These engines have accumulated great experience in the information retrieval area. Therefore in case of building an in-house solution they are the starting point regarding document information extraction and index construction.

One of the main drawbacks of open source search engines (from the enterprise point of view) is that they are built around the idea that document indexing is open to everyone. This is not necessarily true in the enterprise search case; access control and compliance are a must.

Particular attention should be given to the licencing of these technologies. Taking code and adapting it to the company's needs saves a great deal of effort, but some technologies will require release of the new code to the open source community. In many cases, specially within the private sector, this is very problematic.

- GATE (http://www.gate.ac.uk): Java-based information extraction software (see http://www.gate.ac.uk/sale/acl02/acl-main.pdf).
- Harvest (http://harvest.sourceforge.net): is a modular, distributed search system framework with a modular architecture to make it a complete search system.
- ht://Dig (http://www.htdig.org): is a long established WWW search engine. With some work it can be modified and used as a search system. Main issues to consider are security and compliance support for different users within an organisation (this is true for most open source search engines).
- Lucene (http://lucene.apache.org): Apache Lucene has a long track record in the open source search community. Lucene is a cross-platform software built on Java but with ports to different languages such as C/C++, C#, .NET, Python.
- Terrier (http://ir.dcs.gla.ac.uk/terrier): Terabyte Retriever is a robust search engine, readily deployable on large-scale collections of documents. Terrier implements state-of-the-art indexing and retrieval functionalities.

- swish-e (http://www.swish-e.org): Simple Web Indexing System for Humans – Enhanced, it uses filters to convert files to XML and HTML for indexing.
- Xapian (http://www.xapian.org): is a highly adaptable toolkit which allows developers to easily add advanced indexing and search facilities to their own applications. It supports the probabilistic information retrieval model and also supports a rich set of boolean query operators.
- Zebra (http://indexdata.dk/zebra): very well supported search engine that can index records in XML, SGML, MARC, e-mail archives and other formats, it uses a combination of boolean searching and relevance ranking. For more information refer to the above address or the following article http://www.open-mag.com/features/Vol_78/zebra/zebra.htm

6.6 Summary

In this chapter we have seen that the search technology under any scenario of application shares many common features, highlighting the particularities of enterprise search. These peculiarities are related to the specific needs in the enterprise for search solutions that address concerns regarding security, compliance, support and many others.

In particular here we are concerned with the main steps involved in an information retrieval system, from the initial step of locating the data source to the final query and answering processes. Of course there are many things not mentioned here, such as parallel processing, query ranking, n-grams, etc.

Current enterprise search technology and its state of the art is examined in this chapter, not only from the commercial viewpoint but also from the viewpoint of the open source community.

Chapter 7

Information Retrieval and the Baire Distance

7.1 Introduction

IN chapter 6 we have seen the processes involved in building a search solutions. Section 6.5 describes the characteristics of the enterprise search. We saw that personalisation is one of a range of requirements. In order to apply the fast Baire-based hierarchical clustering approach, with a personalisation focus, we have selected a test domain that will allow us to carefully validate and evaluate our approach.

In this chapter we go further to show how this can be implemented in two particular contexts. We will approach personalisation in a simple and straightforward way, that points us towards the context of document retrieval. Then in section 7.3, we will move further in the direction of personalisation, working in the context of the cultural heritage domain. This will be useful to us in order to validate the fast Baire distance-based hierarchical approach to structuring data and support retrieval and other operations.

It will also form a bridge to the use of context in the final application to be discussed in this chapter, relating to directory search. All cases discussed in this chapter incorporate context in one way or another. These cases that we deal with were selected in order to address, from the general perspective to an increasingly specific perspective, the needs of enterprise search.

7.2 Document clustering

When searching large numbers of documents it sometimes makes sense to present these documents in the form of clusters ordered by topic, content, or some other criterion. We may find that the concept of a cluster can be a subjective one. For example, Figure 7.1 shows points in a 2D plane. We can ask ourselves, how many clusters are there? six? two? four? Therefore when applying a clustering technique the method or algorithm used and initialisation parameters will affect the output.

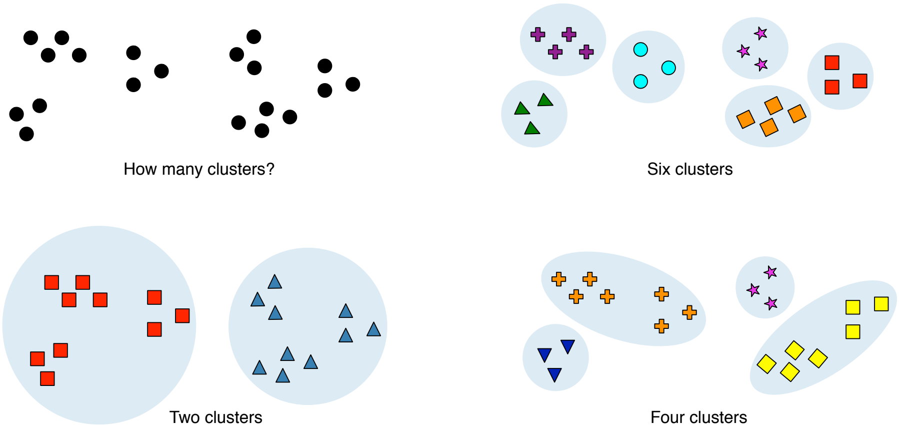


Figure 7.1: *How many clusters? Determining the numbers of clusters can be a difficult question and many times it will be determined by the algorithm's initialisation parameters [164].*

From a machine learning point of view clustering is called *unsupervised learning*. Here we do not have any training data to create a classifier that has learnt to assign a document to a set. In other words, we do not have labels assigned to the documents. In this case no previous knowledge of the number of groups, group size, and the type of document is used

For a large collection of documents a clustering algorithm may create too many clusters in an initial pass. To avoid this problem controlling parameters such as the following can be specified [99]:

- minimum and maximum cluster size
- a matching threshold value for including documents in a cluster
- overlap degree between clusters
- a maximum number of clusters

7.3 Clustering and semantics preservation

In the digital cultural heritage domain, objects (indistinctly, items or artifacts) belong to different contexts (e.g. historical, social, geographical, etc.). Thus, the ultimate purpose of the digital platforms is to help users to discover these contexts, and learn about cultural heritage, through the accessibility and exploration of artifacts, independently from the place, technology or format. In this direction, ontologies have proven to be an extraordinary tool aiding to index and retrieve items recorded in large databases. Nevertheless, it is not unusual that diversity of cultural objects induce potentially big ontologies and/or vocabularies in some domains. One single object may be ontologically described, searched and retrieved by a very small subset of concepts, as compared to the vocabulary's size. In such cases, searching for similar items would suppose processing of huge sparse ($object \times concept$) data structures.

Classical techniques to cluster objects by similarity on these data spaces are not suitable because of their limited effectiveness in handling the available information well [181, 182]. On the contrary, dimensional reduction methods present special opportunities when faced with these structures. In particular, random projection has been shown to hugely improve the computation performance when very large sparse databases are processed [63, 104] while simultaneously preserving characteristics of the original data space. See chapter 2 where we discussed random projection. However, specific complementary methods must be used for clustering purposes, which in turn helps to carry out dataset matching, and to support fast proximity searching.

Massive and high dimensional data spaces often have hidden hierarchical regularity. Following the work in [130], see also chapter 4, we seek ultrametricity in a cultural data set, but also the semantic preservation inside clusters when allowed. An ultrametric is a distance that is defined strictly on a tree. An ultrametric induces a hierarchical structure on data. In previous chapters, chapter 3 and chapter 4, we have compared the Baire ultrametric and the *k*-means algorithm as downstream clustering methods to random projection, finding that the former is faster when grouping objects in the context of chemical structures [130] and astronomy redshifts [31]. Nevertheless, very little is known about the quality of clustering in the context of digital cultural heritage, where semantic preservation inside clusters is relevant.

By *semantic preservation* we mean the original conceptual similarity between two objects in a cluster. Regarding comparison of clustering methods, this is usually focused on evaluation of validity of clusters and algorithmic efficiency. Several validity criteria have been developed in the literature which may be classified as external, internal or relative criteria [69]. In the external approach, groups assembled by a clustering algorithm are compared to a previously accepted partition on the testing data set. In an internal approach, clustering validity is evaluated using data and features contained in the dataset. The relative approach searches for the best clustering result from an algorithm and compares it with a series of predefined clustering schemes. In all cases, validity indexes are constructed to evaluate proximity among objects in a cluster or proximity among resulting clusters.

In our case, prior to clustering, a dimensional reduction is applied. Thus, an interesting question for us is the preservation of original semantic similarity among objects when the dimensional reduction is practiced on the dataset, and a further clustering process is performed on reduced data. In this work, an experiment is designed to test Baire and *k*-means when prior random projection is applied. A data matrix extracted from an ancient folk-music archive containing information about 5000 titles and 9000 inherent characteristics was prepared for the experiment. Because random projection reduces the data matrix into a vector with components in the interval $[0,1[$, as shown in earlier chapters (see chapter 4) different precision levels for clustering purposes are tested.

Next, for each cluster produced by the Baire or *k*-means, semantic similarity of individuals is calculated using the Jaccard index. However, since usually this index measures similarity between two sets (vectors), without consideration of semantic inclusion, we use a modified version. The mean (across different runs of the clustering algorithm) similarity of clusters is calculated in order to compare the clustering methods. Our findings show that semantics are difficult to preserve by these methods, but notwithstanding this, that the Baire approach can be good because it highlights *local* semantic preservation.

In section 7.3.1 the experiment process is presented and transformations applied on datasets are justified. Then section 7.3.4 presents the experiment applied on the cultural dataset and results obtained from the clustering algorithms.

7.3.1 Experiment design

The experiment process used in this work is depicted in Figure 7.2. First, the raw data to be used as input is selected. Usually, data consists of a data matrix where rows represent items in some specific domain, and columns represent characteristics or features associated with these items. In general, we may assume that the dataset is in first normal form in the database sense [43], that is, each cell in the matrix contains just one value. A number of items, and their characteristics, are selected for the next phase.

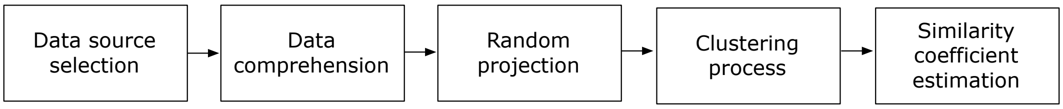


Figure 7.2: *Experiment process design, where five steps are described.*

In the data comprehension step, data completeness, ambiguity and semantic quality of data are evaluated. A pre-process activity is undertaken for analysis purposes. In our case, the main process consists of the identification of semantic data associated with items. Items are annotated following the semantic characteristics, which implies coding the presence or absence of a characteristic for every item in the data structure. The final result in this step is a $\{0,1\}$ matrix where

m items are represented by n attributes. The third step consists of the random projection of the dataset. In this case, a $(n \times 1)$ normalised random vector is used for projection.

A projected dataset is used for clustering purposes in the fourth step. Two clustering algorithms are applied, obtaining a given number of clusters in each case. In the fifth step, intrinsic semantics of clusters, issued from each algorithm, are calculated using a modified Jaccard index. Finally, a hypothesis test is applied to know how significant the difference between the mean similarity of two clustering algorithms is.

7.3.2 On clustering experiments for semantic preservation

In this work, data issued from a digital cultural heritage platform, called Contexta [13], was used to carry out the classification. Contexta aims to integrate and contextualise dispersed, autonomous and heterogeneous cultural information. To achieve this, it uses a middle-ware to integrate distributed data sources with different policies of use, providing uniform access despite multiple data types and formats. Additionally it allows semantic handling of these data and contextualisation based on user and item profiles (e.g. cultural artifact).

One of the main purposes of Contexta when helping users to find cultural items is *situation awareness*. Indeed, in the cultural heritage domain, people using this kind of system are not necessarily interested in single artifacts or lists of ranked items. User needs are inclined to general objectives, searching for elements aiding to compare, interpret, aggregate, analyse, synthesise and discover knowledge [111]. Users receiving recommendations also request explanations in order to support sense-making processes and learning in a contextualised manner. In this setting, this study aims to determine preservation of semantics among items in clusters generated by a specific algorithm, when dimensional reduction is applied.

Let S be the sample data and $s_i \in S, (i = 1, \ldots, n)$, an item in this set. Originally 21 fields are available as item descriptors, but seven characteristics are kept:

uri-identifier, author, content description, title, associated collection, and source. Other attributes are discarded because of redundancy, incompleteness or being uninformative. For semantic purposes a term-based dictionary is created based on the *content description* attribute, allowing an extended search for items. Keywords are produced using the Apache Lucene [109] library, filtering out words (longer than two characters) contained in a stop-list file. Processing the data source we obtain a term-based dictionary with 8674 keywords. The final set of attributes (8680) are considered semantic features. Finally, a matrix is created where the (i, j) element represents the absence or presence (0 or 1) of the jth feature in the ith object. The result is a binary *information matrix* I prepared to carry out the experiments.

7.3.3 Clustering process

Normalisation by column sum is performed in this dataset and the outcome is prepared for the clustering process. In order to assemble clusters from data, random projection is performed first on I. We know that clustering results with the Baire ultrametric are sensitive to the precision level of the projected dataset (see chapters 3 and 4), so seven levels of digit precision are selected for our experiments: two, three, four, five, six, eight and twelve digits of precision. For each precision level, ten random projection vectors are generated, which implies seventy projection outcomes in total. Interestingly, given a precision level, we have found that the resulting projections are practically the same in all cases.

Starting with the Baire methodology, we defined a tree-like structure to store the data's object identifier. Each node has ten children and each value in the projected vector is separated by digits of precision and associated with the tree. The maximum tree depth is given by the digit precision used. For instance, when using four digits of precision we have 10^4 possible branches, and a tree height of four. Once all the data in the projected vector is processed we check each leaf in the tree to identify the clusters. Note that only leaves with more than one element are considered. Thus, for pairwise comparison inside clusters, the 1-item leaves are excluded from the analysis.

Regarding k-means, each experiment uses the number of clusters generated by the Baire method as initialisation parameter. For example, when comparing the Baire clusters produced by four digits of precision, we use the number of clusters with more than two elements as k for k-means (i.e. we exclude the empty and 1-item groups).

Once the clustering process is applied, we evaluate the pairwise semantic similarity within clusters. At this point, similarity between two items may be assessed by the Jaccard index, measuring the number of common items' features, divided by the total number of features present in the respective vectors in I. Given two vector rows $x, y \in I$ (representing the respective items in S), let $F(.)$ be the set of features satisfied by an object in the data description matrix I, i.e. where the matrix value is 1. Then, the Jaccard coefficient of objects x and y in set notation is expressed as:

$$J(x,y) = \frac{|F(x) \cap F(y)|}{|F(x) \cup F(y)|} \tag{7.1}$$

Notice however that this coefficient strongly penalises an item x such that $F(x) \subset F(y)$ and $|F(x)| \ll |F(y)|$.

We propose that two items are semantically identical when the inclusion occurs because they become instances of the same class and/or subclass in an ontological system [150]. In this case, the object y may be considered a specialisation of x. In consequence, a modified Jaccard coefficient is introduced here, which measures the semantic similarity between two objects, and takes into account the inclusion:

$$J_{modif.sim.}(x,y) = \frac{|F(x) \cap F(y)|}{\min\{|F(x)|, |F(y)|\}} \tag{7.2}$$

Note that this similarity is also bounded by 0 and 1.

7.3.4 Results

The process was performed ten times for each precision level and clustering algorithm. On each run, the following measures were calculated over the number of clusters available for analysis:

1. R : average mean similarity among items within clusters;
2. R^*: average maximum similarity among items within clusters;
3. R_*: average minimum similarity among items within clusters.

Given one execution (with a specific random projection vector at a determined precision level), the pairwise similarity among all items inside a cluster is averaged to obtain the mean similarity. Then R is the average of values calculated on clusters generated in such an execution. Equally, R^* and R_* are calculated as the average of maximum and minimum cluster similarity, respectively, among all clusters in a run.

Table 7.1 shows the different average values for every precision level. Also the number of clusters with more than one single element are presented for Baire and *k*-means, where semantic similarity makes sense.

	2 Digit		3 Digit		4 Digit		5 Digits		6 Digits		8 Digits		12 Digits	
	k-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire
R	0.0924	0.0924	0.1105	0.1071	0.1475	0.2302	0.1215	0.3954	0.1197	0.4349	0.1208	0.4398	0.1191	0.4398
R^*	0.6584	0.6584	0.4340	0.3596	0.4074	0.3567	0.4350	0.4806	0.4459	0.5131	0.4487	0.5179	0.4454	0.5178
R_*	0.0027	0.0027	0.0194	0.0311	0.0456	0.1580	0.0248	0.3363	0.0207	0.3792	0.0204	0.3841	0.0216	0.3841
No clusters	99	100	561	740	868	896	602	638	566	594	576	586	558	586
1-item clusters	1	1	60	173	118	1950	30	2923	25	3062	24	3079	23	3079

Table 7.1: *Average semantic similarity for 2, 3, 4, 5, 6, 8 and 12 decimal digits of precision. Where: R, average mean similarity among items within clusters. R^*, average maximum similarity among items within clusters. R_*, average minimum similarity among items within clusters.*

	2 Digit		3 Digit		4 Digit		5 Digits		6 Digits		8 Digits		12 Digits	
	k-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire	*k*-means	Baire
R	0.0037	0.0061	0.0041	0.3463	0.0033	0.0052	0.0029	0.0043	0.0032	0.0016	0.0038	0.0001	0.0037	0.0001
R^*	0.0190	0.0782	0.0098	0.3464	0.0049	0.0037	0.0077	0.0056	0.0088	0.0017	0.0587	0.0000	0.0102	0.0001
R_*	0.0025	0	0.0043	0.3463	0.0035	0.0072	0.0025	0.0042	0.0039	0.0018	0.0032	0.0001	0.0031	0.0001
No clusters	1	0	12	11	8	9	5	11	4	2	4	1	7	1
1-item clusters	1	0	12	8	8	29	5	16	4	5	4	0	7	1

Table 7.2: *Metrics standard deviation. Where: R, average mean similarity among items within clusters. R^*, average maximum similarity among items within clusters. R_*, average minimum similarity among items within clusters.*

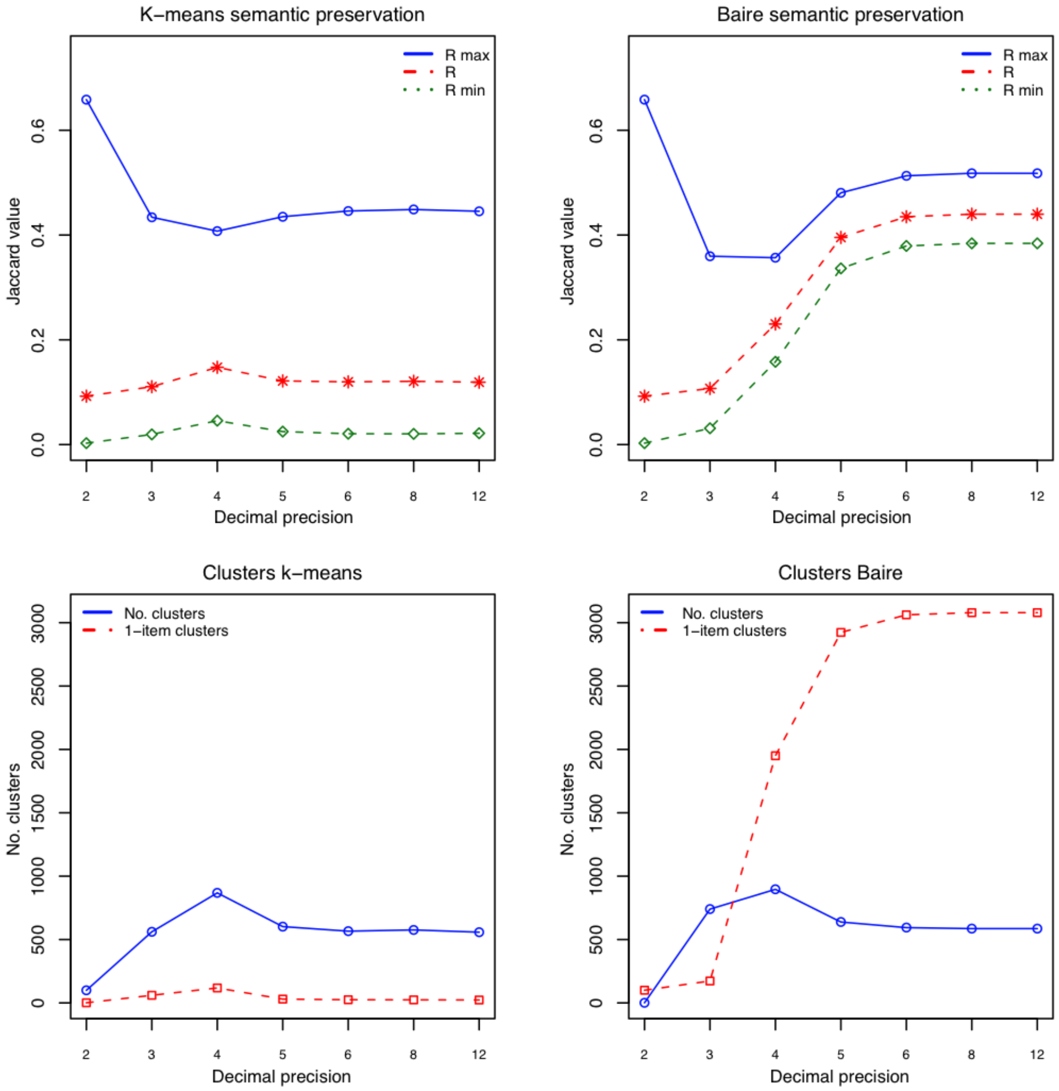


Figure 7.3: ***Top panel***: *Jaccard values. Where: R, average mean similarity among items within clusters. R^*, is average maximum similarity among items within clusters. R_*, is average minimum similarity among items within clusters.* ***Bottom panel***: *Number of clusters for k-means algorithm (bottom left panel) and Baire method (bottom right).*

In order to know whether differences on results obtained for Baire and k-means have a large variation we calculated the standard deviation for each precision level and R, R_*, and R^*. This is explained in Table 7.2, where we note that the standard deviation does not reach 1% of the respective average similarity.

The following results may be observed from Table 7.1 when the precision level increases:

– for the Baire algorithm, R and R_* increase;
– for the k-means algorithm, R and R_* remain almost unchanged;
– Baire is persistently better than k-means in R and R_*;
– the R value is not high for k-means, indicating that semantic similarity is not consistently preserved, but it improves with higher precision in the Baire case.

Additionally these results can be observed graphically in Figure 7.3 where k-means and Baire semantic preservation are presented (top panel). Also in the bottom panel the number of clusters as a function of the digit precision for k-means and the Baire distance are shown.

Despite the fact that semantic similarity is difficult to preserve, we observe that the longer a common prefix within a resulting cluster, the better the semantics of the original data space are preserved. Thus, objects that are semantically closer in the original data matrix I, hashing closer in the 1-dimensional random projected vector, therefore more common prefixes exist among these groups. One could also hypothesise that a heterogeneous original data matrix, in the sense that there is a low semantic similarity among objects, could induce more orphan objects (1-item clusters) in the Baire clustering method. On the other hand, it could be expected that homogeneity implies less orphan elements, but a reasonably good semantic similarity level on the resulting clusters. In fact, the Baire method provides an effective process to detect semantic homogeneity in a matrix. Our experiment indicates that the ratio of the number of orphan objects to the total number of objects, at a high-level precision, could measure how semantically near are rows in I. These considerations will be studied in our future research.

Having looked at new cluster characterisation approaches in domains where we require the preservation of context and of what we can characterise as semantics, we now return to an application to enterprise search.

7.4 On experiments and demonstrator: searching e-mails

In chapter 6 enterprise search has been introduced. In this section we investigate how some of the processes explained there can be implemented. In particular we are interested in e-mail search within the enterprise setting.

The first step before building a prototype was to find a suitable dataset to work with. Unfortunately companies have to be very protective of their data, and thus access to a good dataset was the first problem that we were confronted with. Although there is a good number of research groups dedicated to building data repositories, not any arbitrary data repository is of use to us. A repository containing DNA sequences, economical or astronomical data (to name a few) is of very little use when building a vocabulary-based indexing system (yet a number of similarities can be found regarding methodology).

For this particular application a well suited dataset should contain documents in different formats (e.g. MS Word, PDF, and text) and multi-purpose built documents (e.g. email, manuals, articles). Therefore, a good dataset is one that can emulate the kind of document that companies use in a day-to-day business.

Table 7.3 shows the W3C TREC Enterprise Track [162] corpus. This is from a crawl of public W3C information that comprises 331,037 documents, mainly e-mails. Thus, in order to focus our efforts the e-mail subset was taken from the W3C Enterprise TREC, and a parser was built to extract relevant information that later was automatically inserted into a database.

Type	Scope	Size (GB)	Documents	Avg. docsize (kb)
E-mail	lists	1.855	198,394	9.8
Code	dev	2.578	62,509	43.2
Web	www	1.043	45,975	23.8
Wiki web	esw	0.181	19,605	9.7
Misc	other	0.074	3,538	14.1
Web	people	0.003	1,016	3.6
	all	5.7	331,037	18.1

Table 7.3: *W3C TREC Enterprise Corpus data source description. Here we are interested in the e-mail set.*

Following the design previously shown in chapter 6 the prototype main components include the following:

– Parser design and construction: this is currently based on a set of C++ programs that extract tokens from text files, also it is able to filter stop words and dump the result to a file.

– Metadata structure and fields: this was implemented through a database model able to store e-mails and free text.

– Metadata storage system: MySQL was used to implement the current demonstrator. Also tests with Berkeley Database (BDB) [138] were carried out. An inverted list stores relevant tokens with references to documents.

– Metadata search and matching: a number of Java programs were created to be used as interface between the database and the GUI.

– Web user interface: Java and Tomcat were used to create a GUI Servlet as interface to the database.

Figure 7.4 shows a demonstrator screen-shot using Apache Tomcat [166] as Servlet container in order to produce the GUI. Since we were working with e-mail data the following tailored functions were implemented. This demonstrator can be accessed at [32].

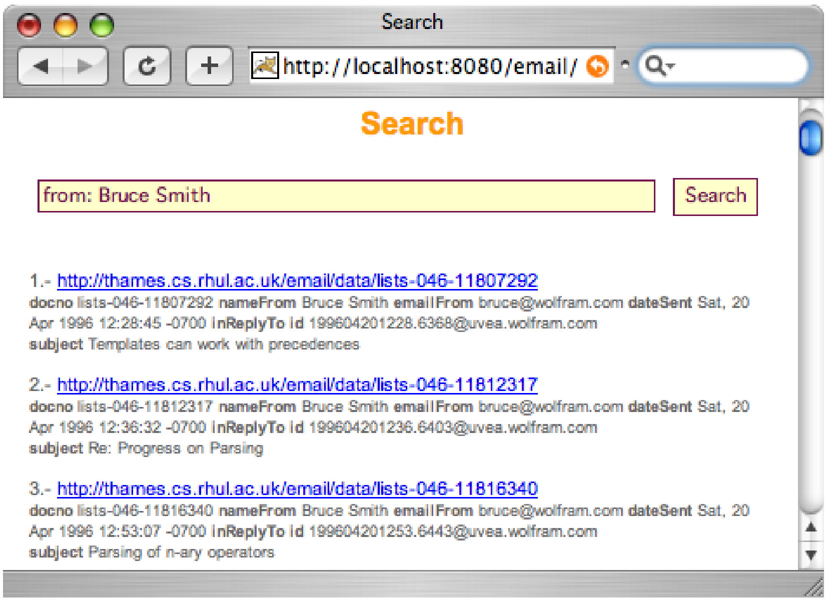
– `file:`*`keyword`* retrieves e-mails by file name

– `from:`*`name`* retrieves e-mails by sender

– `to:`*`email to`* retrieves e-mails by recipient

– `cc:`*`email cc`* retrieves e-mails by cc recipient

- `subject:`*`subject`* retrieves e-mails by subject
- `keyword:` retrieves keywords within the e-mail body



from: Bruce Smith

1.- http://thames.cs.rhul.ac.uk/email/data/lists-046-11807292
docno lists-046-11807292 **nameFrom** Bruce Smith **emailFrom** bruce@wolfram.com **dateSent** Sat, 20 Apr 1996 12:28:45 -0700 **inReplyTo id** 199604201228.6368@uvea.wolfram.com
subject Templates can work with precedences

2.- http://thames.cs.rhul.ac.uk/email/data/lists-046-11812317
docno lists-046-11812317 **nameFrom** Bruce Smith **emailFrom** bruce@wolfram.com **dateSent** Sat, 20 Apr 1996 12:36:32 -0700 **inReplyTo id** 199604201236.6403@uvea.wolfram.com
subject Re: Progress on Parsing

3.- http://thames.cs.rhul.ac.uk/email/data/lists-046-11816340
docno lists-046-11816340 **nameFrom** Bruce Smith **emailFrom** bruce@wolfram.com **dateSent** Sat, 20 Apr 1996 12:53:07 -0700 **inReplyTo id** 199604201253.6443@uvea.wolfram.com
subject Parsing of n-ary operators

Figure 7.4: *E-mail enterprise TREC search application.*

In this demonstrator MySQL [133] is used for storing, exact match and retrieval. This also can be seen as common prefix exact matching.

7.5 Summary

We have analysed semantic similarity among objects when dimensional reduction is applied on a dataset and a further clustering process is performed on dimensionally reduced data.

An experiment was designed to test Baire, or longest common prefix ultrametric, and k-means when prior random projection is applied. A data matrix extracted from an ancient folk-music archive was prepared for the experiment. Different precision levels for clustering purposes were tested and semantic similarity among grouped elements was calculated using a modified version of the

Jaccard index. Our findings show that semantics are difficult to preserve by these methods, because the calculated similarity coefficient achieves moderate values only. On the one hand we can say that the Baire method in this case works as a filtering method for vectors that are semantically similar. On the other hand, once the number of centroids is chosen, *k*-means works by pulling the data points towards the centroids without considering if these points are justifiably close in the original data space.

It was observed that both methods produce an important number of 1-item groups. Nevertheless, our results show that taking advantage of inherent data ultrametricity is a possible strategy for detecting semantic homogeneity in the original dataset. Therefore, our future research will consider alternative data sources, always in the area of semantic analysis, to further exploit this.

Furthermore, an e-mail demonstrator was implemented as proof of concept for an enterprise application, where exact matching was performed. This shows that the longest common prefix can play a very central role in this important field.

Chapter 8
Conclusions

IN this thesis we are particularly concerned with exploratory data analysis. We introduce a novel (ultra)metric distance for clustering. We validate this method by comparing it with long established cluster algorithms such as *k*-means. Moreover, we use real life examples and applications to different science areas as well as information retrieval to exemplify its use.

The proposed method presents a number of advantages when compared with more traditional techniques. When working with numeric data, distances can be interpreted directly and classification carried out. Furthermore, the resulting distances can be fitted strictly to a hierarchical tree. This is very important for classification, storing and fast search.

The datasets used for experiment were the following:

- *Astronomy*: 443,014 objects in 2 dimensions.
- *Chemoinformatics*: 1.2 million compounds in 1,052 dimensions.
- *Bioinformatics*: 4,889 proteins from which 2,542 are hypothetical, where the minimal protein length is of 33 amino acids and the longest 8,161 amino acids, based on an alphabet of 20 letters.
- *Digital heritage*: 5,000 music titles in 8,674 dimensions.
- *Enterprise search*: 198,394 e-mail documents.

A list of the different software systems used through this work includes the following:

- *Programming languages*: Java and C/C++.
- *Statistical analysis*: R and IDL.
- *Ploting*: GNUPLOT and R.
- *Web and search technologies*: Apache Web Server, Apache Tomcat and Apache Lucene
- *Databases*: MySQL and Berkeley Database (BDB).
- *Other*: CLUSTAL, for multiple protein alignment.

Visualisation has also played an important role within this work. This is particularly relevant in the area of exploratory data analysis. Many times patterns are better understood when plotted and trends can be much easier to identify. Thus a great deal of effort was made to present visual data in such a way that can be simple to understand.

8.1 Final remarks and possible extensions

The work presented in this thesis can be carried forward in many directions. The first logical extension is to continue the application of the Baire distance to more data sources, with particular focus on high dimensional spaces. There are many things to discover about the topology and properties of high dimensional spaces, where we expect that ultrametric topology will play an important role.

Other research topics include the following:

> **Baire with sliding window**: on decimal base numerical data the Baire clusters are created, and hence the data is partitioned at the finest precision level, from one number to the next. When obtaining the longest common prefix, a sliding window can be introduced in order to make flexible this data partitioning criterion. For example, in the case of the following two numbers, 0.121 and 0.122, we have that the longest common prefix is $K = 2$, i.e. based on decimal position two. A new criterion

could be that after finding the longest common prefix, we look for the next decimal digit and if this is close enough it can be used to consider the two associated numbers to be part of the same cluster. Following on from our example, we can look at $K = 3$ and if the numbers are close enough we take them as being in the same group. In this case (i.e. for the two numbers 0.121 and 0.122) 1 is close to 2, so with the new definition we can say now that $k = 3$ is the distance between these two numbers. Such a criterion for cluster formation can be used to allow overlapping clusters.

Self indexing ultrametric for retrieval: the longest common prefix has many applications in string matching. Some of most used data structures for string indexing are suffix trees, which are based on the longest common prefix. Also this topic is very closely related to certain compression algorithms. It would be very interesting to explore if an ultrametric embedding could help in improving the use and performance of these data structures.

Appendix A

Baire Algorithm Implementation in Java

In this appendix we present three Java methods that implement useful algorithms. Thus, how to calculate the longest common prefix is shown in A.1. How to obtain the Baire node index from a string is presented in A.2. Finally, how to calculate the Baire distance from given string a and integer n is listed in A.3.

In A.1 two real numbers are read as strings and compared character by character. This is in order to determine how many common prefixes these two strings have. In this particular case the "." and "–" characters are not considered. Therefore they are ignored in the total sum of characters if they appear.

Note that this implementation deals with decimal numbers and scientific notation. For this reason there is a parsing step before obtaining the final string to analyse (see lines 4 and 5).

```

public int commonPrefix(String a, String b) {

    String x = new BigDecimal(Double.toString(Double.parseDouble(a))).
        toPlainString();
    String y = new BigDecimal(Double.toString(Double.parseDouble(b))).
        toPlainString();

    int commonPrefix = 0;

    //retrieve the length of the shortest of two string
    int length = Math.min(x.length(), y.length());

    for (int i = 0; i < length; i++) {
        if ((x.charAt(i) == y.charAt(i))) {
            commonPrefix++;
            if ((x.charAt(i) == '.') || (y.charAt(i) == '-')) {
                commonPrefix = (commonPrefix - 1);
            }
        } else {
            break;
        }
    }
    return commonPrefix;
}
```

Listing A.1: *Baire distance or longest common prefix of two strings.*

In A.2 we retrieve the digit index based on the Baire distance. For example, if we have a $d_{\mathcal{B}} = 3$ for the string "0.015746", we retrieve 001. This is very useful in case we decide to store the data points in a tree, because the digit index will be the leaf of a tree to which a data point belongs.

```java

public String clusterIndex(String a, int distance) {
    String x = new BigDecimal(Double.toString(Double.parseDouble(a))).
        toPlainString();
    String y = '';

    for (int i = 0; i < x.length(); i++) {

        if ((x.charAt(i) == '.') || (x.charAt(i) == '-')) {
        } else {
            y =  y.concat(Character.toString(x.charAt(i)));
        }
    }
    String clusterIndex = y.substring(0, distance);
    return clusterIndex;
}
```

Listing A.2: *Baire index of a string.*

In A.3 the Baire distance is calculated for a given n. Recall the non-trivial part of the Baire distance formula:

$$d_{\mathcal{B}}(x_K, y_K) = inf \;\; 2^{-n} \quad x_n = y_n \quad 1 \leq n \leq |K|$$

```java

public double distance(int n) {
    double baireDistance = 0;
    baireDistance = Math.pow(2, -n);

    return baireDistance;
}
```

Listing A.3: *Calculates the Baire distance from an integer provided by the longest common prefix.*

Appendix B

Enterprise Search Providers

This is a extensive list of companies providing enterprise search technology in 2009. As noted in chapter 6, enterprise search has a number of characteristics that make it different from traditional search engines. For example, special consideration is needed to support effectively and efficiently permissions and file authorisations.

Abrevity: http://www.abrevity.com
Access Innovations: http://www.accessinn.com
AltaVista: http://www.altavista.com see also: Fast
ATG: http://www.atg.com/en/products/commerce_search.jhtml
Attensity: http://www.attensity.com
Attivio: http://www.attivio.com
Autonomy: http://www.autonomy.com/content/home/index.en.html
BA-Insight: http://www.ba-insight.net/products.html
Basis Technology: http://www.basistech.com
Baynote: http://www.baynote.com
Blossom Software: http://www.blossom.com/index.html
Brainware: http://www.brainware.com
BRS/Search: see: OpenText
Business Objects: http://www.businessobjects.com/product/information_discovery, see also: SAP
Clarabridge: http://www.clarabridge.com
Clusty: see: Vivisimo
COGITO: see: Expert System
Collanos: http://www.collanos.com
Collarity: http://www.collarity.com
Concept Searching: http://www.conceptsearching.com/web
Connotate: http://www.connotate.com
Convera: http://www.convera.com
Coveo: http://www.coveo.com/en/default.aspx
Cuadra Associates: http://www.cuadra.com
Data Harmony: see: Access Innovations

Deki Wiki: see: MindTouch
Dieselpoint: http://www.dieselpoint.com/featurematrix.html
Documentum : see: EMC
dtSearch: http://www.dtsearch.com
EasyAsk: see: Progress Software
EMC: http://www.emc.com/products/detail/software/eci-services.htm
Endeca: http://endeca.com
Engenium: see: Kroll Ontrack
Exalead: http://corporate.exalead.com/enterprise/l=en
Excalibur: see: Convera
Expert System: http://www.expertsystem.net/?lang=1
Eyealike: http://www.eyealike.com/index.php
Fast ESP: see: Fast Search Transfer
Fast Search Transfer: http://www.fastsearch.com
Funnelback: http://funnelback.com
Google: http://www.google.com/enterprise/intranet_search.html
Grokker: http://www.grokker.com
Hummingbird Search Server: see: OpenText
IBM: http://www-306.ibm.com/software/data/enterprise-search
IDOL: see: Autonomy
Image-seeker: see: LTU Technologies
Index Engines: http://www.indexengines.com
Information Builders: http://www.informationbuilders.com/products/webfocus/index.html
Inmagic: http://www.inmagic.com/index.html
InQuira: http://www.inquira.com
Instranet: http://www.instranet.com/index.asp
IntelliSearch: http://www.intellisearch.no/Solutions
InXight: see: Business Objects
iPhrase: see: IBM
ISYS: http://www.isys-search.com
IXIASOFT: http://www.ixiasoft.com
K2 Enterprise: see: Autonomy
Kaidara: http://www.kaidara.com
Knova: http://www.knova.com
Kroll Ontrack: http://www.engeniumsearch.com
Legato: see: EMC
Lexalytics: http://www.lexalytics.com/index.php
Liberty IMS: http://www.libertyims.com/index.html
LiveLink: see: OpenText
Longitude: see: BA-Insight
LTU Technologies: http://www.ltutech.com/en
Lucene: http://lucene.apache.org/java/docs
Luxid: see: Temis
MarkLogic: http://www.marklogic.com
Mercado: http://www.mercado.com
Microsoft: http://www.microsoft.com/enterprisesearch

MindServer: see: Recommind
MindTouch: http://wiki.mindtouch.com
MondoSoft: see: SurfRay
MultiTes: http://www.multites.com
MuseGlobal: http://www.museglobal.com
Nervana: http://www.nervana.com
NetWeaver: see: SAP
NorthernLight: http://www.northernlight.com
nStein: http://www.nstein.com
Olive Software: http://www.olivesoftware.com
OmniFind: see: IBM
Ontolica: see: SurfRay
Ontrack Engenium : see: Kroll Ontrack
OpenText: http://www.opentext.com
Oracle: http://www.oracle.com/database/secure-enterprise-search.html
Paglo: http://paglo.com
PicoSearch: http://www.picosearch.com
Polyspot: http://www.polyspot.com/Home.aspx
Powerset: http://www.powerset.com/about
Presto: see: Inmagic
Progress: Software http://www.progress.com/index.ssp
QL2: http://www.ql2.com
Recommind: http://www.recommind.com
Rosette Linguistics Platform: see: Basis Technology
SAIC: http://www.saic.com/products/software/teratext/products
SAP: http://www.sap.com/usa/solutions/informationworkers/enterprisesearch/index.epx
SAS: http://www.sas.com/technologies/analytics/datamining
Schemalogic: http://www.schemalogic.com
Seaglex: http://www.seaglex.com
SearchBlox: http://www.searchblox.com
Semantra: http://www.semantra.com
Siderean: http://www.siderean.com
Silobreaker: http://www.silobreaker.com
Sinequa: http://www.sinequa.com/index.html
SLI Systems: http://www.sli-systems.com
STAR: see: Cuadra Associates
SurfRay: http://www.surfray.com
tazti: see: VoiceTech Group
Techrigy: http://www.techrigy.com
Temis: http://www.temis.com
Terabase: http://www.terabase.com
Teragram: see: SAS
TeraText: see: SAIC
Texis: see: Thunderstone
TEXTML: see: IXIASOFT
TextWorks: see: Inmagic

Thunderstone: http://www.thunderstone.com/texis/site/pages
TREX: see: SAP
Ultraseek: see: Autonomy
Velocity: see: Vivisimo
Verity: see: Autonomy
Vivisimo: http://vivisimo.com
VoiceTech: Group http://www.voicetechgroup.com
Vorsite: http://www.vorsite.com/Default.aspx
WAND: http://www.wand.com/core/AboutUs.aspx
WebFOCUS: see: Information Builders
WebSphere: see: IBM
Wikia: http://search.wikia.com/wiki/Search_Wikia
WordMap: http://www.wordmap.com
X1: http://www.wordmap.com
Xerox: PARC see: Powerset
ZyLAB: http://www.zylab.com

Appendix C

Additional Resources Available from This Thesis

A web area has been set up at the following address: http://thames.cs.rhul.ac.uk/~pedro/dissertation.html. Many resources related to this dissertation are present there, including the following.

Source codes in C/C++, Java and R scripts, including the following:

- C code for parsing and removing stop-list words, from the W3C TREC Enterprise Corpus.
- Java code to export e-mail data to MySQL database.
- Java code for working with the Baire ultrametric.
- R code for analysing data and plotting.
- GnuPlot code for plotting.

Animated 3D plots, including the following:

- R 3D animated plot.
- GnuPlot 3D animated plot.

Data sets:

- SDSS FITS format data set.
- SDSS preprocessed data set.
- SDSS digit distribution data.
- Contexta semantically arranged data set.
- Contexta semantically arranged random projected data.
- Contexta digit distribution data from the random projected vectors.

Bibliography

[1] D. Achlioptas, "Database-friendly random projections," in *PODS '01: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. New York, NY, USA: ACM, 2001, pp. 274–281.

[2] J. K. Adelman-McCarthy, M. A. Agüeros, S. S. Allam, K. S. J. Anderson, S. F. Anderson, J. Annis, N. A. Bahcall, C. A. L. Bailer-Jones, I. K. Baldry, J. C. Barentine, T. C. Beers, V. Belokurov, A. Berlind, M. Bernardi, M. R. Blanton, J. J. Bochanski, W. N. Boroski, D. M. Bramich, H. J. Brewington, J. Brinchmann, J. Brinkmann, R. J. Brunner, T. Budavári, L. N. Carey, S. Carliles, M. A. Carr, F. J. Castander, A. J. Connolly, R. J. Cool, C. E. Cunha, I. Csabai, J. J. Dalcanton, M. Doi, D. J. Eisenstein, M. L. Evans, N. W. Evans, X. Fan, D. P. Finkbeiner, S. D. Friedman, J. A. Frieman, M. Fukugita, B. Gillespie, G. Gilmore, K. Glazebrook, J. Gray, E. K. Grebel, J. E. Gunn, E. de Haas, P. B. Hall, M. Harvanek, S. L. Hawley, J. Hayes, T. M. Heckman, J. S. Hendry, G. S. Hennessy, R. B. Hindsley, C. M. Hirata, C. J. Hogan, D. W. Hogg, J. A. Holtzman, S. ichi Ichikawa, T. Ichikawa, Željko Ivezić, S. Jester, D. E. Johnston, A. M. Jorgensen, M. Jurić, G. Kauffmann, S. M. Kent, S. J. Kleinman, G. R. Knapp, A. Y. Kniazev, R. G. Kron, J. Krzesinski, N. Kuropatkin, D. Q. Lamb, H. Lampeitl, B. C. Lee, R. F. Leger, M. Lima, H. Lin, D. C. Long, J. Loveday, R. H. Lupton, R. Mandelbaum, B. Margon, D. Martínez-Delgado, T. Matsubara, P. M. McGehee, T. A. McKay, A. Meiksin, J. A. Munn, R. Nakajima, T. Nash, E. H. Neilsen, Jr., H. J. Newberg, R. C. Nichol, M. Nieto-Santisteban, A. Nitta, H. Oyaizu, S. Okamura, J. P. Ostriker, N. Padmanabhan, C. Park, J. Peoples, Jr., J. R. Pier, A. C. Pope, D. Pourbaix, T. R. Quinn, M. J. Raddick, P. R. Fiorentin, G. T. Richards, M. W. Richmond, H.-W. Rix, C. M. Rockosi, D. J. Schlegel, D. P. Schneider, R. Scranton, U. Seljak, E. Sheldon, K. Shimasaku, N. M. Silvestri, J. A. Smith, V. Smolčić, S. A. Snedden, A. Stebbins, C. Stoughton, M. A. Strauss, M. SubbaRao, Y. Suto, A. S. Szalay, I. Szapudi, P. Szkody, M. Tegmark, A. R. Thakar, C. A. Tremonti, D. L. Tucker, A. Uomoto, D. E. V. Berk, J. Vandenberg, S. Vidrih, M. S. Vogeley, W. Voges, N. P. Vogt, D. H. Weinberg, A. A. West, S. D. M. White, B. Wilhite, B. Yanny, D. R. Yocum, D. G. York, I. Zehavi, S. Zibetti, and D. B. Zucker, "The fifth data

release of the Sloan Digital Sky Survey," *The Astrophysical Journal Supplement Series*, vol. 172, no. 2, pp. 634–644, 2007.

[3] A. N. Albatineh, M. Niewiadomska-Bugaj, and D. Mihalko, "On similarity indices and correction for chance agreement," *Journal of Classification*, vol. 23, no. 2, pp. 301–313, 2006.

[4] A. Ali and D. Hill, "Giardia intestinalis," *Current opinion in infectious diseases*, vol. 16, no. 5, pp. 453–460, 2003.

[5] M. R. Anderberg, *Cluster Analysis for Applications*. New York: Academic Press, 1973.

[6] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006, ch. Locally-Sensitive Hashing Using Stable Distributions, pp. 55–65.

[7] Arecibo, "Arecibo Observatory," 2005, http://www.naic.edu.

[8] AstroWeb, "Astronomical survey projects," The AstroWeb Database of Internet Resources, 2010, http://cdsweb.u-strasbg.fr/astroweb/survey.html.

[9] ——, "The AstroWeb database of internet resources," 2010, http://cdsweb.u-strasbg.fr/astroWeb/astroweb.html.

[10] ——, "Ground-based telescopes and observatories," The AstroWeb Database of Internet Resources, 2010, http://cdsweb.u-strasbg.fr/astroweb/telnotspace.html.

[11] ——, "Observatories and telescopes," The AstroWeb Database of Internet Resources, 2010, http://cdsweb.u-strasbg.fr/astroweb/telescope.html.

[12] ——, "Space telescopes and observatories," The AstroWeb Database of Internet Resources, 2010, http://cdsweb.u-strasbg.fr/astroweb/telspace.html.

[13] H. Astudillo and et al, "Contexta: Semantic and contextualized management of distributed heterogeneous collections," 2007, fONDEF Chilean Grant D05I10286.

[14] K. Balog, "People search in the enterprise," Ph.D. dissertation, ISLA, University of Amsterdam, 2008, http://staff.science.uva.nl/~kbalog/phd-thesis.

[15] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton: Princeton University Press, 1961.

[16] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data Recent Advances in Clustering*. Springer, 2006, pp. 25–71.

[17] M. W. Berry, *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer-Verlag, New York, 2004.

[18] M. W. Berry and M. Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*, 2nd ed. SIAM, Society for Industrial and Applied Mathematics, 2005.

[19] A. Biegert, C. Mayer, M. Remmert, J. Soding, and A. N. Lupas, "The MPI bioinformatics toolkit for protein sequence analysis," *Nucleic Acids Research*, vol. 34, pp. W335–W339, 2006.

[20] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2001, pp. 245–250.

[21] R. Blashfield and M. Aldenderfer, "The literature on cluster analysis," *Multivariate Behavioral Research*, vol. 13, no. 3, pp. 271–295, 1978.

[22] N. Bolshakova and F. Azuaje, "Cluster validation techniques for genome expression data," *Signal Processing*, vol. 83, no. 4, pp. 825–833, 2003.

[23] N. Brown, "Chemoinformatics—an introduction for computer scientists," *ACM Computing Surveys*, vol. 41, no. 2, pp. 1–38, 2009.

[24] R. J. Brunner, S. G. Djorgovski, T. A. Prince, and A. S. Szalay, *Handbook of Massive Data Sets*. Norwell, MA, USA: Kluwer Academic Publishers, 2002, ch. Massive datasets in astronomy, pp. 931–979.

[25] D. Carmel, D. Cohen, E. Farchi, M. Herscovici, Y. S. Maarek, and A. Soffer, "Static index pruning for information retrieval systems," in *24th ACM SIGIR Conference on Research and Development in Information Retrieval*. New Orleans, Louisiana, USA: ACM, 2001, pp. 43–50.

[26] H.-L. Chan, W.-K. Hon, T.-W. Lam, and K. Sadakane, "Compressed indexes for dynamic text collections," *ACM Transactions on Algorithms*, vol. 3, no. 2, May 2007.

[27] Chandra, "Chandra X-ray Observatory," 2007, http://chandra.harvard.edu.

[28] J.-W. Chang and D.-S. Jin, "A new cell-based clustering method for large, high-dimensional data in data mining applications," in *SAC '02: Proceedings*

of the 2002 ACM symposium on Applied computing. New York, NY, USA: ACM, 2002, pp. 503–507.

[29] C. Charras and T. Lecroq, *Handbook of Exact String Matching Algorithms*. King's College Publications, 2004.

[30] J. Cohen, "Bioinformatics – an introduction for computers scientists," *ACM Computing Surveys*, vol. 36, no. 2, pp. 122–158, 2004.

[31] P. Contreras and F. Murtagh, "Fast hierarchical clustering from the Baire distance," in *Classification as a Tool for Research. Proceedings of the 11th IFCS Biennial Conference and 33rd Annual Conference of the Gesellschaft für Klassifikation*. Dresden: Springer, 2009, in press.

[32] P. Contreras, "E-mail Search Demonstrator," January 2007, http://thames.cs.rhul.ac.uk:8080/email/Search.

[33] P. Contreras and F. Murtagh, "Evaluation of hierarchies based on the longest common prefix, or Baire, metric," 2007, Classification Society of North America (CSNA) meeting, University of Illinois. Urbana-Champaign. IL, USA.

[34] R. M. Cormack, "A review of classification," *Journal of the Royal Statistical Society. Series A (General)*, vol. 134, no. 3, pp. 321–367, 1971.

[35] M. Crochemoree, C. Hancart, and T. Lecroq, *Algorithms on Strings*. Cambridge University Press, 2007.

[36] R. D'Abrusco, G. Longo, M. Paolillo, M. Brescia, E. D. Filippi, A. Staiano, and R. Tagliaferri, "The use of neural networks to probe the structure of the nearby universe," April 2007, http://arxiv.org/abs/astro-ph/0701137.

[37] R. D'Abrusco, A. Staiano, G. Longo, M. Brescia, M. Paolillo, E. D. Filippis, and R. Tagliaferri, "Mining the SDSS archive. I. Photometric redshifts in the nearby universe," *Astrophysical Journal*, vol. 663, no. 2, pp. 752–764, July 2007.

[38] R. D'Abrusco, A. Staiano, G. Longo, M. Paolillo, and E. D. Filippis, "Steps toward a classifier for the virtual observatory. I. Classifying the SDSS photometric archive," 1st Workshop of Astronomy and Astrophysics for Students-Naples, April 2006, http://arxiv.org/abs/0706.4424.

[39] S. Dasgupta, "Experiments with random projection," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 143–151.

[40] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[41] M. Dash, H. Liu, P. Scheuermann, and K. L. Tan, "Fast hierarchical clustering and its validation," *Data and Knowledge Engineering.*, vol. 44, no. 1, pp. 109–138, 2003.

[42] M. Dash, H. Liu, and X. Xu, "'$1 + 1 > 2$': Merging distance and density based clustering," in *DASFAA '01: Proceedings of the 7th International Conference on Database Systems for Advanced Applications*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 32–39.

[43] C. J. Date, *An Introduction to Database Systems*, 8th ed. Addison-Wesley, 2003.

[44] S. Deegalla and H. Boström, "Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification," in *ICMLA '06: Proceedings of the 5th International Conference on Machine Learning and Applications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 245–250.

[45] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer-Verlag Berlin, 2009.

[46] B. Dirking, *Enterprise Search Source Book*. Information Today, Inc., 2008, http://www.nxtbook.com/nxtbooks/infotoday/enterprisesearchsourcebook08.

[47] I. Dondoshansky and Y. Wolf, "BLASTClust, BLAST score-based single-linkage clustering," 2010, http://www.csc.fi/english/research/sciences/bioscience/programs/blast/blastclust.

[48] ——, "BLASTClust, Clustering Non-redundant Sequence Sets," 2010, http://www.ncbi.nlm.nih.gov/Web/Newsltr/Spring04/blastlab.html.

[49] G. M. Downs and J. M. Barnard, *Reviews in Computational Chemistry, Volume 18*. John Wiley & Sons, Inc, 2003, ch. Clustering Methods and Their Uses in Computational Chemistry.

[50] B. Dragovich and A. Dragovich, "p-Adic modelling of the genome and the genetic code," *The Computer Journal*, vol. 53, no. 4, pp. 432–442, 2010.

[51] S. T. Dumais, "Improving the retrieval of information from external sources," in *Behavior Research Methods, Instruments, & Computers*, vol. 23, 1991, pp. 229–236.

[52] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, "Reoptimization of MDL keys for use in drug discovery," *Journal of Chemical Information and Computer Sciences*, vol. 42, no. 6, pp. 1273–1280, 2002.

[53] S. R. Eddy, "Where did the BLOSUM62 alignment score matrix come from?" *Nature Biotechnology*, vol. 22, no. 4, pp. 1035–1036, 2004.

[54] B. Erevitt, *Cluster Analysis*. John Wiley & Sons Inc, 1993.

[55] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *2nd International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[56] M. Farthing, "Giardia intestinalis," *Gastroenterology clinics of North America*, vol. 25, no. 3, 1996.

[57] X. Z. Fern and C. Brodly, "Random projection for high dimensional data clustering: A cluster ensemble approach," in *Proceedings of the Twentieth International Conference on Machine Learning*. Washington, DC, USA: AAAI Press, 2007.

[58] A. Fernández-Soto, K. M. Lanzetta, H.-W. Chen, S. M. Pascarelle, and N. Yahata, "On the compared accuracy and reliability of spectroscopic and photometric redshift measurements," *The Astrophysical Journal Supplement Series*, vol. 135, pp. 41–61, 2001.

[59] P. Ferragina and G. Manzini, "Indexing compressed text," *Journal of the ACM*, vol. 52, no. 4, pp. 552–581, July 2005.

[60] FITS, "Flexible Image Transport System," International Astronomical Union, 2010, http://heasarc.gsfc.nasa.gov/docs/heasarc/fits.html.

[61] P. Flanagan, "Giardia–diagnosis, clinical course and epidemiology. a review," *Epidemiology and infection*, vol. 109, no. 1, pp. 1–22, 1992.

[62] B. J. Ford, "The Discovery of Giardia," *Microscope*, vol. 53, no. 4, pp. 147–153, 2005.

[63] D. Fradkin and D. Madigan, "Experiments with random projections for machine learning," in *KDD 2003: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2003, pp. 517–522.

[64] P. Frankl and H. Maehara, "The Johnson-Lindenstrauss lemma and the sphericity of some graphs," *Journal of Combinatorial Theory, Series B*, vol. 44, no. 3, pp. 355–362, 1988.

[65] K. Fraser, Z. Wang, and X. Liu, *Microarray Image Analysis*. CRC Press, 2010.

[66] R. Freedman and W. Kaufmann, *Universe*, 6th ed. W. H. Freeman, 2001.

[67] J. Gailly and M. Adler, "zlib," 2007, http://www.zlib.net.

[68] GALEX, "The Galaxy Evolution Explorer," 2006, http://www.galex.caltech.edu/index.html.

[69] G. Gan, C. Ma, and J. Wu, *Data Clustering Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics. SIAM, 2007.

[70] J. F. Gantz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva, "The diverse and exploding digital universe: An update forecast of worldwide information growth through 2011," in *IDC White Paper*. IDC, March 2008.

[71] J. F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xhenei, A. Toncheva, and A. Manfrediz, "The expanding digital universe: A forecast of worldwide information growth through 2010," in *IDC White Paper*. IDC, March 2007.

[72] T. Gardner and D. Hill, "Treatment of giardiasis," *Clinical Microbiology Reviews*, vol. 14, no. 1, pp. 114–128, 2001.

[73] Giardia, "The Giardia lamblia organism," 2010, Mission Pharmacal Company, http://www.giardiasis.org.

[74] V. J. Gillet, D. J. Wild, P. Willett, and J. Bradshaw, "Similarity and dissimilarity methods for processing chemical structure databases," *The Computer Journal*, vol. 41, no. 8, pp. 547–558, 1998.

[75] B. Goetz, "Java theory and practice: Where's your point?" IBM, 2003, http://www.ibm.com/developerworks/java/library/j-jtp0114/.

[76] A. D. Gordon, "A review of hierarchical classification," *Journal of the Royal Statistical Society. Series A (General)*, vol. 150, no. 2, pp. 119–137, 1987.

[77] P. Grabusts and A. Borisov, "Using grid-clustering methods in data classification," in *PARELEC '02: Proceedings of the International Conference on Parallel Computing in Electrical Engineering*. Washington, DC, USA: IEEE Computer Society, 2002, p. 425.

[78] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *IEEE Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[79] D. A. Grossman and O. Frieder, *Information Retrieval: Algorithms and Heuristics*, 2nd ed. Springer, 2004.

[80] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[81] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2–3, pp. 107–145, 2001.

[82] ——, "Cluster validity methods: part I," *ACM SIGMOD Record*, vol. 31, no. 2, pp. 40–45, 2002.

[83] J. Hartigan and M. Wong, "A *k*-means clustering algorithm," in *Applied Statistics*, vol. 28, no. 1. Blackwell Publishing for the Royal Statistical Society, 1979, pp. 100–108.

[84] J. A. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.

[85] R. Hecht-Nielsen, *Computational intelligence: Imitating life*. IEEE Press, 1994, ch. Context Vectors; General Purpose Approximate Meaning Representations Self-Organized from Raw Data.

[86] A. Hinneburg and D. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering," in *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 506–517.

[87] A. Hinneburg and D. A. Keim, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceeding of the 4th International Conference on Knowledge Discovery and Data Mining*. New York: AAAI Press, 1998, pp. 58–68.

[88] IVOA, "International Virtual Observatory Alliance," 2009, http://www.ivoa.net.

[89] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computer Surveys*, vol. 31, no. 3, pp. 264–323, September 1999.

[90] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[91] A. K. Jain, A. Topchy, M. H. C. Law, and J. M. Buhmann, "Landscape of clustering algorithms," in *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, vol. 1. Washington, DC, USA: IEEE Computer Society, 2004, pp. 260–263.

[92] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz maps into a Hilbert space," in *Conference in modern analysis and probabilities*, vol. 26. Contemporary Mathematics. American Mathematical Society, 1984, pp. 189–206.

[93] W. Kahan and J. Darcy, "How Java's floating-point hurts everyone everywhere," 1998, http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf.

[94] S. Kaski, "Dimensionality reduction by random mapping: fast similarity computation for clustering," in *IEEE International Joint Conference on Neural Networks*. IEEE, May 1998.

[95] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data. An Introduction to Cluster Analysis*. New Jersey: John Wiley & Son, 1990.

[96] J. R. Kettenring, "Massive datasets," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 25–32, 2009.

[97] A. Y. Khrennikov and S. V. Kozyrev, "p-Adic numbers in bioinformatics: from genetic code to PAM-matrix," 2009, http://arxiv.org/abs/0903.0137.

[98] A. Khrennikov and S. Kozyrev, "Genetic code on the diadic plane," *Physica A*, vol. 381, pp. 265–272, 2007.

[99] M. Konchady, *Text Mining Application Programming*. Charles River Media, 2006.

[100] H. C. Law, "Clustering, Dimensionality Reduction, and Side Information," Ph.D. dissertation, Michigan State University, 2006, http://dataclustering.cse.msu.edu/papers/martin_law_thesis_compact.pdf.

[101] I. C. Lerman, *Classification et Analyse Ordinale des Données*. Paris: Dunod, 1981.

[102] A. M. Lesk, *Introduction to Bioinformatics*, 2nd ed. Oxford: Oxford University Press, 2007.

[103] C. Lewis, "Suffix trees in computational biology," 2010, http://homepage.usask.ca/~ctl271/857/suffix_tree.shtml.

[104] P. Li, T. Hastie, and K. Church, "Very sparse random projections," in *KDD 2006: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, vol. 1. New York, NY, USA: ACM, 2006, pp. 287–296.

[105] J. Lin and D. Gunopulos, "Dimensionality reduction by random projection and latent semantic indexing," in *3rd SIAM International Conference on Data Mining*, San Francisco, CA, USA, March 2003.

[106] H. Lodish, A. Berk, C. A. Kaiser, M. Krieger, M. P. Scott, A. Bretscher, H. Ploegh, and P. Matsudaira, *Molecular Cell Biology*, 6th ed. New York, NY, USA: W. H. Freeman, 2007.

[107] G. Longo, "DAME," Data Mining & Exploration, 2010, http://people.na.infn.it/~astroneural/.

[108] M. Lorr, *Cluster Analysis for Social Scientists*. San Francisco: Jossey-Bass, 1983.

[109] Lucene, "Lucene Apache project," 2009, http://lucene.apache.org.

[110] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. LeCam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[111] E. Mäkelä, O. Suominen, and E. Hyvnen, "Automatic exhibition generation based on semantic cultural content," in *Proceedings of the Cultural Heritage on the Semantic Web Workshop at the 6th International Semantic Web Conference (ISWC 2007)*, November 12 2007.

[112] J. C. McDowell, "Special report astronomy: downloading the sky," *IEEE Spectrum*, vol. 41, no. 8, pp. 35–39, 2004.

[113] M. L. Miller, M. A. Rodriguez, and I. J. Cox, "Audio fingerprinting: Nearest neighbor search in high dimensional binary spaces," *Journal of VLSI Signal Processing Systems*, vol. 41, no. 3, pp. 285–291, 2005.

[114] B. Mirkin, *Mathematical Classification and Clustering: Nonconvex Optimization and Its Applications*. Springer, 1996.

[115] ——, *Clustering for Data Mining: A Data Recovery Approach*. Chapman & Hall/CRC, 2005.

[116] B. Mirkin and P. Fishburn, *Group Choice*. V. H. Winston, 1979.

[117] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Mathematics*, vol. 1, no. 2, pp. 226–251, 2003.

[118] D. Monniaux, "The pitfalls of verifying floating-point computations," *ACM Transactions on Programming Languages and Systems*, vol. 30, no. 3, pp. 1–41, 2008.

[119] H. G. Morrison, A. G. McArthur, F. D. Gillin, S. B. Aley, R. D. Adam, G. J. Olsen, A. A. Best, W. Z. Cande, F. Chen, M. J. Cipriano, B. J. Davids, S. C. Dawson, H. G. Elmendorf, A. B. Hehl, M. E. Holder, S. M. Huse, U. U. Kim, E. Lasek-Nesselquist, G. Manning, A. Nigam, J. E. J. Nixon, D. Palm, N. E. Passamaneck, A. Prabhu, C. I. Reich, D. S. Reiner, J. Samuelson, S. G. Svard, and M. L. Sogin, "Genomic minimalism in the early diverging intestinal parasite Giardia lamblia," *Science*, vol. 317, no. 5946, pp. 1921–1926, 2007.

[120] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *Computer Journal*, vol. 26, no. 4, pp. 354–359, 1983.

[121] ——, "Counting dendrograms: a survey," *Discrete Applied Mathematics*, vol. 7, no. 2, pp. 191–199, 1984.

[122] ——, *Multidimensional Clustering Algorithms*. Physica-Verlag, 1985.

[123] ——, "On ultrametricity, data coding, and computation," *Journal of Classification*, vol. 21, pp. 167–184, 2004.

[124] ——, "Quantifying ultrametricity," in *COMPSTAT 2004 – Proceedings in Computational Statistics*, J. Antoch, Ed. Springer, 2004, pp. 1561–1568.

[125] ——, "Thinking ultrametrically," in *Classification, Clustering, and Data Mining Applications. Proceedings of the Meeting of the International Federation of Classification Societies (IFCS)*, D. Banks, L. House, F. R. McMorris, P. Arabie, and W. Gaul, Eds. Springer, July 2004, pp. 3–14.

[126] ——, *Correspondence Analysis and Data Coding with Java and R*. Clapman & Hall/CRC, 2005.

[127] ——, "Identifying the ultrametricity of time series," in *The European Physical Journal B*, vol. 43, no. 4. Springer Berlin, February 2005, pp. 573–579.

[128] ——, "The Haar wavelet transform of a dendrogram," *Journal of Classification*, vol. 24, no. 1, pp. 3–32, 2007.

[129] ——, "The remarkable simplicity of very high dimensional data: Application of model-based clustering," *Journal of Classification*, vol. 26, no. 3, pp. 249–277, 2009.

[130] F. Murtagh, G. Downs, and P. Contreras, "Hierarchical clustering of massive, high dimensional data sets by exploiting ultrametric embedding," *SIAM Journal on Scientific Computing*, vol. 30, no. 2, pp. 707–730, February 2008.

[131] F. Murtagh and J.-L. Starck, "Pattern clustering based on noise modeling in wavelet space," *Pattern Recognition*, vol. 31, no. 7, pp. 847–855, 1998.

[132] F. Murtagh, J.-L. Starck, and M. W. Berry, "Overcoming the curse of dimensionality in clustering by means of the wavelet transform," *The Computer Journal*, vol. 43, no. 2, pp. 107–120, 2000.

[133] MySQL, "MySQL Community Server," 2009, http://www.mysql.com.

[134] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, March 2007.

[135] G. Navarro, E. S. de Moura, M. Neubert, N. Ziviani, and R. Baeza-Yates, "Adding compression to block addressing inverted indexes," *Information Retrieval*, vol. 3, no. 1, pp. 49–77, July 2000.

[136] G. Navarro and V. Mäkinen, "Compressed full-text indexes," *ACM Computing Surveys*, vol. 39, no. 1, May 2007, http://doi.acm.org/10.1145/1216370.1216372.

[137] G. Navarro and M. Raffinot, *Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences*. Cambridge University Press, 2007.

[138] Oracle, "Berkeley Data Base," 2007, http://www.oracle.com/database/berkeley-db/index.html.

[139] M. L. Overton, *Numerical Computing with IEEE Floating Point Arithmetic*. Society for Industrial and Applied Mathematics. SIAM, 2001.

[140] Palomar, "Palomar Observatory," 2008, http://www.astro.caltech.edu/palomar.

[141] N. H. Park and W. S. Lee, "Statistical grid-based clustering over data streams," *SIGMOD Record*, vol. 33, no. 1, pp. 32–37, 2004.

[142] W. Pearson and D. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 85, no. 8, pp. 2444–2448, 1988.

[143] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, July 1997.

[144] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 2007.

[145] ProteomeCommons, "FASTA and ProteomeCommons," 2010, https://proteomecommons.org/data/fasta/hupo_standard.jsp.

[146] A. Rapoport and S. Fillenbaum, “An experimental study of semantic structures,” *Multidimensional Scaling; Theory and Applications in the Behavioral Sciences*, vol. 2, pp. 93–131, 1972.

[147] V. Reshetnikov, “Sky surveys and deep fields of ground-based and space telescopes,” *Physics-Uspekhi*, vol. 48, no. 11, pp. 1109–1127, 2005.

[148] R. L. Rivest, “The MD5 message-digest algorithm,” MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992, http://people.csail.mit.edu/rivest/Rivest-MD5.txt.

[149] A. D. Rodney, “The biology of Giardia spp,” *American Society for Microbiology*, vol. 55, no. 4, pp. 706–732, 1991.

[150] T. Ruotsalo and E. Hyvönen, “A method for determining ontology-based semantic relevance,” in *Proceedings of the International Conference on Database and Expert Systems Applications DEXA 2007*. Regensburg, Germany: Springer, 2007, pp. 680–688.

[151] M. Sahlgren, “An introduction to random indexing,” in *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering*, Copenhagen, Denmark, 2005.

[152] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, Jan 1998.

[153] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, “Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications,” *Data Mining Knowledge Discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[154] E. Schikuta, “Grid-clustering: An efficient hierarchical clustering method for very large data sets,” in *ICPR ’96: Proceedings of the 13th International Conference on Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 1996, p. 101.

[155] SDSS, “Sloan Digital Sky Survey,” 2008, http://www.sdss.org.

[156] C. Seidel, “Plotting a table of numbers as an image using R,” 2010, http://www.phaget4.org/R/image_matrix.html.

[157] G. Sheikholeslami, S. Chatterjee, and A. Zhang, “Wavecluster: a wavelet-based clustering approach for spatial data in very large databases,” *The VLDB Journal*, vol. 8, no. 3-4, pp. 289–304, 2000.

[158] SIAM-News, “Inquiry board traces Ariane 5 failure to overflow error,” SIAM, vol. 29, no. 8, pp. 1, 12-13, October 1996.

[159] L. Silveira, "Enhanced full-text self-indexes based on Lempel-Ziv compression," Ph.D. dissertation, Universidad Técnica de Lisboa, 2007, http://kdbio.inesc-id.pt/~lsr/phd/book.r.ps.

[160] J. W. Smith and M. S. Wolfe, "Giardiasis," *Annual Reviews of Medicine*, vol. 31, pp. 373–383, 1980.

[161] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy*. San Francisco: Freeman, 1973.

[162] I. Soboroff, A. de Vries, and N. Craswell, "Overview of the TREC 2006 enterprise track," in *The Fourteenth Text REtrieval Conference (TREC 2005) Proceedings*, E. Voorhees and L. Buckland, Eds., November 2005, http://trec.nist.gov/pubs/trec14/papers/ENTERPRISE.OVERVIEW.pdf.

[163] Spitzer, "Spitzer Space Telescope," 2007, http://www.spitzer.caltech.edu.

[164] M. Steinbach, L. Ertöz, and V. Kumar, *New Directions in Statistical Physics: Econophysics, Bioinformatics and Pattern Recognition*. Berlin: Springer, 2004, ch. The Challenges of Clustering High Dimensional Data, pp. 273–309.

[165] Swift, "Swift Gamma-Ray Burst Explorer Mission," 2007, http://www.swift.psu.edu.

[166] Tomcat, "Apache Tomcat," 2008, http://tomcat.apache.org.

[167] A. Trotman, "Compressing inverted files," *Information Retrieval*, vol. 6, no. 1, pp. 5–19, January 2003.

[168] A. van Rooij, *Non-Archimedean Functional Analysis*. Marcel Dekker, 1978.

[169] S. S. Vempala, *The Random Projection Method. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 2004, vol. 65.

[170] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 1073–1080.

[171] VLA, "Very Large Array," 2008, http://www.vla.nrao.edu.

[172] L. Wang and Z.-O. Wang, "CUBN: a clustering algorithm based on density and distance," in *Proceeding of the 2003 International Conference on Machine Learning and Cybernetics*. IEEE Press, 2003, pp. 108–112.

[173] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 186–195.

[174] J. D. Watson, T. A. Baker, S. P. Bell, A. Gann, M. Levine, and R. Losick, *Molecular Biology of the Gene*, 6th ed. San Francisco, CA, USA: Pearson/Benjamin Cummings, 2008.

[175] S. M. Weiss, N. Indurkhya, T. Zhang, and F. J. Damerau, *Text Mining. Predictive Methods for Analyzing Unstructured Information*. Springer, 2005.

[176] WHO, *Addendum: Microbiological agents in drinking water*, 2nd ed. World Health Organization, 2003, ch. Protozoan Parasites (Cryptosporidium, Giardia, Cyclospora), http://www.who.int/water_sanitation_health/dwq/microbioladd/en/index.html.

[177] I. Witten, A. Moffat, and T. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. Morgan Kaufmann, May 1999.

[178] M. Wright, "Fingerprinting and dictionary generation," 2009, http://www.digitalchemistry.co.uk/prod_fingerprint.html.

[179] J. Xiong, *Essential Bioinformatics*. Cambridge: Cambridge University Press, 2006.

[180] XMM-Newton, "ESA XMM-Newton," 2008, http://xmm.esac.esa.int/.

[181] R. Xu and D. C. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.

[182] ——, *Clustering*. IEEE Computer Society Press, 2008.

[183] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *ICDE '98: Proceeding of the Fourteenth International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 1998, pp. 324–331.

[184] X. Xu, J. Jäger, and H.-P. Kriegel, "A fast parallel clustering algorithm for large spatial databases," *Data Mining Knowledge Discovery*, vol. 3, no. 3, pp. 263–290, 1999.

[185] J. Zaat, T. Mank, and W. Assendelft, "A systematic review on the treatment of giardiasis," *Tropical Medicine & International Health*, vol. 2, no. 1, pp. 63–82, 1997.

[186] O. R. Zaïane and C.-H. Lee, "Clustering spatial data in the presence of obstacles: a density-based approach," in *IDEAS '02: Proceedings of the 2002 International Symposium on Database Engineering and Applications*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 214–223.

[187] A. Zhang, *Protein Interaction Networks. Computational Analysis*. Cambridge: Cambridge University Press, 2006.

[188] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 5, pp. 337–343, 1977.

[189] ——, "Compression of individual sequences via variable-rate coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[190] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, pp. 69–82, July 2006.

[191] M. Zvelebil and J. O. Baum, *Understanding Bioinformatics*. Garland Science, 2007.