

University of Wollongong - Research Online

Thesis Collection

Title: Knowledge libraries and information space

Author: Eric Rayner

Year: 2009

Repository DOI:

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material.

Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.

Research Online is the open access repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

University of Wollongong Thesis Collections

University of Wollongong Thesis Collection

University of Wollongong

Year 2009

Knowledge libraries and information
space

Eric Rayner
University of Wollongong

Rayner, Eric, Knowledge libraries and information space, Doctor of Philosophy thesis, School of Computer Science and Software Engineering - Faculty of Informatics, University of Wollongong, 2009. <http://ro.uow.edu.au/theses/3027>

This paper is posted at Research Online.

NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

KNOWLEDGE LIBRARIES AND INFORMATION SPACE

A thesis submitted in partial fulfilment of the requirements for the award of
the degree

DOCTOR OF PHILOSOPHY

from the

UNIVERSITY OF WOLLONGONG

by

ERIC RAYNER, BCompSc(Hons)

**SCHOOL OF COMPUTER SCIENCE AND
SOFTWARE ENGINEERING**

2009

Thesis Certification

I, Eric P. Rayner, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Information and Computer Science, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Eric Rayner
July 27, 2009

Contents

List of Figures	ix
List of Tables	xiii
Abbreviation, Notation and Typographical Conventions	xvii
Abstract	xix
Acknowledgements	xxi
1 Introduction	1
1.1 Thesis Overview	1
1.2 The Contribution of this Thesis	6
1.3 The Organisation of Information	7
1.4 Thesis Methodology	9
1.4.1 Gap in the literature	10
1.4.2 Research hypothesis	11
1.4.3 Experiment	11
1.5 How to Read this Thesis	12
1.6 Thesis Outline	13
1.7 The Nature of Information	14
1.8 Limitations of this Thesis	17
1.9 Knowledge Libraries	18
1.10 Information Space	19
1.11 Information Organisation Terms	23
2 The Organisation of Information	25
2.1 Overview	25
2.2 Introduction	26
2.3 Traditional Classification	27
2.3.1 Faceted classification	29
2.3.2 ISBN, ISSN, MARC and CIP	30

2.3.3	A critique of traditional classification	32
2.4	Computer File Systems	34
2.4.1	A critique of computer file systems	36
2.5	The Database	36
2.5.1	Critique of the database	38
2.6	Information Retrieval	38
2.6.1	Measuring the effectiveness of IR	40
2.6.2	A critique of IR	41
2.7	Data Warehousing	44
2.7.1	Critique	48
2.8	The Internet, World Wide Web and Semantic Web	48
2.8.1	The internet	48
2.8.2	The world wide web	49
2.8.3	The Semantic Web	50
2.9	Discussion and Summary	52
2.10	What this chapter achieved	52
3	Knowledge Libraries	53
3.1	Overview	53
3.2	Introduction	54
3.3	Knowledge Library User Scenarios	55
3.4	The Uses of Knowledge Libraries	69
3.4.1	Knowledge Libraries for Research	69
3.4.2	Knowledge Libraries for Education	71
3.4.3	Knowledge Libraries for Business	72
3.5	Core Knowledge Library Functionality	72
3.5.1	Core knowledge library administration functionality . .	73
3.5.2	Core knowledge library end use functionality	73
3.6	Knowledge Library Security	74
3.7	Extended Knowledge Library Functionality	74
3.7.1	Automatic dimensions	75
3.7.2	Dynamic dimensions	75
3.7.3	Automatic report generation	76
3.7.4	Automatic notification	77
3.7.5	Knowledge library graphical user interface	78
3.8	What this chapter achieved	78
4	Spaces for Information Organisation	79
4.1	Overview	79
4.2	Potential Mathematical Bases	80

4.2.1	Metric Space	81
4.2.2	Vector Space	82
4.2.3	The vector model for information retrieval	83
4.2.4	Lattices and Topological Space	89
4.2.5	Formal Concept Analysis	90
4.2.6	The Relational Model	93
4.2.7	Online Analytical Processing (OLAP)	95
4.3	Space	96
4.4	n -Dimensional Spaces	97
4.5	Nested Spaces	100
4.6	Span	101
4.7	Spans of Points in n -Dimensional Spaces	104
4.8	The Generalised Triangle Inequality and Set Space	105
4.9	Other Properties of Set Distance Functions	106
4.9.1	\subseteq -Reflexivity	106
4.9.2	$\not\subseteq$ -Strict Positiveness	107
4.9.3	$\not\subseteq^d$ -strict positiveness	108
4.10	Signed Distances	109
4.11	What this Chapter Achieved	109
5	Set Spaces	111
5.1	Overview	111
5.2	Manipulating Set Space	112
5.2.1	n -Dimensional Set Spaces	112
5.2.2	Other Properties of n -Dimensional Spaces	113
5.2.3	Dimension Nesting	114
5.2.4	Dilated and Translated Spaces	115
5.3	Set Distance Functions Based on Set Operations	116
5.4	The d_{ij}^M Set Distance function	118
5.4.1	The Triangle Inequality ($\triangle I$)	122
5.4.2	Span	123
5.4.3	The Generalised Triangle Inequality ($G\triangle I$)	125
5.4.4	\subseteq -reflexivity	129
5.4.5	$\not\subseteq^d$ -strict positiveness	129
5.5	What this Chapter Achieved	130
6	L-Collections	133
6.1	Overview	133

6.2	Background	134
6.2.1	Sets	134
6.2.2	Multisets	135
6.2.3	Merges and Joins	136
6.2.4	Indexed Families	137
6.2.5	Rough Sets	138
6.2.6	Fuzzy sets	139
6.3	L -Collections	142
6.3.1	L -collection operators	143
6.4	Sets, Multisets, Fuzzy Sets, Rough Sets and L -Collections . . .	147
6.4.1	Sets and L -Collections	147
6.4.2	Multisets and L -Collections	148
6.4.3	Fuzzy Sets and L -Collections	148
6.4.4	Rough Sets and L -Collections	148
6.5	Extending Proofs Over Sets and Multisets to $\{1\}$, \mathbb{N}_1 and $\mathbb{Q}^{>0}$ -Collections	148
6.6	What this Chapter Achieved	150
7	L-Collection Space	151
7.1	Overview	151
7.2	L -Collection Distance Functions	152
	An L -Collection Distance Function	152
7.2.1	ΔI for $ \mathcal{X} - \mathcal{X} \cap \mathcal{Y} $	153
7.2.2	\mathcal{Q}^d -strict positiveness for $ \mathcal{X} - \mathcal{X} \cap \mathcal{Y} $	153
7.2.3	\subseteq -reflexivity for $ \mathcal{X} - \mathcal{X} \cap \mathcal{Y} $	154
7.3	The $d_{ij}^{\mathcal{M}}$ L -Collection Distance Function	154
7.4	The ${}_k^{\mathcal{M}}d$ L -Collection Distance Function	156
7.4.1	Span and $G\Delta I$ for ${}_k^{\mathcal{M}}d$	159
7.4.2	\subseteq -reflexivity for ${}_k^{\mathcal{M}}d$ distance functions	162
7.4.3	\mathcal{Q}^d -strict positiveness for ${}_k^{\mathcal{M}}d$ distance functions	163
7.5	The ${}_{av}^{\mathcal{M}}d$ L -Collection Distance Function	164
7.5.1	$G\Delta I$ for ${}_{av}^{\mathcal{M}}d$	166
7.5.2	Other properties of ${}_{av}^{\mathcal{M}}d$	174
7.6	What this Chapter Achieved	174
8	Information Space	177
8.1	Overview	177
8.2	Introduction	178
8.3	Networked Space	179
8.4	Classification Space	184

8.4.1	Classification spaces for uncertain and partial classifications	185
8.4.2	Many levelled classification spaces	187
8.4.3	Projected classification spaces	188
8.5	Working with Classification Space	190
8.5.1	Creation	190
8.5.2	Addition and subtraction	190
8.5.3	Point selection	191
8.5.4	Ordering	194
8.6	Attaching Information Units to Points in Classification Spaces	194
8.6.1	Distance	195
8.6.2	Indexing classification spaces	196
8.7	Information Space	198
8.8	Working with Information Space	199
8.8.1	Creation	199
8.8.2	Index Manipulation	199
8.8.3	Information unit selection	202
8.8.4	Selecting points in information space	204
8.9	What this Chapter Achieved	205
9	Basing Knowledge Libraries on Information Space	207
9.1	Overview	207
9.2	Information Space for Questionnaire Knowledge Libraries	208
9.2.1	Example Questionnaire Information Space	210
9.3	Information Space for Research Paper Knowledge Libraries	213
9.3.1	Selecting and Comparing Research Papers	214
9.3.2	Extended Dimensions	215
9.3.3	Further Dimensions	217
9.4	What this Chapter Achieved	218
10	The Efficient Implementation of Knowledge Libraries	219
10.1	Overview	219
10.2	Introduction	220
10.2.1	Distance query	220
10.2.2	Range query	221
10.2.3	k Nearest neighbour query	221
10.2.4	Ranked query	221
10.2.5	Sequential search range query algorithm	222

10.3	Metric Space Algorithms	222
10.3.1	Relative ordering	223
10.3.2	Radius partitioning	224
10.3.3	Hyperplane partitioning	226
10.3.4	Ranked query and k -NN query algorithms	227
10.3.5	A critique of the literature	228
10.4	Adapting Metric Space Algorithms for Set Space	229
10.4.1	Relative ordering for set spaces	230
10.4.2	Radius partitioning for set space	231
10.4.3	Hyperplane partitioning for set spaces	233
10.5	Searching hard spaces	234
10.5.1	Specialised algorithms for searching hard spaces	235
10.5.2	Specialised algorithms for searching n -dimensional spaces	236
10.6	What this Chapter Achieved	238
11	Experimental Results and Discussion	241
11.1	Overview	241
11.2	Introduction	242
11.3	Set Space Radius Partitioning Implementation: Non symmetric Experiments	243
11.3.1	Non Symmetric Experiment setup	244
11.3.2	Non Symmetric Experimental results	247
11.3.3	Discussion of non symmetric results	248
11.4	Variance Experiments	251
11.4.1	Variance experimental results	253
11.4.2	Random edge weight experimental results	253
11.4.3	Euclidean space experimental results	256
11.5	Center Selection and Multiple Tree Experiments	257
11.5.1	Greatest minimum center selection experiment	259
11.5.2	Standard deviation center selection experiment	259
11.5.3	Discussion of center selection experiments	261
11.5.4	Experiment with multiple search trees	261
11.6	Experiments with Multi-Dimensional Spaces	264
11.6.1	Multi-Dimensional experiment setup	264
11.6.2	Multi-Dimensional experimental results and discussion	265
11.6.3	Multiple tree experiments	265
11.7	Set Space Experiments	269
11.7.1	Discussion of set space results	270
11.8	Sequential search algorithms	270

11.8.1	Sequential search for n -dimensional spaces	272
11.8.2	Sequential search over set spaces with set distance func- tions	272
11.8.3	Discussion of sequential search results	273
11.9	Introducing the Sequential-Hybrid Algorithm	274
11.9.1	Discussion of sequential-hybrid results	274
11.10	Summary, discussion and recommendations	275
11.11	What this Chapter Achieved	276
12	Summary, Discussion and Future Work	279
12.1	Chapter Overview	279
12.2	Thesis Summary	280
12.2.1	The importance of information systems	280
12.2.2	The significance of Knowledge Libraries	281
12.2.3	The mathematical basis for Knowledge Libraries	282
12.2.4	Implementing Knowledge Libraries	284
12.3	Discussion	284
12.3.1	The Contribution of this Thesis	285
12.3.2	The more formal development of Knowledge Libraries	286
12.3.3	The flexibility of information space	286
12.4	Future Work	287
12.4.1	The implementation of Knowledge Libraries	287
12.4.2	Graphical Interface	287
12.4.3	Improving existing systems	288
12.4.4	The dissemination of knowledge	289
Appendix A: A Guide to the Accompanying CD		291
Appendix B: Publications Relating to this Thesis		299
Appendix C: Glossary of Information Organisation Terms		301

List of Figures

2.1	A 13-digit ISBN with EAN-13 bar code	31
4.1	A Hasse diagram of a concept lattice of objects = $\{1, \dots, 10\}$ and attributes = {composite, even, odd, prime, square}. . . .	92
4.2	Three balls X, Z, Y in \mathbb{R}^2	102
5.1	Subset element counts for \subseteq -reflexive proofs. $a = X $, $b =$ $ Y - X $	116
5.2	Subset element counts for $\triangle I$ proofs. $a = Z - X - Y $, $b = Z \cap X - Y $, $c = Z \cap X \cap Y $	117
5.3	Subset element counts for $\not\subseteq^d$ -strict positive proofs. $a = X -$ $Y $, $b = X \cap Y $ and $c = Y - X $	117
7.1	Sub L -collection element counts for $\triangle I$ proofs. $a = \mathcal{Z} - \mathcal{X} -$ $\mathcal{Y} $, $b = \mathcal{Z} \cap \mathcal{X} - \mathcal{Y} $, $c = \mathcal{Z} \cap \mathcal{X} \cap \mathcal{Y} $, etc.	153
7.2	Sub L -collection element counts for $\not\subseteq$ -strict positive proofs. $a = \mathcal{X} - \mathcal{Y} $, $b = \mathcal{X} \cap \mathcal{Y} $ and $c = \mathcal{Y} - \mathcal{X} $	153
7.3	Sub L -collection element counts for \subseteq -reflexive proofs. $a =$ $ \mathcal{X} $, $b = \mathcal{Y} - \mathcal{X} $	154
11.1	Frequency of distances for typical 1000-point network spaces with $maxDist = 25$ and 1500, 2000, 2500 and 3000 directed, $weight = 1$ edges.	245
11.2	Frequency of distances for typical 1000-point network spaces with $maxDist = 25$. LHS: 1100 undirected, $weight = 1$ edges. RHS: 2200 directed, $weight = 1$ edges.	247

- 11.3 Data from a range query ($x = 999$, $t = 2$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) over 1000 different, 1000-point, metric (network) spaces generated from networks with 1100 $weight = 1$ undirected edges and $maxDist = 25$. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage). 251
- 11.4 Frequency of distances. **LHS**: a typical 1000-point network space with $maxDist = 255$ and $1.1n$ undirected edges with uniform random weights (integers 1 to 19) . Mean distance: 120.913, standard deviation: 39.2604. **RHS**: a 1-dimensional Euclidean space over integers 0–999. Mean distance: 333.333, standard deviation: 235.702. 255
- 11.5 Data from a range query ($x = 999$, $t = 50$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) over 1000 different, 1000-point, metric (network) spaces (with 1100 randomly weighted undirected edges). **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage). Compare with figure 11.3. 255
- 11.6 Data from the range query $x = 999$, $t = 50$ on 1000 radius partitioning search trees (with randomly selected centers, $branchFactor = 2$ and $leafCapacity = 10$) over a 1000-point uniform Euclidean space. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage). Compare with figure 11.3. 256

11.7	Data from a range query ($x = 999$, $t = 2$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) with specially selected centers over 1000 different, 1000-point, metric (network) spaces (with 1100 $weight = 1$ undirected edges). LHS : distribution of candidate point set size as a (truncated) percentage of space size. RHS : distribution of retrieved to candidate point set sizes (by truncated percentage). Compare with figure 11.3.	260
11.8	Data for the range query $x = 999$, $t = 2$ over 3 radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for each of 1000 different, 1000-point, metric (network) spaces (with 1100 $weight = 1$ undirected edges). The intersection of the 3 resulting candidate point sets was taken as the candidate point set for this search method. LHS : distribution of candidate point set size as a (truncated) percentage of space size. RHS : distribution of retrieved to candidate point set sizes (by truncated percentage).	263
11.9	Distance frequencies for the “first” 1000 points in 2,3,4 and 5–dimensional uniform Euclidean spaces. Distances are truncated in the plot, but not when computing the mean and standard deviation. The 2–dimensional space has 32 coordinates each dimension. The others have 10, 6 and 4 (respectively).	266
11.10	Distribution of retrieved to candidate point set sizes (by truncated percentage) for the range query $x = 0$, $t = 2$ over 1000 radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for uniform 1000-point, 2,3,4 and 5–dimensional Euclidean space.	267

11.11	Distribution of retrieved to candidate set sizes (by truncated percentage) for the range query $x = 0$, $t = 2$ over 1000 different groups of three radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for a uniform 1000-point, 2,3,4 and 5-dimensional Euclidean space. The group candidate set is the intersection of the candidate sets corresponding to each of the three trees. . . .	268
11.12	Data from 1000 range queries ($0 \leq x \leq 999$, $t = 50$) on 1000 different radius partitioning search trees (with random centers, $branchFactor = 2$, $leafCapacity = 10$) over the space $\langle \{0, \dots, 999\}, x - y \rangle$. LHS : distribution of candidate point set size as a (truncated) percentage of space size. RHS : distribution of retrieved to candidate point set sizes (by truncated percentage).	269
11.13	Data from a range query ($x = 999$, $t = 2$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) over 1000 different, 1000-point, (non symmetric) network spaces generated from networks with 2200 $weight = 1$ directed edges and $maxDist = 25$. LHS : distribution of candidate point set size as a (truncated) percentage of space size. RHS : distribution of retrieved to candidate point set sizes (by truncated percentage).	270

List of Tables

2.1	Simplified Star Schema for Nationwide Retail Chain	46
3.1	Dimension types	56
4.1	Eight relations illustrating operations defined in [24]	94
5.1	Distances for three distance functions over $X = \{1, 3\}$, $Y = \{5, 9\}$ and $Z = \{3, 5\}$	123
10.1	Ball to enclosing hypercube volume (4 s.f.) for different n in Euclidian space	237
11.1	Average, over 1000 distinct queries ($t = 2$, $0 \leq x < 1000$), radius partitioning tree search ($branchFactor = 2$, $leafCapacity = 10$) and sequential search range query times (milliseconds) for typical symmetric network spaces generated from (random) $n = 1000$, $e = 1100$; $n = 10000$, $e = 13000$; and $n = 100000$, $e = 150000$ (undirected) networks and non symmetric network spaces generated from (random) $n = 1000$, $e = 2200$; $n = 10000$, $e = 26000$; and $n = 100000$, $e = 300000$ (directed) networks.	248
11.2	Radius partitioning search tree ($branchFactor = 2$ and $leafCapacity = 10$) for a typical, random, 1000-point network space (with 1100 undirected $weight = 1$ edges).	254

11.3	Candidate nodes, collisions, pruned nodes and points, considered and retrieved points by level from a range query ($x = 999$, $t = 2$) on the radius partitioning search tree in table 11.2. The retrieved point set contained 7 points, while the candidate point set contained 198 points, giving a retrieved to considered ratio of approximately 3%.	254
11.4	Typical radius partitioning search tree (with random centers, $branchFactor = 2$ and $leafCapacity = 1$) for a 1-dimensional Euclidean space over integers 0–999. Compare with table 11.2.	258
11.5	Collisions, pruned nodes and points, considered and retrieved points by level from the range query $x = 999$, $t = 50$ on the radius partitioning search tree in table 11.4. Both retrieved and candidate point sets contained 51 points, giving a retrieved to considered ratio of 100%. Compare with table 11.3.	258
11.6	Candidate set sizes for the range query $x = 999$, $t = 2$ over 3 radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for each of 10 different, 1000-point, metric (network) spaces (with 1100 $weight = 1$ undirected edges). The size of the intersection of these 3 sets, and the size of the retrieved set is also displayed.	262
11.7	Space size and query time for the sequential search algorithm (for a C++ implementation, using the <code>cmath sqrt()</code> and <code>pow()</code> functions, on a 1.8GHz machine with 265k memory running Linux) for various n -dimensional spaces with 1000 points in each dimension and 10^6 information elements. “Dimensional distances” d_1, \dots, d_n are combined using $\sqrt{(\sum_{i=1}^n d_i^2)}$	272

11.8	Space size and query time for the sequential search algorithm (for a C++ implementation, using the <code>cmath sqrt()</code> and <code>pow()</code> functions, on a 1.8GHz machine with 265k memory running Linux) for a 3-dimensional space. Each dimension is a set space, based on an underlying space with 1000 points. The algorithm determines distances for 10^6 information units, attached to random points in the space. “Dimensional distances” d_1, d_2, d_3 are combined using $\sqrt{d_1^2 + d_2^2 + d_3^2}$	273
11.9	Retrieved to candidate set sizes (by truncated percentage) for a sequential search range queries over 3,5,7 and 9-dimensional spaces (with 10^6 randomly attached information elements). Each dimension consists of 1000 points. Distances are uniform random integers between 1 and 1000. In each n -dimensional space, the first $n - 1$ dimensions were used to determine the candidate set.	275

Abbreviation, Notation and Typographical Conventions

The set of real numbers is denoted by \mathbb{R} , the set of rational numbers by \mathbb{Q} and the set of natural numbers (integers, strictly larger than 0) by \mathbb{N}_1 . Note that $\mathbb{R}^{\geq 0}$ is the set of real numbers greater than or equal to 0, while $\mathbb{R}^{>0}$ is the set of real numbers strictly greater than 0. Interval notation is used to denote real intervals, so $(0, 1]$ is the set of real numbers less than or equal to 1 and strictly larger than 0. More generally, capital letters, such as L, M, X, Y , are used to denote sets. The power set of any set M (the set of all subsets of M) is denoted $\mathcal{P}(M)$.

Lowercase “math bold font” letters denote vectors, so \mathbf{x} and \mathbf{y} are vectors.

L -collections (introduced in chapter 6) are distinguished from sets by using “math calligraphy font”, so $\mathcal{M}, \mathcal{X}, \mathcal{Y}$ are L -collections.

Enclosing vertical bars are used to denote the cardinality of a set ($|M|$), the cardinality of an L -collection ($|\mathcal{M}|$), the absolute value of a real number ($|d(x, y)|$) and the magnitude of a vector ($|\mathbf{x}|$).

Bold text is used to denote key terms that are defined (or at least described), both within, and (optionally) prior to, the definition. “Double quotes” are used for short quotations (which are also referenced) and when introducing key terms that are not defined.

Finally, *iff* is used as shorthand for “if and only if”.

Abbreviations in this thesis are preceded and introduced by the corresponding, non abbreviated, full term.

Abstract

This research describes and develops **Knowledge Libraries**, idealised systems for organising and presenting information. By providing a mathematical basis, the definition of **information space** establishes a formal foundation for Knowledge Libraries. The definition of information space builds on the new definitions of ***L*-collections**, which generalise sets by allowing a real valued grade to be associated with each element, and **set space**, which generalises metric space to better model the relationships between **information units**.

The **multiple search tree** method improves existing metric space range query algorithms. These algorithms are also generalised to work over set space. The **sequential-hybrid algorithm** enables efficient range queries over multi-dimensional spaces.

Acknowledgements

First and foremost, I would like to thank Dr. Ian Piper, my principal supervisor, for his continuing support throughout my candidacy. Acknowledgement is also due Prof. Martin Bunder for his assistance with the mathematical work in this thesis. Without Prof. Bunder's input, this thesis would not have the emphasis on mathematical correctness that it does.

I would also like to thank my family for their support, particularly my father, Prof. John Rayner, for his willingness to proof read chapters and discuss thesis related matters (at some considerable length).

Chapter 1

Introduction

1.1 Thesis Overview

This chapter provides an introduction—as non technical as possible—to this thesis. This includes an overview of the thesis content, a summary of the main contribution of the thesis, an outline of the topic of **information organisation**, a description of the thesis methodology, instructions on how to read this thesis, an outline of the thesis, a discussion of the nature of information and the limitations of this thesis, a summary of **Knowledge Libraries** and **information space** and, finally, a preamble to the glossary of information organisation terms in appendix C.

Chapter 2 reviews the literature relating to the topic of the organisation of information. Although it appears that there is no body of work that directly addresses this topic, a great many research fields contribute in some way. An exhaustive review is not feasible and is not attempted. Instead, a number of fields that appear to be most relevant to this study are selected. These include **traditional classification**, **computer filesystems**, **databases**, **information retrieval**, **data warehousing**, the **internet**, the **world wide web** and the **semantic web**.

This review is largely non mathematical. The relevant mathematics is reviewed in the preliminary sections of chapters 4,6 and 7—immediately before the associated discussion.

There appears to be a great potential for synergy between the reviewed research fields. It seems likely that the information organisation techniques required to exploit these synergies could also be applied with advantage to other research areas.

Chapter 3 describes the intended operation of **Knowledge Libraries** in a similar spirit to Vannevar Bush’s famous essay “As we may think” [18] and—more recently—Codd, Codd and Salley’s description of “on-line analytical processing” (OLAP) in [25, 26], and Berners-Lee, Hendler and Lassila’s article “The Semantic Web” [10]. The purpose of this chapter is to give the reader—by way of a number of informal Knowledge Library “user scenarios”—an understanding of the objective of this thesis.

This discussion illustrates the importance of the metaphor of **information space** to the organisation of information. Phrases such as “research area”, “knowledge frontier”, “closely related”, “gap in the literature”, etc. indicate the intuitive application of spatial concepts to the organisation of information. This suggests that a definition of information space could serve as the mathematical basis for Knowledge Libraries.

The chapter identifies research, education and business as three main application areas for Knowledge Libraries. Importantly, the core and extended functionality of Knowledge Libraries are identified.

Chapter 4 surveys various mathematical “spaces” that could provide a suitable foundation for information space. These “spaces” are **metric space**; **vector space** and the **vector space model** for information retrieval; **lattices**, **topological space** and **formal concept analysis**; and the **relational model**. Metric space is selected as the most suitable foundation for information space. It is shown how n spaces can be combined to form n -dimensional space. The **projection** and **permutation** of n -dimensional space is discussed and defined, as are **Dimension nested** spaces and **subspace**.

It is shown how a space $\langle M, d \rangle$ can be **nested inside** a space $\langle M', d' \rangle$ where $M' \subseteq \mathcal{P}(M)$ —the power set of M . In order to organise information it is often desirable to make use of spaces such as $\langle M', d' \rangle$ that are **based on** another space $\langle M, d \rangle$ in this way. It is shown that—in part due to the

span of elements that are sets—the metric properties are not suitable for **set spaces** like $\langle M', d' \rangle$. The notion of span is discussed and a definition of the span of n -tuples (points in n -dimensional space) is given. Definitions of **the generalised triangle equality**, \subseteq -**reflexivity** and \mathcal{Z}^d -**strict positive-ness**—metric like properties that are suitable for set spaces—are included.

This chapter provides the beginnings of a mathematical basis for Knowledge Libraries.

Chapter 5 takes a closer look at n -dimensional spaces, set spaces and distances between sets. It is shown that any n -dimensional space will satisfy the generalised triangle inequality if each of its dimensions do. Other properties (of all of the dimensions) are also preserved in the n -dimensional space. It is also shown how **dimension nested** spaces can be “flattened out” by “promoting” nested dimensions so that all dimensions are on the same “level”—without altering the functional characteristics of the space. The chapter also discusses how spaces can be **dilated** and **translated**. All the properties discussed are preserved in dilated spaces. Reflexive and \subseteq -reflexive properties are *not* preserved in translated spaces.

The chapter introduces the **set distance function** $d(X, Y) = |X| - |X \cap Y|$ and shows that it satisfies the triangle inequality, is \mathcal{Z}^d -strict positive and \subseteq -reflexive. The chapter also elaborates on [32], introducing the set comparison d_{ij}^M where both i and j can be any number from 0 to 100 inclusive, or ‘av’. Informally, if $0 < i, j \leq 100$ then i and j work like percentages, so $d_{ij}^M(X, Y) = d(x, y)$ where x is the $i\%$ th closest element of X to Y and y is the $j\%$ th closest element of Y to X . If i is ‘av’ then an average distance, from X to each $y \in Y$, is taken. If j is ‘av’ then an average distance, from each $x \in X$ to Y , is taken. It is shown that d_{ij}^M satisfies the generalized triangle inequality if d does. However only $d_{av\ 0}^M$ and $d_{100\ 0}^M$ are \mathcal{Z}^d -strict positive and \subseteq -reflexive.

This chapter continues to develop a mathematical basis for information space.

While a sufficient mathematical basis for many types of Knowledge Library has now been developed, it is not quite general enough. For example,

information spaces that allow more than one identical **information unit** to be **attached** to the same point cannot be adequately dealt with. The problem in this case is the requirement that each element of a set must be distinct. While multisets would provide a solution for this, it would also be convenient to have “uncertain” and “partial” classifications. To solve these problems, set like objects that allow multiple and partial membership are required.

Chapter 6 reviews sets and set like objects. The chapter begins with **multisets**, before discussing what happens to the union operator when sets are generalised to allow membership “grades” and/or “multiplicities”. **Merge** type union operators sum the grades or multiplicities of elements common to the merged “sets” (think “*merged* traffic”). **Join** type union operators take the maximum grade or multiplicity of elements common to the joined “sets” (think “*conjoined* twins”).

The chapter discusses **indexed families**, **rough sets**, **fuzzy sets** and **L -fuzzy sets**. L -fuzzy sets are the closest to what is required, but the notation, used in [42], does not fit requirements and a merge type union operator—which is required—is not defined. The chapter also defines **L -collections** and extends set operators over L -collections, where L is a **maximisable set**, including both **merge** and **join** type union operators. The chapter defines further operators that allow L -collections to be **scaled** and **cast**. It is shown how L -collections generalise multisets, fuzzy-sets and rough sets. The chapter discusses how proofs for theorems over sets can be extended to certain L -collections.

The definition of L -collections in this chapter will allow us to define information spaces that can cope with uncertain and partial classifications, and multiple, identical classifications.

Chapter 7 extends distance functions over sets of L -collections. This works well for the $d(\mathcal{X}, \mathcal{Y}) = |\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$ distance function (\mathcal{X} and \mathcal{Y} are L -collections), but the d_{ij}^M distance function is not $\mathcal{Q}_{ij}^{d^M}$ -strict positive over sets of L -collections. The chapter defines a new distance function ${}_k^M d$ and shows that ${}_1^M d$ and ${}_{av}^M d$ satisfy the generalised triangle inequality. ${}_k^M d$ preserves \subseteq -reflexivity while ${}_1^M d$ and ${}_{av}^M d$ are $\mathcal{Q}_1^M d$ -strict positive and $\mathcal{Q}_{av}^M d$ -

strict positive (respectively) if d is \mathbb{Z}^d -strict positive.

This chapter extends the mathematical basis from sets to L -collections.

Chapter 8 shows how useful distance functions can be built up over networks (where the nodes *may* have span). It is shown how n -dimensional and many levelled **classification spaces** can be defined. The chapter shows how, using $\{0.5, 1\}$ -collections and $(0, 1]$ -collections, classification spaces that allow uncertain and partial classifications can be defined. It is shown how, using \mathbb{N}_1 -collections, **indexes** can be defined that allow information units to be attached to points in classification spaces. The chapter defines **information space** as, essentially, a classification space with an index. It is shown how information space provides a mathematical basis for Knowledge Libraries. The chapter shows how the core functionalities, identified in chapter 3, can now be defined mathematically.

This chapter completes the definition of information space—a mathematical basis for Knowledge Libraries.

Chapter 9 gives two paper based examples that show how information space provides the desired functionality. The first example is a questionnaire information space, the second, a research paper information space. While these examples are necessarily simplistic—practical information spaces would be too complex for this treatment—they do show how the desirable functionality is provided.

This chapter gives examples of information space being used to provide important Knowledge Library functionality.

Chapter 10 discusses the implementation of the less easily implementable Knowledge Library functionality, in particular, distance query, range query, k nearest neighbour query and ranked query algorithms. Range query is discussed at length as k nearest neighbour query and ranked query algorithms are normally based on range query algorithms. Existing metric space algorithms are generalised to work over set spaces.

Chapter 11 presents the results of experiments conducted to determine the effectiveness of range query algorithms. A clearer understanding of the limitations of existing range query algorithms is developed, and a number of improvements are suggested. Spatial partitioning algorithms can often

be improved using a number of search trees and taking the intersection of the candidate set produced by each tree. The sequential-hybrid algorithm extends the scope of sequential search, as it can be applied to multi-dimensional spaces that have a large (or even infinite) number of dimensions.

Chapter 12 discusses what has been achieved, and what remains to be achieved. The chapter discusses the extent which information space generalises hierarchical filesystems, relational databases, OLAP data cubes, and information retrieval vector spaces.

1.2 The Contribution of this Thesis

The main thrust of this thesis is to provide the basis for Knowledge Libraries; a new class of information organisation system with considerably more power than existing systems. On the way to achieving this goal, a number of other significant contributions are made.

This research:

1. identifies the need for the development of a coherent “information organisation” discipline with computer filesystem, information retrieval, database and other sub disciplines. This new discipline would define common terminology and exploit synergies between these research areas;
2. identifies significant limitations of existing information organisation techniques;
3. identifies the need for the development of better information organisation techniques for scholarly research generally. Such techniques could dramatically increase the pace of scientific development;
4. describes “Knowledge Libraries”—an idealised system for the storage, organisation, and display of information;
5. defines “ L -collections” a generalisation of sets (also multisets, rough sets and fuzzy sets) that allows elements to have membership “grades”

chosen from the set L ;

6. defines “information space” and shows how information space can provide the mathematical basis for Knowledge Libraries;
7. provides effective range query algorithms for information space;
8. shows how Knowledge Libraries and information space generalise hierarchical filesystems, vector space information retrieval, relational databases, and OLAP datacubes.

1.3 The Organisation of Information

In this thesis, **information organisation** is defined as the topic concerned with:

1. the nature of information;
2. modelling information;
3. modelling the relations between “information units”;
4. identifying desirable functionality of information systems;
5. the development of information systems;
6. the development of interfaces to information systems;
7. testing information systems;
8. assessing the value of information systems.

Note that “information system” is taken to be synonymous with “information organisation system”.

Although not currently well recognised in the literature, it should be clear that the subtopics that make up the topic of information organisation are closely related—the proper investigation of one subtopic requires some familiarity with associated topics.

Example 1.1. When developing an information system—so that the way information is represented within the system adequately reflects reality—it is necessary to consider the nature and modelling of information. In order to organise information, it is necessary to have some representation of the relations between the information units in the system. The system interface must be designed so that system functionality is appropriately available to users. As part of the demonstration of the value of the system, it is necessary to test it to determine if it provides appropriate functionality.

Note that, while all the subtopics listed are important for the development of the topic of information organisation, the subtopics are not developing evenly. In particular, the question of precisely *what information is* does not appear to be adequately addressed in the literature. Similarly, while it seems indisputable that information systems—such as the world wide web—are valuable, the author is not aware of the existence of formal methods that measure the value of such systems with precision.

This last point is especially significant as it makes it hard to determine the value of improved information systems. This is unfortunate as such research can have considerable social and economic value. If the value of information systems—and hence research that results in the development of improved information systems—was more readily measurable, more research would be conducted.

A distinction is made between “weak” and “strong” classification.

In **weak classification**, given some collection of information, the information is classified to expedite the retrieval information. In this thesis this weak form of classification is called *indexing*.

In **strong classification**, the classification of information specifies the relative “position” of the information in a “universe of knowledge.” Within the classification, the information is “close to” related material and “far from” unrelated material. In this way classification:

1. links information with other information;
2. provides “pathways” to information;

3. summarises information.

Given this description, the reader should appreciate the close relationship between strong classification and other important information organisation terms such as *abstract*, *background*, *introduction*, *meta information*, *outline*, *overview* and *summary*. Strong classification can provide a formal equivalent, and perhaps ultimately a replacement, for each of these.

1.4 Thesis Methodology

This thesis follows the “gap–hypothesis–experiment” research pattern. Research following this pattern begins with the identification of a “gap in the literature.” The research goes on to formulate a “research hypothesis” which, if supported by the experimental results, will contribute to filling that gap. After this, experiments are designed and conducted to test the hypothesis. Finally, the results are summarised, reported and analysed.

A similar and parallel research pattern would be to begin by identifying a “research problem,” decide on an approach to “attack” the problem, execute this plan, then summarise, report and analyse the results. It should be apparent to the reader that the “gap–hypothesis–experiment” research pattern can be readily mapped onto this more general pattern.

Gap in the literature. *The information organisation technology described in the literature—such as relational databases and information retrieval—does not provide important functionality.*

Research hypothesis. *Information space provides a mathematical basis for the development of Knowledge Libraries that provide this functionality.*

Experiment. *Attempt to develop a mathematical definition for information space and show how this can be used as the mathematical basis for Knowledge Libraries.*

1.4.1 Gap in the literature

The fragmentation evident in the information organisation literature is itself evidence of the lack of important functionality in existing information organisation technology. If better technology were available, it would be expected to result in less fragmented literature.

More specifically, the “similarities” between “queries” and “weighted keyword unit vectors”, in the vector space model for information retrieval does not always accurately reflect the similarities between the underlying “user information requests” and “documents”.

This research does not attempt to formally identify “important functionality”. Perhaps the best way to establish the importance of new functionality involves:

1. an initial “guess” at identifying important new functionality;
2. the development of a system with the functionality;
3. a demonstration of the added value of the new system over pre-existing systems without the extra functionality.

This last step is not always easy to achieve. Surely the added functionality of well established information organisation systems such as relational databases, the vector space information retrieval, the internet, and the world wide web add considerable value—but how can this added value be quantified?

In this research, important functionality is determined by describing some of the many advantages of an imagined system with this functionality. Even if the reader disagrees with some of the details, the approach to organising information by association with points in an information space—where the distances reflect the underlying semantics of the information—has clear utility.

1.4.2 Research hypothesis

This research relies heavily on the metaphor of information space to describe Knowledge Libraries, so it should be clear that information space is required in order to implement Knowledge Libraries.

It is not immediately apparent that:

1. it is possible to define information space flexibly enough so that the metaphor remains useful for a great variety of different forms of information;
2. information space provides a mathematical basis for the core Knowledge Library functionality identified in this thesis;
3. information space provides a practically implementable basis for Knowledge Libraries.

It *is* apparent that naive “information spaces”, such as n -dimensional real coordinate space, are not suited to the organisation of many forms of information. Similarly, readers might wonder if information space provides a sufficient basis to define the functions required to provide the core Knowledge Library functionality. Readers might also doubt that these functions are practically implementable. Perhaps the time or memory complexity of some of these functions render implementation impractical?

1.4.3 Experiment

The experiment involves:

1. the informal development of Knowledge Libraries—an idealised system for organising and presenting information;
2. the development and definition of information space—a mathematical basis for Knowledge Libraries;
3. demonstrating that information space provides an adequate basis for Knowledge Libraries.

1.5 How to Read this Thesis

This thesis investigates the organisation of information. The question of how best to organise and present research provides a key motivation for the research described herein. Presently, many universities¹ require the submission and examination of theses in “printed paper” form. Due to the many advantages of digital formats, it may be that this requirement is dropped in the near future.

One limitation of printed paper theses is that they necessarily present information in a linear sequence, following the order of the pages. The greater flexibility of digital formats could be used to advantage to present information as a “network.” This better reflects the underlying nature of information, as discussed briefly in section 1.7 and more completely in the remainder of this thesis.

The author acknowledges that readers will not begin with the first page of this thesis, then continue, reading each page in sequence until the last. Rather, it is expected that readers are more likely to, for example, glance briefly through the contents, largely skip over the thesis overview, read a number of sections in the introduction in sequence, review the contents pages again, skip ahead to review a chapter of interest, go back to the introduction and continue reading where they left off, skip head to Appendix C to review thesis terms,

Each chapter of this thesis, after this introductory chapter, begins with an overview of chapter contents. Each chapter, apart from the first and last chapter, finishes with a summary of what was achieved in the chapter, with respect to the central argument of the thesis.

It is noted that the overviews are *not* summaries. If the reader wishes to read a chapter summary before reading the chapter, he or she should refer to the summary at the end of the chapter. The intent of the chapter overviews is to provide a type of “narrative contents,” largely for readers already passingly familiar with the contents of the chapter. The overviews provide “links” to chapter content, in a similar manner to the links in web

¹Including the University of Wollongong.

pages (or many other digital documents).

1.6 Thesis Outline

The central argument of this thesis has three main steps.

In the first step, chapter 3 informally describes Knowledge Libraries by way of a number of user scenarios. Spatial terms such as “map”, “dimension”, “coordinate”, “region” and “space” are important when describing the “user experience” of Knowledge Libraries. This suggests that an appropriate definition of “information space” could provide a mathematical basis for Knowledge Libraries.

In the second step, chapters 4-8 provide a mathematical definition of information space. This requires mathematical definitions of, among other things, **dimensions**, **coordinates**, **points** and **spaces**. In mathematics, the term “space” has distinct meanings when discussing **metric space**, **vector space** and **topological space**—as in section 4.2. The definition of **space**² in this thesis is quite general, yet it is *not* consistent with all of the uses of the term in mathematics.

In the third step, chapters 10-11 show how existing metric space algorithms can be improved and generalised to work over information space. Other techniques, that work where the generalised metric space algorithm is ineffective, are also developed.

In each of these three steps, terms are used with the same intent, but with subtly different meaning, as they are used in other steps. In the first step, it is found that spatial metaphors are useful to describe the capabilities of Knowledge Libraries. “Points in space” here is simply a useful metaphor. In the second step “points” are mathematical objects. In the third step, a “point” is a data structure. It should be clear to readers that, for example, a “point” data structure is not precisely the same thing as the mathematical definition of “point”.

The most informal use of terms like “point” is in chapter 3, where these

²A set with a (signed) distance function, see section 4.2.1.

terms describe the “user experience” of Knowledge Libraries. The next level is the mathematical definition, chapters 3-8. The most formal level is in the computer implementation of the algorithms and associated data structures from chapter 10. Here a “point” is a data structure. So the initial informal description of Knowledge Libraries provides the motivation for the development of mathematical definitions, which in turn provide a framework for the development of algorithms and data structures.

1.7 The Nature of Information

The concept of “information” is one that, it seems, everyone *thinks* they have a good understanding of. Yet it is very difficult to translate this “understanding” into a formal definition. A dictionary, for example, presumably contains information. In particular, it contains the “meaning” of words (a type of information?) However when we attempt look up the meaning of a word in a dictionary, we are simply referred to more words. This trail of words is a complex forking one—a network, or more formally a graph—entirely made up of words. The trail is essentially circular—following it, we will never discover the “underlying information” we seek. Does this then imply that the graph *is* the information and no more fundamental “information atom” exists? Surely though, if words are only ever defined in terms of other words, they are ultimately meaningless.

Babies develop social and language skills based on their own direct experience of the world (their senses) and their inner emotional state. They begin by making associations between the facial expressions of their carers and these experiences. This develops into an understanding of “body language”, gestures and basic tones. Comprehension of simple words, such as “yes” and “no”, is also developed in this way. More complex words and sentences are “built-up” from this basic language. This suggests that the fundamental “information atoms” that we are looking for may be basic “sense impressions” and emotions.

Mathematicians and Logicians, who are nevertheless obliged to communicate in words with meanings developed in this way, reject this form of

“knowledge” as it is known that the form of induction³ used to develop this knowledge is logically unsound.

Mathematicians and Logicians prefer to develop knowledge by applying logical inference to a set of basic and “self evident” axioms. The knowledge developed in this way is as reliable as the correctness of the axioms and the validity of the logical reasoning. From the mathematical and logical point of view then, these axioms would be the required “information atoms.”

Scientists, employing the “scientific method” may have more in common with babies than Mathematicians and Logicians! The scientific method involves systematic observation, measurement, and experiment, and the formulation, testing, and modification of hypotheses. In broad terms, the scientific method formalises (to a degree) the very same process used by babies to develop social and language skills. The scientific method relies on the same form of induction and so the “scientific knowledge” generated is similarly vulnerable. It may be that many observations and experiments support a hypothesis or theory, yet a single exceptional result refutes it.

This thesis does not attempt to formally define “information,” nor the “information atom.” Similarly, no attempt is made to formally define “knowledge.” Nor is any existing definition adopted. One reason for this is that no general consensus exists as to the precise meaning of these words. The formal definitions that do exist are only suitable for use within specialised fields and tend to contradict other formalisations, used in other fields.

Example 1.2. Within the field of “expert systems”, “knowledge” is encoded in a formal language in “knowledge bases” (on a computer). In contrast, in the field of “knowledge management” and many other disciplines, knowledge necessarily resides in individuals (human beings).

The difficulty in developing broad formal definitions, and so assigning precise meaning, to common information related terms such as “information”, “knowledge”, “understanding”, “meaning” and “intellect” will likely surprise many researchers. Even the term “science” is not well defined, lead-

³That is, the development of general rules from repeated specific observations.

ing to ambiguity and confusion as to precisely what qualifies as a science⁴. Indeed, while most researchers—early on in their careers—develop quite precise ideas about the use of terms such as “introduction”, “abstract”, “summary”, “outline”, “overview”, “background” and the best way to structure a research paper or thesis, no formal definitions exist, and there is surprisingly little consensus between researchers.

Example 1.3. In the view of at least one experienced researcher (an examiner of this thesis), an introduction should summarise three things: the research problem that is being investigated; the approach that was taken to attack the problem; and the results and their significance. In contrast, for another seasoned researcher (the supervisor of this thesis) an “introduction” “introduces” readers to the material and definately should *not* summarise results. According to this view, the sections or chapters mistakenly called “introductions” by many (perhaps even the majority) of researchers, are more correctly called “executive overviews.”

The above example highlights the fact that, while many researchers may think that they have a precise understanding of what constitutes an “introduction,” no such consensus exists. With this in mind, it is appropriate for researchers (including supervisors and examiners) to relax strict views and adopt a more utilitarian stance.

While it is *not* appropriate for this thesis to adopt formal definitions for terms like “information” and “knowledge,” it is reasonable to make some general observations. There appears to be a rough hierarchy of “information related” terms (“information nodes”). “Raw data” is commonly placed at the bottom of this hierarchy. “Information” is placed further up, and terms such as “knowledge,” “meta information,” “understanding,” and “meaning” still further up. In general, the content of nodes towards the top of the hierarchy is more interconnected and contextualised, and is more valuable—especially if it is reliable, accessible, and unambiguous. Unfortunately the increasing

⁴Note that a formal definition of a class can be used as a test to determine class membership.

complexity towards the top of the hierarchy contributes towards ambiguous content of limited reliability and accessibility!

It would appear that human understanding is a top-heavy structure. By improving the precision, reliability and accessibility of complex interconnected and contextualised information, information organisation systems should extend the reach of human understanding.

In this thesis the term “information space” is used both in a general sense, to describe the “universe of information” and, more specifically, as the name of a mathematical model of this information universe. The term “Knowledge library” is used to describe an information organisation system. In this term the word “knowledge” is used to reflect that the system is useful for managing information, and its interconnections and context. The choice of this term, from amongst the terms towards the top of the “information hierarchy,” is somewhat arbitrary.

1.8 Limitations of this Thesis

This thesis does not commit to formal definitions of information and knowledge. While this results in the development of a broadly applicable system—the Knowledge Library—it does limit the degree to which the results herein can be formally appraised.

Because of the very broad subject matter addressed in this thesis⁵ and the great many closely related topics, it has not been possible to fully explore the relevance of all related research. The relevance of many topics, such as artificial intelligence and neuroscience for example, has not been explored at all. This is undoubtedly a serious limitation of this study. Unfortunately, it has proven to be humanly impossible for a single researcher, in a finite time

⁵The non-mathematical literature review (see chapter 2) identifies traditional classification, faceted classification, computer filesystems, database systems, information retrieval systems, data warehousing, the Internet, the World Wide Web, and the Semantic Web as closely related topics. The mathematical literature review (see the initial sections in chapters 4 and 6) identifies metric space, vector space, the vector model for information retrieval, lattices, topological space, formal concept analysis, the relational model, multisets, indexed families, rough sets and L -fuzzy sets as important related mathematical objects.

period, in a thesis of finite length, to fully explore all the potentially relevant literature. The literature review in this thesis however, does provide a solid, extensive and detailed—if not exhaustive—foundation for this research.

Note that this thesis *does* consider, and provide for, the dynamic, changeable nature of information. See, in particular, section 2.3.3, scenarios 1, 1.1, and 1.2 in section 3.3, the definition of **add_node** in section 8.3, and section 8.8.2.

1.9 Knowledge Libraries

The value of information is becoming ever more apparent. Clearly, we (the users of information systems) need the right information at the right time in order to make the right decisions. Human society has responded to this need by developing ever more sophisticated techniques to gather information. Computer memory and processing speeds have also been increasing at an exponential rate in recent years. In contrast, the key research[24, 56, 79] that underpins our main information organisation systems is many decades old.

Critically, information retrieval systems do not adequately model the relationships between information. As a result, they do not always allow access to the right information. These systems also fail to inform users of the relationships that exist between their “information units”, nor do they provide meaningful overviews of their organisational structure.

Knowledge Libraries are idealised information organisation systems. A Knowledge Library provides precisely the information organisation functionality we require (whether we know it or not) based on the assumption of an accurate model of the relationships between the information elements in the library. They allow us to develop an overview of the manner in which the information is organised.

Knowledge Libraries rely on a spatial metaphor for organising information. By developing a mathematical definition of this “information space” this research is able to provide a mathematical basis for Knowledge Libraries. This thesis distinguishes between “core” and “extended” functionality. Core

functionality supports essential Knowledge Library features such as determining the distance between points, and selecting information elements. The extended functionality can be loosely described as a collection of “convenience features”.

This research relates the the problem of the implementation of core Knowledge Library functionality to the existing literature on searching metric spaces. It is discussed how, by generalising the existing metric space algorithms to work over set spaces, important core Knowledge Library functionality is provided. This thesis generalises the existing algorithms in precisely this way, and develops more effective algorithms.

1.10 Information Space

This thesis attempts to define “spaces” that are flexible enough to model the relationships between “information units”—research papers, web sites, math equations, digital photographs, quotations, questionnaires algorithms, numbers, etc. These relationships are modelled by “distances”—the smaller the distance, the closer the relationship.

It is found that there are often many categories of relationship that can exist between information units.

Example 1.4. Consider a space of research papers. We might want to compare and contrast research papers in the space by subject, authorship, title, length, publication date, “quality”, citations, ...

The approach of this research to this is to assign each relationship category a “dimension”. So if it is found that there are n different relationship categories for a certain type of information unit, an n -dimensional space is required to model these relationships.

Example 1.5. Consider an n -dimensional real coordinate space. Each dimension consists of a set \mathbb{R} and a “distance function” $|x - y|$ for all $x, y \in \mathbb{R}$. The n -dimensional space consists of the set of n -tuples of reals $\mathbb{R} \times \dots \times \mathbb{R}$

with the distance function

$$d((x_1, \dots, x_n), (y_1, \dots, y_n)) = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

for all $(x_1, \dots, x_n), (y_1, \dots, y_n) \in \mathbb{R} \times \dots \times \mathbb{R}$.

It will become apparent that real coordinate spaces, such as the one in the example, are *not* general enough for our purposes.

From the example, it is tempting to define “space” as a set of “points” M with a distance function $d : M \times M \rightarrow \mathbb{R}^{\geq 0}$. It can be seen that each dimension of an n -dimensional space is itself a space. It is possible to combine n spaces $\langle M_1, d_1 \rangle, \dots, \langle M_n, d_n \rangle$ into an n -dimensional space $\langle M, d \rangle$ where $M = M_1 \times \dots \times M_n$ and

$$d((x_1, \dots, x_n), (y_1, \dots, y_n)) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

for all $(x_1, \dots, x_n), (y_1, \dots, y_n) \in M$ and for some $p \in \mathbb{R}$. When a space $\langle M_i, d_i \rangle$ is used as a dimension of another space $\langle M, d \rangle$, the points of $\langle M_i, d_i \rangle$ become “coordinates” of $\langle M, d \rangle$.

As it happens, this is *not quite* general enough. It will be seen that L -collections—that allow duplicate and “graded” elements—rather than just sets, are required. Also, sometimes “signed distance functions” are useful so, for example, $d(x, y) = -d(y, x)$. All signed distances $d(x, y)$ are associated with a corresponding unsigned distance $|d(x, y)|$.

In order to simplify this presentation, this introductory discussion continues as if these further generalisations were not required. The deficiencies of—and necessary adjustments to—this discussion will be made apparent when the more advanced material is introduced.

It is often desirable, given distances between information unit attributes, to define distances between sets of information unit attributes.

Example 1.6. We have a set of authors M with a distance function d where $d(x, x) = 0$ and $d(x, y) = 1$ for all $x, y \in X$ where $x \neq y$. But because research papers can have a number of authors, we require a distance function d' that gives the distance between sets of authors $X, Y \subseteq M$. $d'(X, Y)$ is defined in terms of the distances $d(x, y)$ between the elements x of X and the elements y of Y .

It is often the case that spaces that model the relationships between one type of information unit (in the example, “authors”) can be the basis for spaces that model the relationships between another type of information unit (in the example “research papers”). Similarly, spaces that model the relationships between one type of information unit can be dimensions of spaces that model the relationships between another type of information unit. Both of these types of “nesting” can go on for any number of “levels”.

Adequately flexible spaces may have points that are sets or n -tuples or simple elements. The elements of sets or n -tuples can themselves be sets or n -tuples or simple elements... and so on. The critical question is:

“What are the desirable properties of these spaces?”

If the desirable properties of these spaces can be identified, and it can be shown that a particular space has these properties, then that space can be expected to function as preferred. If appropriate spaces with these properties exist, then this research can claim a certain measure of success.

This research is limited to spaces that *positively* represent information unit attributes—that is, spaces where points represent attributes⁶ of attached information units. Spaces that may *negatively* represent information unit attributes do not have the same desirable properties.

This research focuses on the properties of distance functions that give distances $d(X, Y)$ between points X and Y where

1. X represents a “query”—the information unit attributes that are desired, and Y represents the attributes of attached information units;

⁶Or sets or n -tuples of attributes. Or sets or n -tuples of sets or n -tuples of attributes...

2. both X and Y represent the attributes of attached information units;
3. both X and Y represent the attributes of *sets of* attached information units.

In general, this research finds that the metric properties⁷ are not sufficiently flexible for set space distance functions.

If X represents desirable attributes, we want $d(X, Y) = 0$ whenever Y “satisfies” X —that is, whenever Y represents all the attributes X (and perhaps more). We also want $d(X, Y) > 0$ whenever Y “fails to satisfy” X —that is, whenever Y *does not* represent all the attributes X .

Example 1.7. Given an information space of mathematical monographs we might have a set X of requirements and a set Y , representing a monograph:

$X = \{\text{“about: set theory”, “level: introductory”, “pages: 20 or less”}\}.$

$Y = \{\text{“about: set theory”, “level: introductory”, “pages: 18”, “author(s): Smith”, “title: An introduction to set theory”}\}.$

We want $d(X, Y) = 0$, even though $X \neq Y$. Reversing, if Y was a set of requirements and X a set, representing a monograph, we want $d(Y, X) > 0$, so $d(X, Y) \neq d(Y, X)$.

From this it is apparent that reflexivity is too weak, and strict positiveness is too strong.

If the points in a space are sets⁸ it is found that, because it does not take the “span” of these sets into account, the triangle inequality is too restrictive.

Section 4.9 defines suitable properties that correspond to metric properties.

1. \subseteq -reflexivity—a stronger property corresponding to reflexivity.
2. $\not\subseteq^d$ -strict positive—a weaker property corresponding to strict positiveness.

⁷Reflexivity, strict positiveness, symmetry, and satisfying the triangle inequality.

⁸Or n -tuples of sets...

3. the (span) generalised triangle inequality—a more general inequality corresponding to the triangle inequality.

It is often useful to be able to “base” distance functions on other distance functions so that spaces can be nested inside one another. Flexibility is also desirable—for example, it is often useful to adjust the relative significance of different dimensions.

This thesis defines a number of suitable distance functions. Theorems and worked examples demonstrate that these functions have the desirable properties.

If the points in a space have spans (real numbers) then the information units attached to these points are associated with these spans. Relatively general information units and information units containing information about disparate topics are attached to points with greater spans than more specific information units. So the span associated with an information unit can be a useful indicator of the “breadth” of the information.

1.11 Information Organisation Terms

A detailed glossary of terms used in this thesis has been prepared (see Appendix C). This glossary has a dual purpose. First, it describes relevant objects or concepts and so provides a ready reference for this thesis. Second, it describes some important terms in the “language of information organisation”. As such, it provides a useful introduction to this thesis.

Chapter 2

The Organisation of Information

2.1 Overview

This chapter reviews the literature surrounding the topic *the organisation of information*.

As is discussed in section 1.6, the literature does not provide broad formal definitions of information, so fails to adequately answer the question “what is information?” Similarly, while the literature does develop numerous systems to organise information, many of which appear to promote understanding, these systems do not appear to be based on any theoretical study which addresses the question “how should information be organised to promote understanding?”

Section 2.3 reviews **traditional classification**. **Faceted classification** solves a number of problems and remains somewhat underdeveloped. It may be that “classification” and “information” are not really distinct concepts.

Section 2.4 discusses **computer filesystems**. A computer filesystem organises computer files, which can encode a great variety of information. Computer filesystems are generally hierarchical. Often additional search tools are required.

Section 2.5 focuses on the **database**. It appears that computer filesystems-

tems can be treated as a type of database.

Section 2.6 reviews **information retrieval**. The retrieval of information using an **information retrieval system**—generally relies on **polythetic classification**, while data retrieval—that is, the retrieval of information using a **database management system**—generally relies on **monothetic classification**.

Section 2.7 discusses **data warehousing**. Data warehousing increasingly involves using a number of **dimensional data marts** to provide specialised business data analysis.

Section 2.8 reviews the **Internet**, the **World Wide Web** and the **Semantic Web**. Both the internet and the world wide web have been successful in communicating huge amounts of information. Our assessment is that the semantic web, while attempting to address important issues, tries to solve too many problems at once—without the benefit of a clear and precise understanding of what is being attempted.

Section 2.9 discusses and summarises these reviews. Research efforts into the problem of the organisation of information have been isolated and fragmented and have failed to identify the “big picture”. The research community must respond to the information explosion with a thorough investigation of the problem of the organisation of information.

Section 2.10 outlines what has been achieved in this chapter. The chapter surveys and reviews the literature relating to the topic of the organisation of information. The literature appears to be somewhat fragmented, lacking a common theoretical basis.

2.2 Introduction

As is discussed in section 1.7, the literature does not provide broad formal definitions of information, so fails to adequately answer the question “what is information?” Similarly, while the literature does develop numerous systems to organise information, many of which appear to promote understanding, these systems do not appear to be based on any theoretical study which addresses the question “how should information be organised to promote un-

derstanding?” As will be seen from the literature review in this chapter the relevant literature is generally more focused and is centred around the discussion, modelling and analysis of practical applications that seek to provide a solution to specialised “information organisation” problems.

2.3 Traditional Classification

The term “classification” has been described as “[a] division or category within a system... [89]” Another meaning is “[the] process of putting things into groups according to similarities or relationships[89].” “Classification” is also used as an abbreviation for “classification system.”¹

The purpose of classification has been described as

...to show relatedness or association between documents. Association between documents may be shown by placing them in the same class or it may be exhibited by showing relatedness between the two classes containing those documents [93] (p130).

From its earliest beginnings, the science of classification has been intertwined with the development of human thought.

The history of classification is almost necessarily a history of the attempts to organise human thought. Since man began his long endeavours to distinguish and understand the parts of his universe, he has consciously or unconsciously formed some system in which those parts were related to one another [93] (quoting W.C. Berwick Sayers)

In the 18th century, Carolus Linnaeus created the **tree of life**—a hierarchical classification scheme of life: kingdom, family, class, order, family, genus and species. His classification scheme was based on how organisms looked (their phenotype). In the mid-19th century Ernst Haeckel (a contemporary of Charles Darwin) used the metaphor of a tree to describe the evolutionary relationships among living organisms (a phylogeny) [61].

¹Classification has also been used to refer to “...the process of identifying that information which requires protection in the interests of preserving national security [69].” This is *not* how the term is used in this thesis!

Perhaps inspired by the tree of life, Melvil Dewey developed the **Dewey Decimal Classification**, abbreviated DDC, scheme (*circa* 1875). One of the main advances of DDC was that it specified a *relative* ordering of books (to one another).

Before DDC, it was standard practice for libraries to organise their collections by assigning each book a particular location on their shelves (an *absolute* ordering system) [81]. This system could not readily accommodate changes to the library's collection such as the purchase of new books, or the retirement of old books.

In 1996:

... [DDC] is used in 135 countries and translated into more than thirty languages. It is certainly the classification system best known by most library users in the United States. Ninety-five percent of all U.S. public and school libraries and a quarter of all college and university libraries use this system. [81]

The DDC system was the basis for the **Universal Decimal Classification**, abbreviated UDC, scheme (*circa* 1900) developed by Paul Otlet and Henri La Fontaine. The UDC builds on DDC, making use of the concept of a “universe of information”, and including auxiliary signs (e.g. +, /, :) to express relations of various kinds between two (or more) subjects. The UDC was nurtured, since its conception, by the International Federation for information and Documentation (FID), which was called the Institut International de Bibliographie (IIB) until 1938 [64]. In 1992 the UDC Consortium (UDCC) took over from FID, assuming ownership of UDC [96].

The UDC scheme appears to be a faceted classification system, although it predates the notion of faceted classification (attributed to Ranganathan *circa* 1930, see section 2.3.1)!

In Australia, the CSIRO uses UDC to index their library (see [2]), although to date no attempt has been made to use the faceted properties of UDC to create a “multi-faceted” search tool.

The **Library of Congress** developed its own classification system, abbreviated LCC, around 1900. LCC was clearly strongly influenced by DDC,

one main difference being that main classes in LCC are represented by letters rather than decimal numbers [81]².

The other main difference between LCC and DDC is that LCC simply attempts to classify the books appearing in the library of congress, while DDC is an attempt to classify all knowledge.

2.3.1 Faceted classification

The DDC is largely an **enumerative classification**—essentially a list of classes. It is also a **top-down classification**—the root of the classification hierarchy corresponds to the entire universe, with ever smaller subdivisions towards the leaves of the tree.

Another approach is to *synthesise* classes by joining together a number of separate concepts. This is called **analytico-synthetic classification**. In **faceted classification**, subject matter is organised by first identifying a number of *fundamental categories*, also known as *facets*. Key terms are grouped into facets and—for complex subjects—may themselves be hierarchically organised (within each facet). “Building-up” classes from component terms in this way is called **bottom-up classification**. [16]

Example 2.1. We want to develop a faceted classification scheme to classify socks. The facets *colour*, *pattern*, *yarn*, *purpose*, and *length* seem suitable. Colour terms include *black*, *grey*, *brown*, *green*, ... Pattern terms include *plain*, *striped*, *spotted*, *checkered*, ... etc. Classes are formed by selecting a term from each facet. In this way we have ready made classes for ‘green, checkered, cotton, work, calf-length’ socks and ‘grey, plain, wool, hiking, knee-length’ socks etc.

Note that an enumerative form of the classification scheme described in the example would be far longer and require substantial repetition.

²In [70] Palmer notes with suspicion that DDC, (using decimal notation) managed to find just ten main classes of knowledge, while LCC and Colon classification, using letters to represent different classes of knowledge, found a number of main classes that closely matched the number available letters.

Faceted classification is attributed to Ranganathan through his “colon classification” scheme, which he began work on in 1924 [93, 13]:

The presence of books with multi-faceted subjects was a fact [...] Library classification should reckon with them [93] (quoting Ranganathan).

The concept of faceted classification has influenced the development of other major classification systems. As already noted, although it predates faceted classification by some 30 years, Universal Decimal Classification [96] is close to being simply a faceted version of Dewey Decimal Classification. Bibliographic Classification 2, abbreviated BC2, is a faceted version of Bliss’s Bibliographic Classification[14].

It is a general failing of faceted classification schemes that they seem to be more about determining an appropriate classification for books (knowledge) than providing an effective index:

It must be realised that synthesis of notation is a device to help the classifier, not the user. It enables the classifier to specify composite subjects which may not have been foreseen by the classificationist (*i.e.* the compiler of the scheme) but which nevertheless arise in documents. [38] (p16).

2.3.2 ISBN, ISSN, MARC and CIP

The **International Standard Book Number System**, abbreviated ISBN, was developed to allow book publishers and distributors to use computers to automate inventory control and order processing. It was introduced in the United Kingdom in 1967 and approved as ISO standard 2108 in 1970. In 1984 the ISBN system was extended to cover computer software.

In January 1, 2007 ISBN numbers changed format from 10 digits to 13 digits as numbers were running out. The new format numbers will include a three digit prefix that identifies the industry. This will make ISBN numbers identical with machine readable EAN-13 barcodes. ISBNs will initially be prefixed with 978. As existing numbers are used up, ISBNs will be assigned with the prefix 979.

The new format ISBN number is divided into five parts of variable length, separated by hyphens or spaces. The five parts are: the three digit industry prefix, the group identifier (identifying a country, area, or language; up to five digits long), the publisher identifier (smaller for larger publishers), the title identifier (identifies a specific edition; up to six digits long; may be preceded by space filling zeros), and the check digit [41].

Note that the ISBN does not attempt classify the “content” of publications.



Figure 2.1: A 13-digit ISBN with EAN-13 bar code

The **International Standard Serial Numbering System**, abbreviated ISSN, has a completely different format and administrative body to the ISBN. While ISBN numbers are intended to identify monographs, ISSN numbers may be assigned to an entire series of monographs, where each individual monograph in the series has its own ISBN.

An ISSN number consists of seven digits (assigned free of charge by the ISSN International Centre), followed by a check digit (which may also be an upper case ‘X’), and preceded by the letters ISSN. The eight digits that make up the ISSN number are separated into two groups of four by a hyphen [3].

Note that, just like the ISBN, the ISSN does not attempt classify the content of publications.

The library of congress **cataloguing-in-publication data**, abbreviated CIP, is a condensed version of the **machine readable record**, abbreviated MARC. The purpose of the CIP is to provide libraries with detailed cataloguing information in a standard format along with each book. Participating publishers provide the LOC with the full text of material nearing publica-

tion, and (in theory) receive a catalogue record (CIP data) generated by the LOC in time for publication, which they print unaltered in the title page of the published book [68].

The full MARC contains additional information and is available on the LOC online catalogue. It is distributed in electronic form to libraries, publishers, and suppliers. MARCs are organised into bibliographic, authority, holding, classification, and community information format blocks. The bibliographic block includes author, title and (multiple) subject classification fields as well as physical (number of pages, size, binding) property information. Individual fields can be decoded according to a complex system involving information contained in leaders, directories, three digit tags, and terminating characters[47].

2.3.3 A critique of traditional classification

Classification schemes impose a structure on knowledge.

It must be clearly borne in mind... that the classification of knowledge should be the basis of the classification of books: that the latter obeys in general the same laws, follows the same sequence... A book classification... is a classification of knowledge... [13] (preliminary pages, quoting W.C. Berwick Sayers).

...in general the closer a classification can get to the true order of the sciences and the closer it can keep to it, the better the system will be and the longer it will last [13] (preliminary pages, quoting Ernest Cushing Richardson).

Because human knowledge is imperfect, and our understanding of the universe is constantly evolving, the way knowledge is organised in classification schemes needs to evolve along with human understanding. Yet, to be considered useful and reliable, classification schemes need to remain constant over time.

Classificationists have dealt with these conflicting requirements by using hierarchical classes. As human understanding of a subject area evolves, further levels are added to the hierarchy. Class numbers for these new levels

consist of the class number for the base class, with added symbols assigned to differentiate between the new classes.

In this way the general structure of the classification scheme is preserved and the existing class numbers do not change. As the class numbers provide “access paths” to books, this *access path consistency* is reassuring to users.

Hierarchical classification schemes however do have their shortcomings. One limitation is that they only provide a single organisational perspective. For example, Dewey’s subject classifications are of little use if one is attempting to locate a book with a particular title, or by a certain author. It is also often inconvenient that each book can only be assigned a single subject as books are often “about” multiple subjects.

Fortunately faceted classification can solve these problems.

Another problem common to both faceted and hierarchical classification schemes occurs when subjects develop unevenly. When this happens, subjects at a low level of the hierarchy (with long class numbers) are further expanded (creating even longer class numbers) while subjects nearer the top of the hierarchy (with shorter class numbers) remain undeveloped. Ultimately, these imbalances provoke structural changes to the class hierarchy. When this occurs, class numbers need to be reissued. This *access path inconsistency* is undesirable³.

Sometimes weak classification systems need to be augmented by providing further classifying information within the classified objects themselves. Conversely, especially effective classification systems may be able to represent information traditionally contained within classified objects.

In fact, with a richly descriptive classification system... the information in the classification system becomes a valuable resource in its own right. In many cases one will find that information that was previously provided in the classified documents, such as the structure of the organisation, information about project participation and ownerships, etc. is duplicated in the classification system [40].

³The *dynamic* dimensions, discussed in section 3.3 scenario 1, may suffer from *access path inconsistency*.

It is noteworthy to observe that *all scientific research is, in a sense, also classification research*. For example, research papers define and relate ideas in the text of the paper. Research papers also describe and define relationships between other research papers through citations and the reference list.

It may be that if precise and universal definitions of “classification” and “information” existed, the distinction between classification and information would not be clear-cut. To illustrate, **data** can be described as *information minus context*, **information** as *data plus context* and **classification** as *information plus context*.

Following this nomenclature, a set of numbers—without any context to indicate what the numbers mean—would be data. A table of numbers plus a description of what the numbers signify, some analysis and an interpretation, would be information. Classification adds further context to information—context that relates that information to other information. But information already includes context!

Non-fiction books are associated with titles, authors, call numbers, publishers, publication dates, ISBNs, CIPs, languages, subjects, contents, glossaries, indexes, references, chapters, sections, subsections,... All of this information is “in the book”. Some of this information might be part of a particular classification of a book—precisely how much is somewhat arbitrary.

2.4 Computer File Systems

Computers read and write digitised information to and from a variety of media. At the most basic level, reading a block of data from a disk requires that the disk drive is operational and properly connected to the computer, the disk is rotating at the correct speed and the read-head is positioned over the correct sector and track. When these conditions are satisfied the read-head is activated and information is read off the disk until the end of the block. If the operating system implements multiprogramming—as modern operating systems do—it is also necessary to check that another process is not currently writing to that particular block. If the operating system implements some

from of data security it is necessary to check that the user has clearance to access the data.

The encoding scheme and data type must also be recognised before digitised data can be put to use—should the sequence of 0s and 1s in the data block be interpreted as a number, a sequence of numbers, an image, text,... etc.

The concept of the computer **file** is an abstraction—an interface that allows access to information without the need to explicitly consider details such as the data’s location on disk, block length, process blocking, security, encoding, ... etc. Files are identified by *name*. Many operating systems support two-part file names with each part separated by a period. The second part, called the *file extension*, indicates the *type* of the file. For example the file extension “.txt” normally indicates that “letter.txt” is a text file.

The file abstraction suggests that files “contain” data. The operating system achieves this helpful deception by mapping the file name to a block (or blocks⁴) of data on disk (or other media).

There are numerous different ways information can be “packed” into files. *Pile files*, for example, contain variable length records. Records may have different fields, or similar fields in different orders. In *index sequential files* all records are the same length and consist of the same number of fixed-length fields in a particular order. One field—usually the first in each record—is called the *key field*. The key field uniquely identifies each record. Records in index sequential files are ordered by their key field value.

Files have a number of attributes such as file name, address, creator, owner, creation time, time of last access, current size, ... etc. A number of different file systems—each specifying its own list of file attributes—are currently in use.

User demand for a file organisation system led to the development of the concept of the **directory**. A directory is a special type of file that can “contain” other files. Directories are sequential files where each record is a

⁴Many operating systems support file *segmentation*—where a single (variable sized) file is mapped to a number of (fixed sized) data blocks called segments. This facilitates effective memory management.

list of file attributes. The file name is the key field. Operating systems can use directories to “look-up” data corresponding to files.

Directories are used in hierarchical file systems. For example, in the Unix operating system the “root” directory—often designated by /—forms the base of the hierarchy. Files can have the same name if they are not in the same directory. Files can be unambiguously identified using their absolute *path*—the path from the root directory, through subdirectories, to the file—the symbol / separates directory names. For example the absolute path /usr/shared/dictionary.txt refers to the file dictionary.txt in the shared directory in the usr directory in the root directory.

Most operating systems also implement **link files** that record the absolute address of another file—the file they “link” to. Link files effectively allow hierarchical file system files to be located via a number of different paths.

[60, 88, 92]

The term “file system” (also written “filesystem”) has been defined as “a structure for keeping and locating data in files ...[66].” Of course the *purpose* of a filesystem is to make it easy to find and access these data files.

2.4.1 A critique of computer file systems

While hierarchical computer file systems have demonstrated the ubiquity of hierarchical classification, they have also demonstrated the shortcomings of this approach. Additional search tools are required as files can easily get lost in large—arbitrarily arranged—hierarchies. The restriction that files be indexed solely by name has also proven inconvenient. It would be useful to be able to group and retrieve files by type, size, owner, creation date, ... and other attributes as need dictates.

2.5 The Database

The earliest known use of the term “data base” was in November 1963, when the System Development Corporation sponsored the symposium “development and management of a computer-centred data base”[91]. In March 1976

the journal “ACM Transactions on Database Systems” was launched with papers selected from the international conference on very large data bases: September, 1975[45]. Prior to this, the term “databank” was also used with the same meaning.

Although it appears that “database” was introduced without, a variety of definitions have since been attempted. Although simplistic definitions abound, such as “a model of some aspect of the reality of an organisation[54, 12]”, the general consensus appears to be better reflected by definitions such as

A shared collection of logically related data, and a description of this data, designed to meet the information needs of an organisation[95].

Most authors agree that a database must either be shared, or have the capacity to be shared, between a number of users or programs. So, in general, databases must address security and data integrity issues common to multi user systems.

There is also general agreement that databases are *self describing*. Usually a database includes a **data dictionary**⁵—kept separate from the data itself—that describes the data in the database. This is intended to provide *program–data independence*, that is, to make it easy for a number of different programs to use the same database.

There is greater agreement on the meaning of other terms relating to databases. A **database management system**, abbreviated DBMS, is “a software system that enables users to define, create, maintain, and control access to the database[95].” A **database system** is “a collection of application programs that interact with the database along with the DBMS and database itself[95].” An **application program** is “a computer program that interacts with the database by issuing an appropriate request... to the DBMS[95].”

The relational model, introduced by Codd in [24], provides the theoretical basis for most databases. Indeed the term **relational database**—that

⁵Also known as a *data directory*, *system catalogue*, or *metadata*.

is, a database based on the relational model—is often taken to be synonymous with the term database. Section 4.2.6 discusses the relational model in greater detail.

Data can be reasonably defined as “raw facts without structure or context” and **information** as “structured data in context.” If these definitions are followed then Codd’s “atomic values” are data while his relations—that provide a structure and context for the data—are information. So although Codd called his model the “relational model of data[24]”, “relational model” would be a less controversial name as it is arguable whether it is a model of data, or a model of information.

Structured query language, abbreviated SQL, was introduced by Chamberlin and Boyce in [21]. SQL is a language for querying databases; it provides an interface between users or application programs and their database. SQL further promotes program–data independence as SQL statements are easily copied from one application program to another—even if the application programs are written in different languages.

2.5.1 Critique of the database

Although database systems are often contrasted with file-based systems[95] (called “file processing systems” in [55]) it may be more accurate to treat filesystems as a type of database.

One reason for the continued distinction between filesystem and database topics is that each topic has built-up its own distinct terminology and paradigms that make it difficult to appreciate the commonality of purpose that underlies these topics.

2.6 Information Retrieval

Calvin Moores coined the term “information retrieval” in his 1948 M.I.T. Master’s thesis[39]. According to [7], information retrieval, abbreviated IR, “deals with the representation, storage, organisation of, and access to information items.” More specifically, IR is about modelling user *information*

needs with *queries*, and then matching these queries with “documents” in a collection.

The *document* then is the fundamental “information unit” of IR—just as *atomic values* are the fundamental information unit in the relational model. Originally, information retrieval “documents” were actual paper and print documents that physically existed. Over time the term has become more general and now can refer to any information at all. Often the information is in the form of a computer file.

An **IR system** then is a mechanism—usually a computer program—that accepts queries and outputs sets of documents that it deems to be “relevant” to these queries. In early IR systems, a document was either relevant to a query or it was not. If a document was deemed relevant to a query by the IR system then the document was said to be “retrieved” by the query. More recent IR systems attempt to rank documents according to how relevant they are to the query, rather than divide the collection into retrieved and not retrieved documents.

IR has been contrasted with data retrieval systems (relational databases for example). According to some, IR queries are “inherently vague” and so sets of retrieved documents are “not exact” [90] while, in data retrieval systems “a single erroneous object among a thousand retrieved objects means total failure[7].” This is unfortunate as precise and accurate IR becomes a contradiction in terms!

The distinction between information and data retrieval is expressed more precisely by Rijsbergen who notes that **monothetic classification** involves “classes defined by objects possessing attributes both necessary and sufficient to belong to a class[79]” whereas, in **polythetic classification** “each individual in a class will possess only a portion of all the attributes possessed by all the members of that class. Hence no attribute is necessary nor sufficient for membership to a class[79].” Generally, data retrieval relies on monothetic classification whereas information retrieval relies on polythetic classification.

The vector model is perhaps the most widely used mathematical basis for information retrieval. Section 4.2.3 discusses the vector model for information retrieval. Probabilistic models such as the *binary independence retrieval*

model, introduced in [80], are also popular.

2.6.1 Measuring the effectiveness of IR

Unfortunately it appears that no techniques have yet been devised to measure the benefits of efficient and effective IR. These benefits, such as reduced research and development time and increased research quality, are quite significant.

Measures do exist to compare and contrast the effectiveness of IR between IR systems. *Precision* and *recall* are perhaps the most widely used of these measures. Given the set D of *retrieved* documents and the set T of documents that are *relevant* to a query (so D is the IR systems approximation of the set T)

$$\begin{aligned} \textit{precision} &= \frac{|D \cap T|}{|D|}, \text{ and} \\ \textit{recall} &= \frac{|D \cap T|}{|T|}. \end{aligned}$$

Another basic measure that is sometimes used is

$$\textit{fall out} = \frac{|\neg T \cap D|}{|\neg T|}.$$

Usually a set of queries is used, with the average precision and recall values over all queries being used to compare IR

A significant deficiency of these measures is that the set T of relevant documents must be subjectively determined by a “panel of experts” for each query. This is complicated by the fact that, because queries can often be interpreted differently, the set T may be different for different people, or even the same person at different times.

Modern IR systems rank documents by relevance, rather than simply determining if each document is relevant. These systems do not produce a set D of retrieved documents. In order to compute precision and recall values

for such systems we define

$$p_j = \frac{|D_j \cap T|}{|D_j|},$$

$$r_j = \frac{|D_j \cap T|}{|T|}$$

where D_j is the set of the first j ranked documents. So, if d_j is the j^{th} ranked document, $D_j = \{d_1, \dots, d_j\}$.

One method to compare modern IR systems involves plotting points (r_j, p_j) for each $j = 5, 10, 15, 20, 30, 50, 100$. The resulting curve, after connecting the points, is indicative of the effectiveness of the IR system and can be used to compare and contrast IR systems.

It is often desirable to compute a single value to use to compare IR systems. There are various different functions that have been used for this. The *average precision at seen relevant documents* value is the average p_j for each relevant document d_j . The *R-Precision* value is $p_{|T|}$.

Various methods exist to combine precision and recall measures. The *precision at recall level r* value is p_j for the j where $r_j = r$. The *harmonic mean* of precision and recall values for the set D_j is

$$F_j = \frac{2p_j r_j}{p_j + r_j}.$$

See [7, 79, 86].

2.6.2 A critique of IR

IR, as it currently exists, has a number of weaknesses. While not explicitly a requirement, IR systems rely on keywords—perhaps with supplementary information such as weighting and word position—to index documents. User information requests are also processed into queries that consist of keywords.

Much of the imprecision in IR is due to the fact that sets of keywords often fail to adequately model user information requests and documents. This is especially apparent if the documents are research papers. Researchers often

need to either *create new words* or *extend the meaning of existing words* to adequately describe aspects of their research.

When researchers create new words and these words are used to index their research papers, information “islands” are created. In order to discover a research island, one must know the special terminology that is used on that particular island, but if you know the terminology, then you must have already discovered the island! This isolating circularity creates research communities that are unaware of parallel research in related fields.

When researchers extend the meaning of existing words and these words are used to index their research papers, information “swamps” (also known as “infogluts[90]”) are created. Keyword searches that include a *polyseme* (that is, a word that has different, but related, meanings) or a *homonym* (that is, a word that has different unrelated meanings) tend to retrieve documents that do not use the word with the meaning that the user intended and hence are not *about* what the user requires.

The solution to these problems may involve providing some kind of dictionary of keywords, and differentiating between each of the specific meanings of polysemes and homonyms. The gene ontology[35] is one such research effort. In many IR systems, even without differentiating between specific meanings of polysemes and homonyms, thousands of keywords are used to index documents. Providing a readily browsable dictionary of keywords for these systems would be quite problematic.

IR systems rely on the (polythetic) classification of documents—all documents that are retrieved by a query belong to the same class. IR systems break a fundamental rule of classification, see section 2.3.3, in that these classes do not reflect “real” divisions of knowledge. Indeed by using homonyms, it is possible to create separate documents that, while being *about* completely different things, are retrieved by the same queries. Similarly, using synonyms, it is possible to create separate documents that, while being *about* the same thing, may never all be retrieved by the same query.

By inverting the similarity function used to rank documents, it is often possible to create a function that gives a “difference” between documents. However—because IR classes are not based on sensible divisions of

knowledge—these differences are simply not useful as anything but plagiarism detectors.

IR systems sometimes automatically choose the keywords used to index documents from documents in the collection. Generally keywords that occur in too many or too few documents are not used. Because adding and removing documents to and from the collection changes the relative frequency of keyword occurrence in the collection, adding documents can have the effect of adding or removing keywords from the index—as can removing documents.

In some systems weights are also assigned to keywords in the index. If weights are associated with index keywords, these can also change. As a result of this, a query that retrieved a particular document may not continue to retrieve the document after documents are added or removed from the collection. As discussed in section 2.3.3, *indexing consistency* over time is desirable and *access path inconsistency* such as this should be avoided.

Library users enjoy the benefits of a significant amount of information *not* contained in the classification scheme. The size and style of the library building; the layout of the shelves; the weight and smell of a book; the colour and feel of the pages; the number, age and browsing habits of other library users. This is all helpful information about the use and usefulness of the library collection. Library users cannot help but add to their knowledge of the extent and organisation of the library collection simply by walking through the library building. Cues such as these are not well emulated by IR systems.

As has been discussed, computer filesystems are “a method for storing and organising computer files and the data they contain to make it easy to find and access them.” While information retrieval “deals with the representation, storage, organisation of, and access to information items.” It is implicit that these “information items” are computer files of some description. It would appear that definitions of computer filesystems and information retrieval, such as these, could easily be interchanged. It is possible to conclude from this that computer filesystems can be information retrieval systems and that information retrieval systems can implement computer filesystems.

2.7 Data Warehousing

There are two central personalities in the data warehousing community: Inmon, who advocates entity relationship (third normal form) data warehouses[50] and Kimball, who advocates dimensional models[75]. Although there is substantial overlap in the points of view of these two authors, they disagree on many important aspects of data warehousing.

When there is disagreement, this review adopts Kimball's position. There are a number of reasons for this:

1. dimensional models have become the dominant paradigm for data warehousing in recent years;
2. dimensional models are quite relevant to this thesis;
3. Inmon's criticism of dimensional models in [49] appears to be based on a number of misunderstandings.

Generally **data warehousing** is about building databases that have been optimised for analysis. Data warehouses are contrasted with **operational databases** that record and retrieve the everyday transactions of an organisation. While a data warehouse is a generic solution to a business's data analysis needs, **data marts** provide specialised analysis. According to Kimball's vision, a data warehouse consists of number of data marts.

Dimensional data marts are commonly implemented either by using a star-join schema and a relational database⁶, or a multi-dimensional data cube.

A **star-join schema** consists of a single *fact table* and a number of *dimension tables*. The primary key in the fact table consists of a number of domains that are foreign keys⁷. Each of these foreign keys is the primary key in one of the dimension tables. Domains in the fact table that are not foreign keys are generally numeric and their values can be meaningfully summed[75].

A number of terms have been used to informally describe operators on dimensional data marts. These operators are used to produce *reports* that

⁶See section 4.2.6 for a discussion of the relational model.

⁷See section 4.2.6 for definitions of "the primary key" and "foreign key."

summarise the data. These reports are themselves tables with domains chosen from dimension and fact tables. These operators **slice and dice** the data mart: they use arbitrary dimensional constraints to aggregate fact table data. **Drill down** operators expand reports to show more detail by adding further domains (table columns). **Drill up** operators summarise reports by removing domains (table columns)[75].

Example 2.2.

Time Dimension Table		Fact Table
Time_id		Time_id
week		Location_id
month		Product_id
year		Employee_id
		Sales

Location Dimension Table		Product Dimension Table
Location_id		Product_id
State		Product_Type
City		Description
Postcode		Supplier
Store		Wholesale price
		Margin

Employee Dimension Table	
Employee_id	
Name	
Contact	
Responsibilities	
Salary	

Table 2.1: Simplified Star Schema for Nationwide Retail Chain

In the star schema data mart illustrated, the data is *sliced and diced* to answer question “what was the total profit—week 5, all products, all employees?” In response to which, the data mart might generate the report:

Week 5 Total Profit

\$517 567

Where “profit” is determined by multiplying the sales volume of a product with its margin. Following on from this, users might *drill down* by *state*, perhaps generating the report:

Week 5	NSW	QLD	VIC	WA	NT	TAS	SA
Total Profit	\$120 583	\$97 187	\$85 249	\$69 357	\$65 184	\$48 797	\$31 210

Users can continue to *drill down* by *Product_Type*, generating the report:

Product_Type	NSW	QLD	VIC	WA	NT	TAS	SA
consumable	38 452	27 561	24 386	21 175	20 576	16 037	11 056
power tool	31 014	24 398	22 009	17 485	17 625	12 867	10 465
hardware	29 764	25 875	23 187	19 321	17 662	12 987	7 105
gardening	10 873	10 267	8 393	7 012	6 403	4 974	2 035
sale item	10 480	9 086	7 274	4 364	2 918	1 932	549
Week 5							
Total Profit	\$120 583	\$97 187	\$85 249	\$69 357	\$65 184	\$48 797	\$31 210

If users wish to focus on South Australia, they can *drill up* on *SA*, generating the report:

Product_Type	SA
consumable	11 056
power tool	10 465
hardware	7 105
gardening	2 035
sale item	549
Week 5	
Total Profit	\$31 210

Of course the actual values in these reports would depend on the data stored in the data mart.

Data warehousing is closely related to a number of other topics including on-line analytical processing, abbreviated OLAP⁸, data mining, web farming, executive information systems (EIS), decision support systems (DSS), and business intelligence (BI).

⁸Section 4.2.7 discusses OLAP briefly.

2.7.1 Critique

The huge amount of data that is being collected and stored can be a valuable resource—if managed and analysed correctly. By making this resource available, data management and analysis tools—like data warehouses—can be quite valuable. The commercial world is beginning to realise this. This has resulted in an explosion of interest in this area. However, science has yet to catch up with this rapidly developing area and many of the publications on these topics have little—even inadequate—scientific basis.

Formal definitions and mathematical models are scarce and, as a result, there are a number of different opinions on precisely what data warehousing is and how it should be done. This inconsistency and imprecision is undesirable. As data warehousing itself acknowledges.

Information from one business process should match with information from another. If two performance measures have the same name, then they must mean the same thing. Conversely, if two measures don't mean the same thing, then they should be labelled differently... Consistency also implies that common definitions for the contents of the data warehouse are available for users[75].

If the metaphor of an n -dimensional space is taken seriously, points in the space are n -tuples of dimension table primary keys (foreign keys in the fact table). Row values in the fact table that are not part of the primary key are information associated with points in the space. Row values in dimension tables that are not primary keys are used to select sets of dimension table primary keys, and so specify regions in the space.

2.8 The Internet, World Wide Web and Semantic Web

2.8.1 The internet

The Internet is a worldwide computer network. Computers attached to the internet, called “hosts”, communicate with one another by sending “packets”

of information. Smaller computer networks are connected together using dedicated packet-switching computers called “gateways” or “IP routers” or “Intermediate Systems” [94].

The internet protocol suite specifies how information is transmitted between computers in the network. The internet protocol suite is organised in “layers”, with each layer providing the services required by the layer above. Although the open systems interconnection basic reference model, abbreviated OSI reference model, describes seven layers [105], the internet protocol suite is not consistent with this model and formally has four layers: application, transport, network and datalink [94]. In practice, a fifth “physical” layer—supporting the datalink layer—is sometimes added.

To communicate using the Internet system, a host must implement the layered set of protocols comprising the Internet protocol suite. A host typically must implement at least one protocol from each layer [94].

2.8.2 The world wide web

The world wide web, abbreviated www or web, is a marriage of hypertext to the internet. There are five key technologies, developed for the www, that make the www possible: hypertext markup language, uniform resource identifiers, hypertext transfer protocol, web browsers, and web servers.

Hypertext markup language (HTML) is used to structure web pages. HTML can also include embedded scripting language code which can affect the behaviour of web browsers and other HTML processors [27]. In 2000, HTML became an international standard (ISO/IEC 15445:2000) [37].

Uniform resource identifier (URI), syntax is used to identify and locate web pages. If a page is available on the internet, its URI includes a domain name, which is associated with an internet protocol (IP) address (IP is a network layer communication protocol [94]) by the domain name system (DNS) [9, 65]. All URIs are formed with a scheme name, followed by a colon character, and the remainder of the URI.

Hypertext Transfer Protocol (HTTP) is an application layer communications protocol used to transfer information “on the www”. Its main use

has been to help publish and retrieve HTML hypertext pages. Resources to be accessed by HTTP are identified using URIs using the http or https URI schemes. HTTP communication usually takes place over TCP/IP internet connections[36]. TCP is a transport layer communication protocol.

Web browsers are software applications that display web pages and allow users to follow hyperlinks to other web pages.

Web servers are computers (or computer programs) that accept HTTP requests from web browsers and “serve” HTTP responses along with web pages—such as HTML documents and linked objects (images, etc.).

After a web browser user types the URI of a web page into a web browser, or follows a hypertext link to that page, the web browser resolves the URI into an IP address using the DNS. The browser then requests the resource by sending an HTTP request to the web server at that address. Having received the required files from the web server, the browser then renders the page onto the screen as specified by its HTML (and other web languages). Any images and other resources are incorporated to produce the on-screen web page that the user sees.

2.8.3 The Semantic Web

Although there were earlier rumblings, such as [11], [10] is generally accepted to be the initial position paper for the “Semantic Web”. In essence, the semantic web is an attempt to establish a framework to support a new class of “software agent” that “understands the web”. The plan to achieve this is centred around creating “ontologies” that allow web content to be “marked-up” with “meaning” tags—just as HTML tags currently mark-up web pages with “structure” tags.

At the core of the semantic web is this attempt to define formal “knowledge ontologies” that describe and relate knowledge. These ontologies are precisely what is required to improve IR (see the discussion in section 2.6.2). However the semantic web is far more ambitious than this!

A motivational idea behind database systems is to allow a number of independent application programs to work with the same collection of data.

The database software design philosophy of separating data structure, storage and retrieval from “programming logic” is time tested and has proven highly effective. The semantic web represents an attempt to extend this idea—turning the entire web into a self describing database.

Perhaps the main weakness of the semantic web project is the scope of its ambition. Despite many decades of trying, artificial intelligence, abbreviated AI, researchers have not been able to accomplish the wholesale translation of human knowledge into “machine comprehensible” format. It is also not clear that the semantic web project has developed any substantial new theory to tackle this problem.

It may have been better to present the semantic web as an effort to:

1. support and improve IR—by identifying keyword polysemes, homonyms and synonyms—*and*
2. develop a class of software that uses IR—similarly to how existing database applications use databases.

It seems likely that a modest expression of the goals of the semantic web such as this—clearly *not* requiring the development of new AI theory—would have, more rapidly, resulted in practical achievement.

It may also be true that reputable research journals would have less hesitation publishing “semantic web” articles if the initial article [10] was published in a research journal, rather than a popular, albeit scientifically orientated, magazine.

However the semantic web is interpreted, activity aimed at creating knowledge ontologies should not be lightly disregarded. This is vital work. Agreed “meaning ontologies” need to be developed to assist in navigating the developing “terminology jungle”⁹. Furthermore, the idea of improving IR to the point that applications can rely on IR, rather than data retrieval, is intriguing.

⁹Even the terminology and acronym inventions of the World Wide Web consortium, abbreviated W3C, alone could benefit from such ontologies!

2.9 Discussion and Summary

Traditional methods of organising information are becoming inadequate as the amount information available increases (see section 2.3.3.) The research response to the problem has been quite fragmented. In general, researchers in the topics reviewed in this chapter appear to not be well informed about relevant parallel research in other, related, topic areas. The similarities and differences between these topics are also not well understood.

One could readily conclude that it is long past time for the research community to get its house in order. One way to address the problem would be to introduce a new “organisation of information” discipline. Specialised topics such as classification, databases, information retrieval, data warehouses, etc. would become sub disciplines of this new discipline. As a discipline, the organisation of information would map out the relationships between its sub disciplines and provide a dictionary of common terminology. Researchers in general could look to this discipline to assist in describing relationships between, and providing common terminology dictionaries for, their own research areas.

As yet there are no proven techniques that “solve” the problem of the organisation of information. Faceted classification, database systems, information retrieval, data warehousing, the internet, the web and the semantic web provide partial and incomplete solutions.

2.10 What this chapter achieved

This chapter surveyed, reviewed and concisely summarised the literature relating to the topic of the organisation of information. This summary gives an indication of the state of the topic as a whole.

There is little evidence of any attempt to establish an overarching theoretical basis for this topic. Nevertheless there appears to be significant common ground between the numerous specialised, independently developed, information organisation systems. This suggests that these specialised systems may be instances of a more general, and theoretically significant, system.

Chapter 3

Knowledge Libraries

3.1 Overview

This chapter presents a number of scenarios that illustrate the use of “knowledge libraries”. The key terms and concepts illustrated are listed at the beginning of each scenario. A commentary is provided to help interpret each scenario. The purpose of this section is to begin forming an understanding of what knowledge libraries are, and the uses to which knowledge libraries may be put.

The chapter goes on to discuss three of the main application areas of knowledge libraries: research, education, and business. The potential impact of knowledge libraries in each of these areas is assessed. This further clarifies the nature and use of knowledge libraries and helps establish their value.

Section 3.5, identifies “core” knowledge library functionality. This is the functionality that provides essential support for fundamental knowledge library features. Two classes of core functionality: administration and end use functionality are identified.

Although security is not classed as core functionality, it is essential for many applications. Section 6 discusses how the security requirements of these applications can be met by specifying access restrictions on arbitrary user classes.

Section 3.7 identifies “extended” knowledge library functionality and dis-

cuss how this functionality can be achieved. The core and extended functionality together provide all the knowledge library functionality that section 3.3 (knowledge library user scenarios) and section 3.4 (the uses of knowledge libraries) indicate knowledge libraries should possess.

Section 3.8 briefly summarises what has been achieved in this chapter.

3.2 Introduction

Advances in the organisation and management of information like information retrieval, relational databases and data warehousing use computer technology to increase the speed and reliability of existing techniques.

Information retrieval automatically constructs keyword indexes of entire collections—rather than just single documents—and automatically retrieves digitised documents that are indexed by keywords in user queries.

Relational databases enable users to record information in data tables, perform automated searches, and generate new tables from old.

Data mining techniques enable users to automatically search for patterns in unindexed data and analyse the results.

In summary, computer technology has been used to:

1. construct elaborate indexes for retrieving documents;
2. organise data in tables;
3. search for patterns in data.

In all three cases computer technology has been used to automate what were time consuming, error prone, tedious manual processes. The resulting tools certainly open-up new possibilities and, to users more accustomed to traditional manual techniques, may seem to offer the ultimate in convenience.

Another approach is to attempt to imagine entirely new paradigms for information interaction—idealised tools that completely satisfy *user needs* with respect to information organisation and interaction. This approach—if successful—can lead to fundamental, rather than incremental improvements.

Notable early examples of this approach include H.G.Wells’ book “World Brain” [98] and Vannevar Bush’s essay “As We May Think”[18]. Some of the ideas expressed in these works were later realised by innovations such as the computer mouse, word processing, email, hypertext, icon based interfaces, the internet and the world wide web.

Codd, Codd and Salley’s description of “on-line analytical processing” (OLAP) in [25, 26] is another significant visionary effort, as is Berners-Lee, Hendler and Lassila’s article “The Semantic Web”[10]. In these cases however, the omission of mathematical definitions has lead to some ambiguity and confusion.

This thesis presents an attempt to imagine, define, mathematically model, design, and implement and test core functions of *knowledge libraries* for the organisation and presentation of information.

3.3 Knowledge Library User Scenarios

This investigation of knowledge libraries begins by imagining—by way of a number of user scenarios—the way knowledge libraries could be used. These scenarios will help to clarify what knowledge libraries are, what activities they support, and what functionalities they provide. The scenarios also introduce some informal terminology, some of which will be given more formal definitions.

Scenario 1: Joe User — I.T. company employee

Key terms introduced. *Knowledge library, map, dimensions (manual, automatic, dynamic), coordinate, region, information units, information elements, attach, attach point, information space, granularity (of information space).*

Concepts illustrated. *Knowledge library, dimensions, selection, ordering, automatic report generation.*

Joe User sits down at his computer and opens a *knowledge library*. Using graphical displays, summary statistics and text he very rapidly sizes up the contents of the library: its size, areas of relevance, coverage, interconnectedness, level of sophistication etc.

Let us call this view a *map*. In many ways this metaphor is quite apt. Yet care must be exercised not to over extend the metaphor—most information spaces are not Euclidian and cannot be readily represented by points in Euclidian spaces and so cannot be mapped in any conventional sense.

Joe can see that the library contains a lot of information, very little of which is relevant to the task at hand. He browses through the available *dimensions*...

Each dimension orders and filters the information in the library according to its own rules. Some of these dimensions are *manual*: they impose their own fixed ordering on the information in the library that does not change. As the library grows, new information has to be added manually to these dimensions. Other dimensions are *automatic*: as new *information units* are added to the library, the dimension's ordering algorithm automatically attaches the information to the correct coordinates in the dimension. Some dimensions are *static*—the coordinates and distances *do not change* automatically and must be added or removed manually. Still other dimensions are *dynamic*—their coordinates and distances are generated automatically and may constantly change as information units are added to and removed from the library¹.

manual, static	automatic, static
manual, dynamic	automatic, dynamic

Table 3.1: Dimension types

Joe selects a number of compatible dimensions that look suitable to begin his search. By selecting sets of *coordinates* in each dimension, Joe rapidly

¹See the example of a manual, static dimension in section 3.5 and examples of automatic and static dimensions in sections 3.7.1 and 3.7.2 respectively

zeros in on the information he is looking for. Whenever he makes a selection, the map changes to depict the *region* he has selected.

When Joe is satisfied that the region he is currently viewing precisely describes the information he is after, he begins browsing through some of the *information elements* that are *attached* to a point in the space within the region. Before he views an information element he briefly examines the map of the point to which it is attached—getting a quick summary of the characteristics of the information including its coordinates in the *information space*.

After examining a number of information elements Joe realises he has found what he was after. The *granularity* of the information space is fairly fine—with one information element containing about a paragraph of text, or a single diagram or image, or a single mathematical equation, theorem or definition. The information Joe was looking for is contained in a few closely related information elements. Joe selects a couple of further information elements that provide useful background information. One of the dimensions Joe has open provides a sensible ordering and with a click of a button he has generated a report—an electronic document that includes all the information Joe has selected. Joe adds a few notes and with the click of another button he adds the report to another knowledge library that he and his co-workers use to organise and share their work. ...

Commentary. *Knowledge libraries rely on a spatial metaphor to organise and present information. Let us call the “space” underpinning a knowledge library its “information space”. In this thesis a careful definition of information space is developed. This definition is the mathematical basis for knowledge libraries. This scenario clarifies that information is, almost always, an organised collection of other information—so a region of one information space could easily be an information element of another information space.*

Scenario 1.1: I.T. company employee manager

Concepts illustrated. *Using information space as a management tool, automatic notification; “employee”, “time”, “job type” dimensions.*

[continued from scenario 1] ...In another office, Joe’s team manager observes the rapidly growing project information space with satisfaction. Her information space has an “employee” dimension which has a coordinate for each employee in her team, a time dimension and a number of “job type” dimensions. She can easily observe the progress of each employee as well as collaborations between employees. She has set her knowledge library to notify her whenever certain performance targets are achieved or whenever progress lags in certain critical areas. ...

Commentary. *“Watching” information space regions of interest for changes and comparing the same region over time are features with clear utility that reinforce the spatial metaphor and underscore the desirability of having a flexible and accurate spatial basis for information organisation. Given an n -dimensional information space, users may well wish to add a further “time” dimension. In this dimension:*

1. *“past” coordinates are attached to records of the n -dimensional information space from the past;*
2. *the “current” coordinate is attached to the current n -dimensional information space—the space without the time dimension;*
3. *“future” coordinates can be attached to “performance targets”, “project milestones” etc.*

Scenario 1.2: I.T. company project management—time dimensions

Concepts illustrated. *Optimistic, realistic and cut-off time dimensions. “precedence” dimensions. “milestone” information elements.*

[continued from scenario 1.1] ...These key milestones had been worked out in consultation with the client during the initial design and planning phase of the project. The client was presented with simplified dimensions that map out how the project would be completed in fairly broad terms—essentially time and precedence dimensions to which “milestone” information elements are attached. Three time dimensions were used: “optimistic”, “realistic”, and “cut-off”. The optimistic dimension classifies milestones by their earliest conceivable completion date. The realistic dimension classifies milestones according to the company’s most accurate projection of when the milestone will be reached. The cut-off dimension classifies milestones by the date, after which, the project may be considered a failure. The precedence dimension represents the precedence relation between milestones. ...

Commentary. *The “milestone” information elements indicate how the information space should progress over time. Milestones can be compared with information space summaries to get an indication of how “close” an information space is to a preset target. Dependency dimensions give a partial order of project milestones—indicating which milestones must be completed before other milestones.*

Scenario 1.3: I.T. company project management

Concepts illustrated. *Access controls. Different types of information elements. Automatic report generation.*

[continued from scenario 1.2] ... The client maintains the ability to monitor the overall progress of the project through these dimensions—without being able to access details that the company considers commercially sensitive. The company also prepared more complex and detailed dimensions for its own use. Managers prepare work schedules by progressively breaking up project milestones into far more detailed and precise requirement specifications.

The information elements in the client’s information space are project milestone specifications and progress reports. The information elements in the company’s information space include research and progress reports, detailed designs, source code, detailed project milestone specifications, individual employee information spaces...

The client can, with the click of a button, generate a progress report. This report is automatically generated from information in the company’s information space. In a similar process, information elements in individual employee information spaces are automatically generated by employee activity in other information spaces. These employee information spaces are like employee résumés. They give an indication of the employee’s value to the company and their areas of expertise.

Commentary. *This scenario outlines the use of a large and complex knowledge library. Importantly:*

1. *Access controls can limit the availability of information in an information space, allowing an “information subspace” with limited information to be made more widely available and perhaps used for a different purpose.*
2. *“Progress reports” can be automatically generated by comparing past and present regions in information space.*
3. *User interactions with an information space can be recorded as information elements in a more expansive information space (add “user”, “activity” and possibly “time” dimensions).*

Scenario 2: Robert — Researcher

Key terms introduced. *Knowledge precedence. Knowledge space.*

Concepts Illustrated. *Automatic notification (new information element). Automatic notification (unsatisfied knowledge dependencies). Knowledge dependencies. Knowledge space. Distance between sets (Robert's knowledge space and required topics).*

When Robert logs onto his computer in the morning he is greeted with a message: it seems there has been a new publication in his field of expertise. Robert opens the paper, but before he begins to read a warning appears on the screen: he is informed that he hasn't yet mastered all the topics necessary to comprehend this paper. Robert ignores the message and begins to read the paper anyway, but he quickly discovers that he is unable to fully understand it.

Robert reviews the *knowledge dependencies* of the paper and observes that there are only a few topics that he needs to master, and they are not far from his personal knowledge space. It doesn't take Robert long to extend his *knowledge space* to satisfy all the dependencies. Because the paper looks fairly interesting and complex, Robert decides to review a few dependent topics that he hasn't used in a while as well.

Robert reads the paper and this time it makes perfect sense—in fact Robert can see that the paper has implications that the author did not think about. Because, if accepted, it is a publishable result (that adds a few points to his publication record), Robert writes a short extension of the paper, including the new implications. Robert notes with satisfaction that after a short on-line debate, his extension is accepted, the paper is updated, and he is credited with a number of publication points. ...

Commentary. *Robert’s area of expertise is modelled by a region in the research information space. He has set his knowledge library to notify him whenever information elements are attached to this region. Similarly, Robert’s knowledge is modelled by a region in information space—this is Robert’s “knowledge space”. The background knowledge required to understand papers uses precedence links—similar to citation links. Contributions, including extensions, to this information space are worth “publication points” and must be assessed and accepted by referees (with appropriate background knowledge).*

Scenario 2.1: Robert—skill assignment and development

Key term introduced. *Apex topic.*

Concepts Illustrated. *Using information space to model skills and expertise. Using distance to determine skill match. Automatic generation of study sequence. Employee value as a function of their knowledge space.*

[continued from scenario 2] ...Robert proceeds to look over the day’s work roster. The company that he works for has been successful in winning a number of new contracts—in part because they were able to demonstrate that they had all the skills and expertise that the projects required. Unfortunately the company’s unprecedented success has left it over committed—employees with skills in key areas are already fully committed to other projects.

The company has done a search and assigned available employees with the closest skill match to the unfilled positions. Robert has been assigned one of these positions! He reviews his new responsibilities and notes the *apex topics* he must master. A study sequence has been automatically prepared for him. Robert notes with satisfaction that he will need to do a significant amount of

study—extending his knowledge space and making him a considerably more valuable employee—resulting in an increased pay rate! ...

Commentary. *Company employees record their knowledge in the company’s knowledge library—effectively combining their own “knowledge space” with the company’s. The information space has a precedence dimension—making it possible to automatically identify viable study sequences. Employee pay rates can even be automatically determined from their work output and knowledge space.*

Scenario 2.2: “The company”—information valuation

Concepts Illustrated. *Company valuation as a function of the company’s knowledge space.*

[continued from scenario 2.1] ...The company has been very successful with the types of projects it has been working on lately and is keen to expand its operations to include larger, more complex, and more lucrative projects. The company executives have considered expanding by recruiting more staff, but they have decided it would be more effective to acquire a range of expert employees all at once by buying another company. They have been negotiating with a number of partner companies and have found a good match. An executive notes: “if company x ’s knowledge space is added to our own, then we will be in an excellent position to compete for the projects we’ve been missing out on.”

Commentary. *The systematic organisation of information in knowledge libraries paves the way to effective information valuation. In this scenario “the company” wants to improve its knowledge space. A monetary value to the company of such improvements can be calculated. This enables the company to assign a value to the knowledge space of other companies.*

Scenario 3: Ruth—Research proposal development

Concepts Illustrated. *Knowledge library as a file system/web browser/database server. Security. IRIS. Using information space to demonstrate the feasibility of research and to value information. Automatic generation of background knowledge requirements.*

Ruth is beginning a new research project. Her computer desktop/file system/web browser/database server is a fully integrated knowledge library/information space. This single program provides an interface to all information stored in Ruth's computer or on the internet. Ruth has a number of security settings. The majority of Ruth's work is stored on her computer at the University and (automatically) backed-up on her computer at home. Only she can view this material. Other material is can be viewed (and perhaps modified) by her supervisor. Yet other material can be accessed by members of her research group. Further material (not stored on her computers) is accessible through the international research information space (IRIS). IRIS is of particular interest to Ruth.

Ruth has just completed the preliminary phase of her research. During this phase Ruth completed a research proposal with Supervisor. The proposal document was attached to IRIS. Ruth and Supervisor were able to demonstrate:

1. the research was feasible—given certain specified resources and within the specified time period;
2. if successful, the research would result in valuable new knowledge;
3. Ruth and Supervisor had the necessary background knowledge/expertise to conduct this research.

Ruth used her knowledge library to good effect—she was able to show that the new information units her (successful) research would create would satisfy unfulfilled requirements in a number of strategic research initiatives and large cooperative research projects.

Thanks to the classification of her research proposal on IRIS, Ruth was able to use her knowledge library to generate a list of all background knowledge required for this research. She was also able to show, thanks to information space representations of her and Supervisor's knowledge that she and Supervisor, between them, satisfied much of the background knowledge requirements. Furthermore, Ruth showed that the distances between her and Supervisor's current knowledge (grouped together) and the unsatisfied knowledge requirements was not great. ...

Commentary. *Computer desktops, computer file systems, web browsers and database servers all preform similar functions related to the organisation and presentation of information. It may be more coherent for a single program—a knowledge library—to provide this functionality. This scenario further illustrates the utility of being able to accurately determine distances between information elements.*

Scenario 3.1: Ruth—the literature review

Concepts Illustrated. *IRIS—The International Research Information Space.*

[continued from scenario 3] ...Ruth begins the initial phase of her research—reviewing background information/research. As she reviews information on IRIS, she notes its usefulness and implications for her own research. She notes when:

1. she confirms that an area of research is useful in the way she thought it would be;
2. research areas she had thought would be useful, but that on closer inspection turn out not to be;
3. research areas that have unanticipated (positive) implications/usefulness.

As an active researcher, this information is of interest to other researchers in Ruth's area. Ruth publishes many of her research notes on IRIS where they can be viewed and commented on by others. These observations:

1. help with the quality control of articles;
2. help establish the correct classification of articles;
3. help Ruth establish links with other researchers in her area;
4. establish the correct classification of Ruth's research (and so the relationship between Ruth's research and existing literature).

By the time Ruth finishes her research project she will have already have published many observations on IRIS that firmly connect her research to IRIS. Ruth will not need to write a separate literature review as her supervisors/examiners will be able to readily access this material on IRIS. ...

Commentary. *Advances in the organisation and presentation of information may change the way science is communicated. A single coherent international system for managing scientific research such as IRIS would undoubtedly have many benefits for the international research community. The introduction of such a system could even herald a new age of scientific cooperation resulting in a dramatically increased pace of scientific development.*

Scenario 3.2: The International Research Information Space (IRIS)

Concepts Illustrated. *IRIS*

[continued from scenario 3.1] ...IRIS is a free (grant funded), distributed research information space server. IRIS publishes research papers (and information spaces), reviews, critiques, paper modifications (including classifications) and (IRIS) dimensions. While many IRIS dimensions—maintained by member institutions—are freely available (grant funded), IRIS does allow certain institutions (e.g. academic publishers) to charge for access to their dimensions in exchange for making their collections freely available on IRIS. Individual dimension maintainers also control what is attached (e.g. research papers etc.) to their dimensions and, of course, the creation, destruction, span and distance between coordinates.

Commentary. *IRIS appears to offer such significant advantages it is worth discussing further how it could work. It may be possible, for example, for existing academic publishers to contribute to IRIS while continuing to maintain income derived from copywrited material.*

Scenario 4: A census data information space

Concepts Illustrated. *Using information space as a database, using information space to store and display questionnaire responses.*

Census data are collected and entered into a government, on-line, information space. Users can access information from this site according to their security profile. Individual census forms (after processing and supplementa-

tion with information from other sources) are the information elements in this space. There is a dimension for each question and a coordinate for each answer.

Users are readily able to form queries that answer questions like:

1. “What is the average income of people in suburb x ?”
2. “What is the total (weekly) private fuel consumption of electorate y ?”
3. “How does the educational profile of the population of state a differ from the population of state b ?”

Commentary. *Although the main use envisaged for knowledge libraries is to organise and present fairly complex information elements such as reports, research papers, web sites etc. it is desirable for information space to be flexible enough so that sensible results are obtained when comparing sets of far simpler information elements—such as numbers. In general, given a questionnaire (such as a census form or exam) a knowledge library should have the capacity to store and present the results. A questionnaire consists of n questions. Questions can have a number of different standard answers (e.g.. tick one of five boxes). Questions may also have short written responses (e.g.. age, name, address, short comment).*

Scenario 5: Edna—Knowledge Libraries in the Classroom

Key term introduced. *Knowledge requirement*

Concepts Illustrated. *Knowledge space, using knowledge libraries as a motivational tool, using knowledge libraries and knowledge spaces to identify effective teams.*

Edna the educator looks over the days lesson plan. Although the class has a diverse background, homework assignments have been set to ensure that all the class has the required background knowledge for today's lesson. When viewed as a whole, the class' knowledge space is quite impressive—some of the students have been doing some extra reading in their areas of interest. Indeed one of her students has been systematically acquiring the foundation *knowledge requirements* for a Veterinary Science degree!

Edna finds that knowing the ambitions of her students is a great help in motivating students. She is always sure to mention when a learning module satisfies a knowledge requirement in an area of particular interest to one of her students. Edna also uses this information to divide the students into suitable groups for class projects. She makes sure that—between them—each group has the basic competencies required to tackle each project.

Commentary. *This scenario illustrates two important uses of knowledge libraries. First, knowledge libraries can be used to systematically record the knowledge requirements and dependencies of vocations in such a way as to make available to students and educators an overview of the learning requirements of vocations of interest. Second, knowledge libraries can be used to model the knowledge and understanding of students. This information can be used to tailor lessons to the needs of students and identify suitable teams for larger projects.*

3.4 The Uses of Knowledge Libraries

This section discusses three of the main application areas of knowledge libraries: research, education, and business.

3.4.1 Knowledge Libraries for Research

Because information space models the real relationships between information units, knowledge libraries, based on accurate information spaces, will provide accurate distances between information units. This will make it easier to

discover synergies between different research areas and to identify gaps in the literature.

Knowledge libraries containing documents relating to both research and the application of research will clarify the relationships between research and research applications. This, along with other data such as the number of citations, will make the valuation of research easier and more accurate. Indeed, standard research publication could be merged with patenting, so that research papers double as patents and researchers automatically receive remuneration from the commercial development of their research.

In an ideal system, researchers may choose to get their research *certified*. Users of knowledge libraries could then choose to retrieve only those research papers that have been approved by a *certification authority* that they trust. Such certification authorities could replace some of the functions of today's research publishers.

Because the position of a research paper in information space relates the paper quite precisely to other research, this information need not be duplicated in the paper itself. So “introduction”, “background”, “literature review” sections of knowledge library papers can be substantially reduced. Knowledge libraries can also make clear precisely what background knowledge is required to understand research papers.

Research knowledge libraries can be more finely grained than existing research databases. Research papers must tell a complete story—perhaps introducing a new idea, relating it to the literature, motivating the idea, and proving its viability. However the information elements of a research knowledge library can be much simpler—a theorem, a proof, a counter example, a problem description, a critique... The information space itself can provide the thread that weaves these elements into a larger fabric. Also, a single information element can embody a larger idea by referring—by way of direct links—to a number of other information elements.

Impact Statement. *Knowledge libraries could simplify, clarify and streamline the process of research—helping researchers identify and take advantage of research synergies. Knowledge libraries should reduce the costs and time required to do research, while increasing and clarifying our appreciation of the value of research.*

3.4.2 Knowledge Libraries for Education

Using precedence dimensions, educational knowledge libraries can be constructed to give users a precise understanding of knowledge dependencies that exist between the learning modules they contain. Information spaces that contain learning modules, vocational requirements, job descriptions and practical applications can be used to give students a good understanding of the real world value of knowledge and an ability to link education and career goals.

Because knowledge libraries provide “maps” of their contents, they can be used to give students overviews of knowledge and an understanding of how learning modules and learning objectives fit together.

Because student use can be monitored, and knowledge spaces can model student knowledge, educational knowledge libraries have great potential for flexible delivery and self study based learning. Study programs can be effortlessly—and automatically—developed that target the learning requirements of individual students!

Impact Statement. *Knowledge libraries can help us develop a flexible education system that can rapidly adapt to meet the demands of a diverse and changing society. Increased flexibility and better modelling of student knowledge will result in greater efficiency—enabling students to learn more in less time. Knowledge libraries will also clarify the value of knowledge, resulting in increased student motivation.*

3.4.3 Knowledge Libraries for Business

Business knowledge libraries can be used as an organisational tool for organising large projects. Time dimensions can be used to provide a timeline and information elements can outline the jobs that must be completed at each point in time. Other dimensions can be used to assign staff and resources to jobs. Details can be added as the project is developed.

Portions of these organisational knowledge libraries can be made accessible to customers—allowing customers to track project progress and access information—such as the contact details of the employees responsible for each aspect of the project. In this way, knowledge libraries can be used as presentational tools to liaise with customers.

Knowledge libraries can be used to identify and store the knowledge spaces of individual employees. The sum of these knowledge spaces is a knowledge space that represents a company’s know-how. The capabilities of other company assets—such as machinery—can be similarly represented. So knowledge libraries can be used as a representational tool to identify a company’s capabilities for the benefit of potential customers.

Because knowledge libraries can be used to store and represent employee knowledge, and they can also be used to store and represent jobs and the contribution of each employee to each job, knowledge libraries can be used as a tool to identify and value staff knowledge and productivity.

Impact Statement. *Knowledge libraries can be used to help organise large projects, value the contribution of employees and present the capabilities of companies to potential customers. Knowledge libraries will simplify, clarify and streamline the way businesses are managed by integrating organisational, presentational and accounting business aspects.*

3.5 Core Knowledge Library Functionality

This section identifies “core” knowledge library functionality based on the discussion about knowledge libraries to this point. Core functionality pro-

vides essential support for fundamental knowledge library features. Functionality that is not core can either be provided as a by-product or combination of core functionality, or is not an essential aspect of a fundamental feature of knowledge libraries.

Manual, static dimensions, as discussed in section 3.3, scenario 1, are provided for by the core functionality.

Example 3.1. The American Mathematics Society maintains a hierarchical Mathematics Subject Classification[87]. Certain mathematical journals ask research paper authors to determine the subject classification code(s) for their papers. No algorithms are available to automate this process, so the subject classification is classed as a *manual* dimension. The subject nodes (coordinates) that exist and their location in the hierarchy are periodically reviewed by experts, rather than being determined by some algorithm, so the subject classification is classed as a *static* dimension.

Functionality that changes the structure of a library is *administration functionality*, while functionality that does not change the structure of a library is *user functionality*.

Adding and removing information units to a knowledge library is classed as user functionality. Adding and removing coordinates and dimensions is classed as administrator functionality.

3.5.1 Core knowledge library administration functionality

Core administration functionality includes the creation and destruction of knowledge libraries, dimensions and coordinates, and the specification of distances between coordinates.

3.5.2 Core knowledge library end use functionality

Core user functionality includes adding and removing information units to and from knowledge libraries, and displaying all manner of statistics relat-

ing to regions in subspaces in knowledge libraries—including the distance between such regions and retrieving information elements in regions.

3.6 Knowledge Library Security

Security is an important issue for knowledge libraries. Assuming users can be authenticated by username and password, and that messages between users and knowledge libraries can be secured by public key encryption, the question discussed here is: *What knowledge library functionality should users have access to and how can this be regulated?*

Two classes of users—administrators and end users—have already been identified. For many applications more are required. For example, the “administrator” user class may need to be further subdivided. Perhaps into superusers, who have the ability to create and destroy knowledge libraries, and system administrators, who do not. It may also be important to allow some users to add information units to the library and prevent other users from doing so. Many further divisions are warranted for some applications. Ideally, super users should be able to:

1. create user classes;
2. specify precisely what functionality users belonging to the class can access;
3. add and remove users from user classes.

3.7 Extended Knowledge Library Functionality

This section discusses knowledge library functionality that, while important, is not regarded as essential. An outline of how this functionality can be provided is given.

3.7.1 Automatic dimensions

Information units added to a knowledge library are automatically associated with coordinates in automatic dimensions.

Example 3.2. If the information units are text documents and the coordinates of a dimension are sets of keywords then we can automatically attach each document to the coordinate that is the set of all indexed keywords that appear in the text of the document.

Example 3.3. If the information units are addresses and the coordinates of one dimension are latitude/longitude pairs and we could use a geographic information system to automatically attach addresses to coordinates.

Algorithms also exist for classifying faces [73], voices [74], fingerprints [62] and digital images [53].

3.7.2 Dynamic dimensions

In dynamic dimensions the coordinates—and the distances between coordinates—themselves are automatically determined from the information elements in the library.

Example 3.4. Text documents are the information elements of an information space. The space has a “keyword” dimension, the coordinates of which are sets of keywords chosen from an “index”. The keywords that are in this index are selected from the text of the information elements of the space. Keywords that occur too frequently or infrequently are not included in the index, so adding a new document to the space can result in the addition or removal of keywords from the index, and coordinates from the keyword dimension.

Formal concept analysis (see section 4.2.5) could also be used to create dynamic dimensions, the coordinates of which are formal concepts.

3.7.3 Automatic report generation

Information elements are attached to points in information space. Regions in information space—sets of points—contain sets of information elements. Points are n -tuples of coordinates. Coordinates in precedence dimensions are partially ordered. Because information elements are associated with coordinates, the partial ordering of a precedence dimension can be used to partially order information elements. These partial orderings can be used to present information in a sequence.

Example 3.5. A company uses an information space to organise a large project. This project space has a “time” precedence dimension. Coordinates in this dimension are date/times. For any pair of coordinates x, y in this dimension, x is dependent on y if y is “earlier” than x . Descriptions of completed jobs are attached to points in the space, and so, to coordinates in the time dimension. The client is interested in recent progress. All points in the space with a coordinate in the time dimension “after” a certain time are selected. The attached completed job descriptions are retrieved and are presented to the client in time order. Thus the information space is used to generate a report on recent progress.

Example 3.6. An information space is used to organise a set of mathematical theorems. The information elements of the space are mathematical theorems and corresponding proofs. Many of these theorem–proofs are dependent upon other theorem–proofs in the space. This is modelled using a “requires” precedence dimension. The mathematical knowledge of individual users is modelled by regions in the space. Users are familiar with all theorems attached to points in their knowledge space. A user is interested in a particular theorem–proof that is outside of their knowledge space. All coordinates in the “requires” dimension that are outside of the user’s knowledge space and upon which the theorem–proof is dependent are selected. All theorem–proofs attached to these coordinates are retrieved and presented to the user in “requires” order. Thus the information space is used to generate

a report on all—currently unknown—required background knowledge.

Note that this example illustrates automatic report generation, *the automatic generation of background knowledge requirements*, and *the automatic generation of study sequences*.

3.7.4 Automatic notification

Users may want to be notified whenever certain events occur (or fail to occur). For example, users may want to be notified whenever information units are attached to points in particular regions of a space. Similarly, users may want to be notified when coordinates in particular dimensions are deleted. It is also possible that users may want to be notified when events fail to occur—for example, if an information unit is not attached to a point in a particular region of a space (by a certain time).

There are a number of methods that can be used to implement this functionality.

One way, is to record descriptions of all “events” that occur. The notifications a particular user requires are also recorded as part of their user profile. When a notification request is received from a user (perhaps when the user “logs on”) all events that have occurred since the last notification request are compared with the notification requirements in the user’s profile. A message containing details of the event is sent to the user if a match is found.

Another way is to associate notification requests with regions in information space. Whenever an event occurs that affects a region, a check of notification requests associated with that region is made. If a request is “triggered”, a message containing details of the event is sent to the user (who posted the notification request).

3.7.5 Knowledge library graphical user interface

An interactive map of an information space could serve as a graphical user interface for a knowledge library². Such a map may have a column displaying coordinates in each dimension currently of interest—perhaps drop-down menus could be used to select dimensions. These coordinates should be selectable—either by a mouse click, or perhaps by typing a formula into a text box. Another column displays a list of information elements attached to points in the region that the user has selected. A mouse click on an information element “opens” the element—displaying the information it contains. Other controls/buttons/menus allow users—with appropriate privileges—to perform other function, such as attaching information units to the space or creating coordinates etc.

If the coordinates in each dimension are ordered, pairs of dimension can be selected to provide the axes for scatter graphs that plot points with attached information units. Viewing n dimensions in this way requires $(n^2 - n)/2$ scatter graphs. Displays such as this can be used to provide an intuitive feel for the distribution of information elements in an information space.

3.8 What this chapter achieved

This chapter described Knowledge Libraries and outlined some of their advantages. Key Knowledge Library terms were introduced and the core functionalities of Knowledge Libraries were identified. In this way, this chapter motivates and guides the further development of Knowledge Libraries in later chapters.

²Here the term “map” is used in the specialised sense as defined in scenario 1, section 3.3.

Chapter 4

Spaces for Information Organisation

4.1 Overview

This chapter reviews existing “spaces” in the literature. The suitability of these spaces for providing a mathematical basis for Knowledge Libraries is assessed. Because it is well known and has precisely those properties intuitively expected of a space, metric space is chosen as the starting point for this research. The metric properties are, however, not sufficiently general, so the more general **set space** is defined.

Section 4.2 consists of definitions and critiques of **metric space**, **vector space** and the **vector model for information retrieval**, **lattices** and **topological space** and **formal concept analysis**, and the **relational model**. Various attempts at a mathematical basis for **online analytical processing** were also reviewed, but no good candidates were found. This may be due to confusion relating to how a context of meaning relates a mathematical model to its subject amongst online analytical processing model authors.

Section 4.3 summarises the suitability of these spaces to provide the mathematical basis for Knowledge Libraries and outlines the way forward that has been chosen.

Section 4.4 defines the function $G_p^{d_1 \dots d_n}$ which is used both to compare n -tuples, and in the definition of n -dimensional space. This section also defines \subseteq for n -tuples and discuss how contexts of meaning relate n -dimensional spaces to the subjects they model. A dimension selection operator is also defined, which is used to give examples of **permutation** and **projection**.

Section 4.5 discusses how spaces can be nested inside one another. In this section, **dimension nesting**, **subspace**, **based on** and **set distance function** are defined. The section shows how set distance functions can be **based on** other distance functions.

Section 4.6 discusses how the properties of sensible set distance functions differ to the properties of sensible non-set distance functions. It is discussed how the d' -**span** of a set may be defined.

Section 4.7 contrasts distance functions between sets of n -tuples with distance functions between n -tuples of sets. This section defines the $G_p^{d_1 \dots d_n}$ -**span** points in n -dimensional spaces.

Section 4.8 defines **the generalised triangle inequality**, denoted $G\Delta I$, and **set space**.

Section 4.9 discusses and defines other properties of set distance functions— \subseteq -**reflexivity**, $\not\subseteq$ -**strict positiveness** and $\not\subseteq^d$ -**strict positiveness**.

Section 4.10 discusses signed distance functions and corresponding (unsigned) distance functions. This research is normally only concerned with the properties of (unsigned) distance functions.

Section 4.11 briefly summarises what has been achieved in this chapter.

4.2 Potential Mathematical Bases

This section reviews, and in some cases extends, various mathematical definitions from the literature that could potentially provide a mathematical basis for information space. This section should provide the reader with an adequate understanding of the relevant mathematics.

4.2.1 Metric Space

The metric space properties are those most mathematicians would expect a distance function to have.

Metric space definition. *A metric space is a pair $\langle M, d \rangle$ where M is a set and d is a function, called a **metric**, which maps pairs of elements of M into $\mathbb{R}^{\geq 0}$. The following axioms are satisfied by d :*

1. *Reflexivity* $(\forall x \in M) d(x, x) = 0$.
2. *Symmetry* $(\forall x, y \in M) d(x, y) = d(y, x)$.
3. *Strict Positiveness* $(\forall x, y \in M)(x \neq y \rightarrow d(x, y) > 0)$.
4. *Triangle Inequality* $(\forall x, y, z \in M) d(x, y) \leq d(x, z) + d(z, y)$.

Metrics have precisely those properties that are normally intuitively expected of a “spatial” point to point distance function. For this reason, the metric properties can serve as a useful “reality check” for other “spaces”. Various algorithms and data structures for searching metric spaces have been surveyed in [22].

This research uses $\Delta\mathbf{I}$ as shorthand for “the triangle inequality”. Furthermore, if a function d maps pairs of elements of a set M into \mathbb{R} , without necessarily having the other properties enumerated above, d will be called a **signed distance function** over M . If d maps pairs of elements of M into $\mathbb{R}^{\geq 0}$, d will be called a **distance function** over M . If $x, y \in M$ then $d(x, y)$ is the corresponding **distance** between x and y . Similarly, a pair $\langle M, d \rangle$, where d is a (signed) distance function over M , will be referred to simply as a **space**.

The **similarity** between two objects is a related concept. A **similarity function** over a set M maps pairs of elements of M into $\mathbb{R}^{\geq 0}$. A **sensible similarity function** is hereby defined as any similarity function sim where, for all $x, y \in M$:

1. $sim(x, y) = 0$ when x and y are completely “dissimilar”;

2. $\text{sim}(x, y)$ is strictly increasing as x and y become less “dissimilar”, more “similar”;
3. $\text{sim}(x, x) > \text{sim}(x, y), \text{sim}(y, x), 0$ for any $y \neq x$;
4. $\text{sim}(x, x) = \text{sim}(y, y) = 1$.

Note that a sensible similarity function must appropriately reflect the actual “dissimilarity” and “similarity” of the underlying objects. This is normally a non mathematical judgement—if the actual similarity were mathematically defined, there would be less need to define sim .

As $\text{sim}(x, y) = 0$ for “dissimilar” x, y , strict positiveness may not be satisfied. Similarly, as $\text{sim}(x, x) > 0$, reflexivity is not satisfied. So sensible similarity functions are never metrics. Indeed, $\text{sim}(x, y)$ should generally decrease as $d(x, y)$ increases.

4.2.2 Vector Space

Field definition. A field $(X, +, \cdot)$ is a set X together with addition $+: X \times X \rightarrow X$ and multiplication $\cdot: X \times X \rightarrow X$ operators.

Addition is associative and commutative; there is an identity element $\mathbf{0} \in X$ where, for all $x \in X$, $x + \mathbf{0} = x$; for each $x \in X$ there is inverse element $y \in X$ where $x + y = \mathbf{0}$.

Multiplication is commutative; it is distributive over addition so, for all $x, y, z \in X$, $x(y + z) = (xy) + (xz)$; there is an identity element $\mathbf{1} \in X$, with $\mathbf{1} \neq \mathbf{0}$, where $1x = x$ for all $x \in X$; each $x \in X$, other than $\mathbf{0}$ has an inverse element $y \in X$ where $xy = \mathbf{1}$.

Example 4.1. A common example of a field is \mathbb{R} with standard addition and multiplication. Note that \mathbb{N}_1 is *not* a field as not every element of \mathbb{N}_1 has a multiplicative inverse in \mathbb{N}_1 .

Vector Space definition. A **vector space** $(V, F, +, \cdot)$ is a set V and a field F , with **vector addition** $+: V \times V \rightarrow V$ and **scalar multiplication** $\cdot: F \times V \rightarrow V$ operators. The elements of V are called **vectors**.

Vector addition is associative and commutative; each $v \in V$ has an inverse element w ; there is an identity element $\mathbf{0} \in V$.

Scalar multiplication shares its identity element $\mathbf{1} \in F$ with field multiplication; it is distributive over vector and field addition, so $a(v + w) = av + aw$ and $(a + b)v = av + bv$ for all $a, b \in F$ and $v, w \in V$; and it is compatible with field multiplication, so $a(bv) = (ab)v$ for any $a, b \in F$ and $v \in V$. [63]

Example 4.2. A common example of a vector space is n -dimensional real coordinate space $(\mathbb{R}^n, \mathbb{R}, +, \cdot)$. Here $\mathbb{R}^n = \mathbb{R} \times \dots \times \mathbb{R}$. For all $(x_1, \dots, x_n), (y_1, \dots, y_n) \in \mathbb{R}^n$, $(x_1, \dots, x_n) + (y_1, \dots, y_n) = (x_1 + y_1, \dots, x_n + y_n)$ defines vector addition. For all $a \in \mathbb{R}$, $(x_1, \dots, x_n) \in \mathbb{R}^n$, $a(x_1, \dots, x_n) = (ax_1, \dots, ax_n)$ defines scalar multiplication. The identity element is $(0, \dots, 0)$.

4.2.3 The vector model for information retrieval

Importantly, n -tuples, called “vectors”, have been used as a mathematical basis for information retrieval. The use of the term “vector” in this context is questionable as the vector model defines no inverse or identity elements and, indeed, does not define or use scalar multiplication and vector addition.

Vector Model. n **index terms** (keywords) are associated with the n -dimensions of an $(\mathbb{R}^{\geq 0})^n$ coordinate space—one index term per dimension. Each **document** in the collection is associated with a “unit vector” $\mathbf{y} = (y_1, \dots, y_n)$ in the space where, for $1 \leq i \leq n$,

$$y_i = \frac{t_i f_i}{\sqrt{\sum_{j=1}^n (t_j f_j)^2}}.$$

Each **raw term frequency** f_i is the number of times the i^{th} index term

appears in the document and $t_i = \log(c/c_i)$ —where c is the total number of documents in the collection and c_i is the number of documents where $f_i > 0$ —is the **inverse document frequency**. Each index term must appear in at least one document, but not all documents, in the collection and each document must include at least one index term. User **queries**, consisting of index terms, are associated with unit vectors in a similar manner. The **similarity** between two unit vectors $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$ in the space is given by

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n x_i y_i.$$

[82, 83]

Note that this is the dot product $\mathbf{x} \cdot \mathbf{y}$. Furthermore, note that, if “vector magnitude” is defined so that $|\mathbf{x}| = \sqrt{\sum_{i=1}^n x_i^2}$ for any $\mathbf{x} = (x_1, \dots, x_n) \in (\mathbb{R}^{\geq 0})^n$ —as for Euclidian space—it can be appreciated that $\sqrt{\sum_{j=1}^n (t_j f_j)^2}$ is the “magnitude” of $(t_1 f_1, \dots, t_n f_n)$ where raw term frequencies have been weighted by their inverse document frequency. Normalising this gives the “unit vector” \mathbf{y} . Note that, in the vector model, as \mathbf{x} and \mathbf{y} are unit vectors, $\text{sim}(\mathbf{x}, \mathbf{y})$ is the cosine of the angle between \mathbf{x} and \mathbf{y} .

The inverse document frequency, commonly abbreviated *idf*, is intended to reflect how effective an index term is at differentiating between documents in the collection. Note that each t_1, \dots, t_n is positive and documents are associated with n -tuples (y_1, \dots, y_n) where at least one $y_i \neq 0$.

Properties of the vector model

It can be readily verified that, if a tuple (y_1, \dots, y_n) is generated from raw term frequencies f_1, \dots, f_n and another tuple (y_1^*, \dots, y_n^*) is generated from raw term frequencies f_1^*, \dots, f_n^* where, for $1 \leq i \leq n$, $f_i^* = a f_i$ then $(y_1, \dots, y_n) = (y_1^*, \dots, y_n^*)$ —which is what is required. This equality does not hold in the more general case where $f_i^* = a_i f_i$ which is also sensible. In general the *only* case where $(y_1, \dots, y_n) = (y_1^*, \dots, y_n^*)$ and not all of y_1, \dots, y_n are 0 is when $f_i^* = a f_i$ for $1 \leq i \leq n$.

Theorem 4.1 *If not all of y_1, \dots, y_n are 0 and $(y_1, \dots, y_n) = (y_1^*, \dots, y_n^*)$ then, for $1 \leq i \leq n$, $f_i^* = af_i$.*

PROOF. As $(y_1, \dots, y_n) = (y_1^*, \dots, y_n^*)$ we have, for $1 \leq i \leq n$,

$$\frac{t_i f_i}{\sqrt{\sum_{j=1}^n (t_j f_j)^2}} = \frac{t_i f_i^*}{\sqrt{\sum_{j=1}^n (t_j f_j^*)^2}}.$$

There is at least one i where $f_i \neq 0$. For each of these i there is an $a_i \in \mathbb{R}^{\geq 0}$ so that $a_i f_i = f_i^*$. From this, and the equation above

$$a_i^2 = \frac{\sum_{j=1}^n (a_j t_j f_j)^2}{\sum_{j=1}^n (t_j f_j)^2}.$$

But the RHS of this equation does not vary with i , so $a = a_i$. If f_i is zero so are y_i, y_i^* and f_i^* and so $af_i = f_i^*$.

For the following discussion, define $U^n = \{\mathbf{x} | \mathbf{x} \in (\mathbb{R}^{\geq 0})^n, |\mathbf{x}| = 1\}$, so U^n is the set of all unit vectors in $(\mathbb{R}^{\geq 0})^n$.

While *sim*, as defined in the vector model, is a sensible similarity function between “keyword weighting” unit vectors¹, it *does not* provide the basis for a sensible similarity function between documents as the similarities between these unit vectors do not necessarily reflect the similarities between the corresponding documents². Although sensible similarity functions are never metrics, we might hold out hope for the obvious “distance”

$$d_1(\mathbf{x}, \mathbf{y}) = 1 - \text{sim}(\mathbf{x}, \mathbf{y}).$$

Indeed, fairly obviously, $d_1(\mathbf{x}, \mathbf{x}) = 0$, $d_1(\mathbf{x}, \mathbf{y}) = d_1(\mathbf{y}, \mathbf{x})$ and $d_1(\mathbf{x}, \mathbf{y}) > 0$ whenever $\mathbf{x} \neq \mathbf{y}$ for $\mathbf{x}, \mathbf{y} \in U^n$. However $\triangle I$ does not hold for d_1 .

¹See the definition of *sensible similarity function* in section 4.2.1.

²See the discussion on keyword indexing of documents in section 2.6.2.

Example 4.3. $\mathbf{x} = (1, 0)$, $\mathbf{y} = (0, 1)$, $\mathbf{z} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Now $d_1(\mathbf{x}, \mathbf{y}) = 1$ where as $d_1(\mathbf{x}, \mathbf{z}) = d_1(\mathbf{z}, \mathbf{y}) = 1 - \frac{1}{\sqrt{2}}$. For ΔI to hold we require $1 \leq 1 - \frac{1}{\sqrt{2}} + 1 - \frac{1}{\sqrt{2}}$ which is $2 \leq \sqrt{2}$.

This suggests that a better choice for a “vector model” distance function would be, for any $\mathbf{x}, \mathbf{y} \in U^n$,

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{1 - \sin(\mathbf{x}, \mathbf{y})^2}.$$

This is the sine of the angle between \mathbf{x} and \mathbf{y} . Again, fairly obviously, $d_2(\mathbf{x}, \mathbf{x}) = 0$, $d_2(\mathbf{x}, \mathbf{y}) = d_2(\mathbf{y}, \mathbf{x})$ and $d_2(\mathbf{x}, \mathbf{y}) > 0$ whenever $\mathbf{x} \neq \mathbf{y}$. Because the sine of the angle between two vectors in U^n obviously satisfies ΔI it should be clear that d_2 also satisfies ΔI . From this it can be concluded that $\langle U^n, d_2 \rangle$ is a metric space.

A vector model for composite documents

There are numerous variants of the vector model. Significantly, if a *composite document* consists of a number of different fields then an n -dimensional $(\mathbb{R}^{\geq 0})^n$ coordinate space can be set up for each field. This allows users to choose to retrieve documents using index terms from particular fields. Similarity scores from different fields can be combined (with appropriate weights) to give an overall similarity [34, 29].

Example 4.4. All the (composite) documents in a collection consist of the fields “title”, “abstract”, “body” and “references”. These documents are represented by tuples such as $\mathbf{y} = (y_1, \dots, y_n, \dots, y_{2n}, \dots, y_{3n}, \dots, y_{4n})$. The terms y_1, \dots, y_n are calculated from words in the document’s title, the terms y_{n+1}, \dots, y_{2n} are calculated from words in the document’s abstract etc—in a similar manner to the terms in the vector model. Also, $\mathbf{y}_{title} = (y_1, \dots, y_n)$, $\mathbf{y}_{abstract} = (y_{n+1}, \dots, y_{2n})$ etc. so users can choose to retrieve documents using index terms from only document titles, or abstracts etc. The contributions of each field can also be weighted to give an overall query \mathbf{x} to composite

document \mathbf{y} similarity such as

$$\begin{aligned} \text{sim}(\mathbf{x}, \mathbf{y}) = \\ 0.5\text{sim}(\mathbf{x}, \mathbf{y}_{\text{title}}) + 0.3\text{sim}(\mathbf{x}, \mathbf{y}_{\text{abstract}}) + 0.15\text{sim}(\mathbf{x}, \mathbf{y}_{\text{body}}) + 0.05\text{sim}(\mathbf{x}, \mathbf{y}_{\text{references}}). \end{aligned}$$

Note the weights in this example sum to unity which, although not apparently a formal requirement, seems reasonable.

Critique of the vector model

The inherent limitations in using context free keywords to organise information has already been discussed (see section 2.6.2). As the vector model relies on this same method, it suffers from the same limitations.

This research has also discussed the limitations of automatic indexing (see sections 2.3.3 and 2.6.2). In the vector model, the inverse document frequency $t_i = \log(c/c_i)$ varies with the total number c of documents in the collection and the number of documents c_i where the raw frequency of the i^{th} index term is greater than 0. So the vector associated with a document depends on the documents already in the collection. This vector will change as documents are added to the collection and certainly should not be expected to be the same for the same document in different collections.

This problem with *indexing consistency* can be solved by using a “reference collection” to determine the inverse document frequencies for all index terms. Different collections can be “synchronised” with each other in this way. Of course the problem of choosing a reference collection that will remain appropriate over time remains.

“Vector addition” and scalar multiplication could be defined so that, for any $\mathbf{x}, \mathbf{y} \in (\mathbb{R}^{\geq 0})^n$ and $s \in \mathbb{R}^{\geq 0}$, denoting $\mathbf{x} = (x_1, \dots, x_n)$, $\mathbf{y} = (y_1, \dots, y_n)$:

1. $\mathbf{x} + \mathbf{y} = (x_1 + y_1, \dots, x_n + y_n)$ and
2. $s\mathbf{x} = (sx_1, \dots, sx_n)$

as one would expect.

Now vector addition can be used to connect query \mathbf{x} and document \mathbf{y} “unit vectors”: $\mathbf{x} + \mathbf{z} = \mathbf{y}$. Solving this for \mathbf{z} gives a “distance vector” $\mathbf{y} - \mathbf{x}$ with a magnitude no greater than $\sqrt{2}$. This suggests the distance function

$$d_3(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{y} - \mathbf{x}|}{\sqrt{2}}.$$

We have $d_3(\mathbf{x}, \mathbf{x}) = 0$, $d_3(\mathbf{x}, \mathbf{y}) = d_3(\mathbf{y}, \mathbf{x})$ and $d_3(\mathbf{x}, \mathbf{y}) > 0$ whenever $\mathbf{x} \neq \mathbf{y}$ for any $\mathbf{x}, \mathbf{y} \in (\mathbb{R}^{\geq 0})^n$. Because the Euclidian distance between points in U^n obviously satisfies ΔI , it should be clear that d_3 also satisfies ΔI . So $\langle U^n, d_3 \rangle$ is a metric space.

Another important question is *what is it that the vector model actually models!?* It is not a very good model of the “information space” of relations between documents. Even if a good model of this space could be constructed with context free index terms, it should have a stronger relation between index terms with similar meanings. Indeed the generalised vector space model (GVSM), which correlates index terms, has demonstrated improved precision and recall rates for some collections[102]—albeit at the expense of model simplicity and indexing response time.

Some researchers have adopted the view that the vector model is a model of the *process* of information retrieval itself—a “vector *processing* model”[84, 82]. This view makes little sense though. Surely the utility of the vector model for information retrieval is proportional to how accurately it reflects the actual relations between queries and documents.

The appeal of the vector model is that it:

1. is fairly simple,
2. can be used to automatically index collections,
3. automatically matches queries and documents, and it
4. has response time, precision and recall rates that are comparable or better to other established automatic indexing and retrieval techniques [7].

However, because the vector model does not accurately reflect the actual relations between queries and documents—and indeed between “user needs” and queries—it is doomed to give “approximate” results.

4.2.4 Lattices and Topological Space

In order to develop the notion of formal concept analysis, lattice and finite topological space are defined. Lattices are also part of the mathematical basis for L -fuzzy sets, which are discussed in section 6.2.6. Also, “networked space”, which is defined in section 8.3, is related to these concepts.

Lattice definition. A lattice (L, \leq) is a set L with a partial ordering \leq where every pair $x, y \in L$ has

1. a **join** $z \in L$, denoted $z = x \vee y$, where $x, y \leq z$ and $z \leq z'$ for any $z' \in L$ where $x, y \leq z'$.
2. a **meet** $z \in L$, denoted $z = x \wedge y$, where $z \leq x, y$ and $z' \leq z$ for any $z' \in L$ where $z' \leq x, y$. [100]

Note that a set with a partial ordering is a *partially ordered set*—often abbreviated *poset*. So a lattice is a type of poset.

Finite Topological Space definition. A finite topological space is a finite set of sets T where:

1. $\emptyset \in T$.
2. If $X, Y \in T$, $X \cup Y \in T$.
3. If $X, Y \in T$, $X \cap Y \in T$.

[19]³

³Although in this reference, a topological space is a pair (T, A) where T is a set of subsets of A —including \emptyset and A . As A is always $\bigcup T$, this notation seems somewhat redundant.

It should be clear that these definitions are closely related. Indeed, if we define a partial ordering \leq on a finite topological space T so that, for all $X, Y \in T$, $X \leq Y \iff X \subseteq Y$ then (T, \leq) is a lattice.

4.2.5 Formal Concept Analysis

Importantly, lattices have been used as part of the mathematical basis for formal concept analysis—abbreviated FCA. FCA, introduced by Rudolf Wille in 1982 in [99], is a method of data analysis, representation and organisation. FCA requires “formal contexts” and “concept lattices”.

Formal Context definition. A **formal context** is a triple $(\mathbf{O}, \mathbf{A}, R)$ where \mathbf{O} is a set of **objects**, \mathbf{A} is a set of **attributes** and $R \subseteq \mathbf{O} \times \mathbf{A}$ is a relation where xRy iff an object $x \in \mathbf{O}$ has attribute $y \in \mathbf{A}$ [99, 100].

Concept Lattice definition. Given a formal context $(\mathbf{O}, \mathbf{A}, R)$

$$O^A = \{x \in \mathbf{O} \mid \text{for all } y \in A, xRy\}$$

for some $A \subseteq \mathbf{A}$. Similarly

$$A^O = \{y \in \mathbf{A} \mid \text{for all } x \in O, xRy\}$$

for some $O \subseteq \mathbf{O}$. A **concept lattice** is a lattice (L, \leq) where

$$L = \{(O^{A^O}, A^O) \mid O \subseteq \mathbf{O}\}$$

is a set of **formal concepts**. O is the **extent** and A is the **intent** of each $(O, A) \in L$. For all $(O_1, A_1), (O_2, A_2) \in L$, $(O_1, A_1) \leq (O_2, A_2)$ iff $O_1 \subseteq O_2$ (or equivalently $A_1 \supseteq A_2$) [99, 100].

Note that O^A is the set of all objects in \mathbf{O} that have all the attributes in A . Similarly, A^O is the set of all attributes in \mathbf{A} common to all objects in O .

It can be shown that:

- (1) $O^A = O^{A^O} \iff A^{O^A} = A^O$;
 (2) for each O^{A^O} there is an O^A where $O^A = O^{A^O}$ and
 (3) for each A^{O^A} there is an A^O where $A^O = A^{O^A}$.
 So $L = \{(O^{A^O}, A^O) | O \subseteq \mathbf{O}\} = \{(O^A, A^{O^A}) | A \subseteq \mathbf{A}\}$.

The **join** of $(O_1, A_1), (O_2, A_2) \in L$ is $(O_1, A_1) \vee (O_2, A_2) = (O_1 \cup O_2, A_1 \cap A_2)$.

The **meet** of $(O_1, A_1), (O_2, A_2) \in L$ is $(O_1, A_1) \wedge (O_2, A_2) = (O_1 \cap O_2, A_1 \cup A_2)$.

If $L = \{(O_1, A_1), \dots, (O_n, A_n)\}$, the **supremum** of L is

$$\bigvee_{i=1}^n (O_i, A_i) = (\bigcup_{i=1}^n O_i, \bigcap_{i=1}^n A_i)$$

and the **infimum** of L is

$$\bigwedge_{i=1}^n (O_i, A_i) = (\bigcap_{i=1}^n O_i, \bigcup_{i=1}^n A_i).$$

Clearly $\{O | (O, A) \in L\}$ and $\{A | (O, A) \in L\}$ are topological spaces.

Example 4.5. Figure 4.1 below is a Hasse diagram that depicts a concept lattice for objects consisting of the integers from 1 to 10, and attributes composite (c), square (s), even (e), odd (o) and prime (p). Line segments connect formal concepts. Iff x and y are formal concepts connected by one or more line segments and x is below y in the diagram, $x \leq y$. The supremum is at the top of the diagram, while the infimum is at the bottom.

FCA has been applied to the problem of information retrieval [59]. In this approach, documents are indexed just as they are in the Boolean model for information retrieval⁴ except *documents* are called *objects* and *index terms* are called *attributes*. The retrieval process steps users through formal concepts until they arrive at one with a manageably small extent.

So FCA has been applied to developing *user interfaces*, rather than indexing models, for information retrieval. Although Hasse diagrams are useful

⁴Essentially a vector model where the coordinates in document and query vectors are all either 0 or 1.

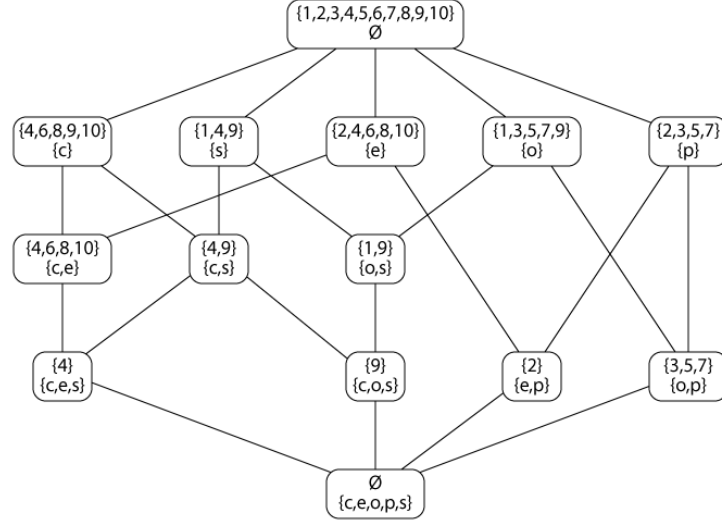


Figure 4.1: A Hasse diagram of a concept lattice of objects = $\{1, \dots, 10\}$ and attributes = $\{\text{composite, even, odd, prime, square}\}$.

when applied to very small collections, when applied to larger collections they rapidly become unmanageable.

FCA, just like other systems that automatically generate indexes, does not address the problem of indexing consistency—the formal concepts that index the collection appear and disappear as documents are added and removed. However this problem is limited as formal concepts only disappear when there are no documents in the collection with corresponding attributes. So concept lattices maintain *access path consistency*.

Given a concept lattice (L, \leq) , an obvious function to use for a formal concept distance would be, for all $x, y \in L$,

$$d(x, y) = \begin{cases} 0, & y \leq x. \\ 1, & y \not\leq x \end{cases}$$

Now the distance between a “query” concept $x = (O_1, A_1)$ and a “document indexing” concept $y = (O_2, A_2)$ is 0 iff $A_1 \subseteq A_2$ and 1 iff $A_1 \not\subseteq A_2$. So documents that have a superset of the attributes specified in the query will “match”—which is what is desired. For any $x \in L$, $d(x, x) = 0$ so d is reflexive. It is also easy to show that ΔI holds. However d is neither sym-

metric nor strict positive as $d(x, y) = 0$ and $d(y, x) = 1$ when $A_1 \subseteq A_2$ and $A_1 \neq A_2$.

4.2.6 The Relational Model

The relational model, which provides the mathematical basis for relational databases, was defined by Codd in 1970 in [24]. Given an n -ary relation $R \subseteq S_1 \times \dots \times S_n$, S_j is called the j^{th} **domain** of R for $1 \leq j \leq n$, that is, R is of **degree** n . A dot notation is used to distinguish between domains of different relations, so $Q.S_1$ refers to the domain S_1 of a relation Q whereas $R.S_1$ refers to the domain S_1 of a relation R .

A **data bank** is a set of (time-varying) relations. If a domain is a set of relations (in the data bank) then the relations are called **nonatomic values** of the domain. Otherwise the domain elements are called **(atomic) values**.

A **selection** operator π_T —where $T = i_1 \dots i_k$ with $i_1 \leq n, \dots, i_k \leq n$ and $k \leq n$ —is defined so that $(s_{i_1}, s_{i_2}, \dots, s_{i_k}) \in \pi_T(R) \iff (s_1, \dots, s_n) \in R$. So the selection operator gives a relation $\pi_T(R)$ that is a **permutation** (domain reordering) and/or **projection** (domain removal) of R .

A **primary key** of R is a set of domains that uniquely identifies each element of R . By this, Codd means if there is a bijective mapping from R to $\pi_{i_1 \dots i_k}(R)$, $\{S_{i_1}, S_{i_2}, \dots, S_{i_k}\}$ is a primary key of R . If R has more than one primary key, one is arbitrary selected to be **the primary key**⁵. A **foreign key** of R is a set of domains that is not the primary key of R , but is the primary key of some other relation (in the data bank).

A **natural join** operator $*$ is defined so that $Q * R = \{(a, b, c) | aQb \text{ and } bRc\}$ for binary relations Q, R . **3-join**, for binary relations P, Q, R , is defined so that $P * Q * R = \{(a, b, c, d) | aPb \text{ and } bQc \text{ and } cRd\}$. The **tie** of R is defined by $\gamma(R) = \{(s_1, \dots, s_{n-1}) | (s_1, \dots, s_n) \in R \text{ and } s_1 = s_n\}$. The **natural cyclic 3-join** of P, Q, R is $\gamma(P * Q * R)$.

For every n -ary relation R , where $n > 2$, there is a corresponding binary relation pR where $(s_1, \dots, s_p) {}^pR (s_{p+1}, \dots, s_n) \iff (s_1, \dots, s_n) \in R$. This

⁵Primary keys with fewer domains are preferred in practice. The potentially confusing terminology **primary key** and **the primary key** is due to Codd.

allows the above operations on binary relations to be extended to n -ary relations.

Finally the **restriction** of Q by R , denoted $Q_T|_UR$ where

1. Q is a relation of degree n and R is a relation of degree m ,
2. $T = i_1 \dots i_k$ with $i_1 \leq n, \dots, i_k \leq n$ and similarly,
3. $U = j_1 \dots j_k$ with $j_1 \leq m, \dots, j_k \leq m$ and
4. $k \leq \min\{n, m\}$,

is defined so that $Q_T|_UR = Q'$ where Q' is the largest subset of Q such that $\pi_T(Q') = \pi_U(R)$.

Example 4.6.

P			Q				R		
S_1	S_2		S_1	S_2			S_1	S_2	
1	1		2	2			3	3	
2	1		1	3			3	2	

$P * Q$			$P * Q * R$				$\pi_{413}(P * Q * R)$		
S_1	S_2	S_3	S_1	S_2	S_3	S_4	S_1	S_2	S_3
1	1	3	1	1	3	3	3	1	3
1	1	3	1	1	3	2	2	1	3
2	1	3	2	1	3	3	3	2	3
			2	1	3	2	2	2	3

$\gamma(P * Q * R)$			$(P * Q * R)_{14 13}(P * Q)$			
S_1	S_2	S_3	S_1	S_2	S_3	S_4
2	1	3	1	1	3	3
			2	1	3	3

Table 4.1: Eight relations illustrating operations defined in [24]

In the table, the domains S_1, \dots, S_4 should be read as being relation qualified, so $P.S_1 \neq Q.S_1$ for example. Omitting the dot notation, $\{S_1\}$ and $\{S_1, S_2\}$ are primary keys of P ; $\{S_1\}$, $\{S_2\}$ and $\{S_1, S_2\}$ are primary keys of

Q ; $\{S_1, S_4\}$, $\{S_1, S_2, S_4\}$, $\{S_1, S_3, S_4\}$ and $\{S_1, S_2, S_3, S_4\}$ are primary keys of $P * Q * R$. Note that $P * Q * R$ is isomorphic to $(^2(P * Q)) * R$.

In 1974 Chamberlin and Boyce extended Codd’s work, developing a “Structured Query Language”—SQL for short⁶—to support user interaction with relational databases[21]. Many commercial database systems implement SQL. The American National Standards Institute (ANSI) has been publishing standards for SQL since 1986 [51]. The International Organisation for Standardisation (ISO) ratified this standard in 1987 and has been publishing standards for SQL since then[37]. Quite clearly, the relational model is the mathematical foundation for a large industry based around relational database technology.

Significantly, the relational model *does not* define any distance functions between the elements (n -tuples) of n -ary relations. An obvious distance function would be, for all $x, y \in R$,

$$d(x, y) = \begin{cases} 0, & x=y \\ 1, & x \neq y. \end{cases}$$

Now $d(x, x) = 0$, $d(x, y) = d(y, x)$, $d(x, y) > 0$ for $x \neq y$ and it is easy to show that d satisfies ΔI . So $\langle R, d \rangle$ is a metric space.

4.2.7 Online Analytical Processing (OLAP)

The term OLAP was coined in [25] and [26]. Unfortunately mathematical definitions were not provided. There have been a number of attempts at providing a mathematical foundation for OLAP since, including—but by no means limited to—[57, 5, 97, 46, 31]. Unfortunately, these authors treat a mathematical model as a kind of mathematical equivalent of a database. This is not as it should be! A mathematical model of a subject should describe the significant relationships between only the *absolutely essential* “bones” of the subject.

OLAP model authors have failed to identify the “bones” of OLAP and

⁶Originally “Structured English Query Language”, with the acronym SEQUEL.

include “dimension names”, “attribute names” etc. in their models. This is misguided and results in unnecessary complications. Rather than being part of the model, “dimension names”, “attribute names” etc. should be part of a description—a *context of meaning*—that relates the model to its subject. The failure of OLAP model authors in this respect is inexplicable as Codd correctly leaves “domain names” out of his relational model in [24].

4.3 Space

Vector space is quite general and well known. However, for the purposes of this research, scalar multiplication is redundant and the vector addition properties—while suitable enough—do not express the characteristics information space should have.

Lattices and topological spaces do not have a function that can be used to determine the distance between their elements—the distance function defined when discussing formal concept analysis in section 4.2.5 is not sufficiently general. These spaces also seem somewhat artificial⁷.

The relational model also lacks a function for comparing relations—although given such a function, n -ary relations can be treated as sets of points in n -dimensional space.

Metric space is well known and has precisely those properties intuitively expected of a “space”. Of the “spaces” surveyed, metric space seems like the best choice to provide a mathematical basis for Knowledge Libraries. However, as will be seen, the metric properties will need to be generalised.

Note that this thesis has developed a general definition for space, in section 4.2.1. That is, a **space** is a pair $\langle M, d \rangle$ where d is a (signed) distance function over M . Unfortunately this definition is not consistent with all uses of the term in the literature.

⁷Because $\mathcal{P}(X)$ is often used to define the distance between subsets of X , “set spaces” (see section 4.8) are often discrete topological spaces. Also “networked spaces” (see section 8.3)—though not directly related—are somewhat reminiscent of lattices.

4.4 *n*-Dimensional Spaces

As will be seen, points in information space are *n*-tuples. To find distances between such *n*-tuples, *n* spaces can be combined together into a single, *n*-dimensional space.

$G_p^{d_1 \dots d_n}$ **definition.** *If, for $1 \leq r \leq n$, $\langle M_r, d_r \rangle$ is a space and $x_r, y_r \in M_r$*

$$G_p^{d_1 \dots d_n}((x_1, \dots, x_n), (y_1, \dots, y_n)) = \left(\sum_{r=1}^n (d_r(x_r, y_r))^p \right)^{\frac{1}{p}}.$$

In this thesis, $G_p^{d_1 \dots d_n}$ is **based on** d_1, \dots, d_n as the function $G_p^{d_1 \dots d_n}$ requires the functions d_1, \dots, d_n to be defined.

$G_p^{d_1 \dots d_n}$ is often an appropriate choice for a distance function between *n*-tuples. $G_2^{d_1 \dots d_n}$ is the well known "Euclidian" distance function. $G_1^{d_1 \dots d_n}$ has a number of names including the "Manhattan" or "block" distance function and the "taxicab metric." $\lim_{p \rightarrow \infty} (G_p^{d_1 \dots d_n})$, which can also be written as $G_\infty^{d_1 \dots d_n}$ is known as the Chebyshev distance function or the "uniform metric." More generally, $G_p^{d_1 \dots d_n}$ is known as the Minkowski distance function, or the L^p distance function (when this different notation is used).

***n*-dimensional space definition.** *If, for $1 \leq r \leq n$, each $\langle M_r, d_r \rangle$ is a space and each $x_r \in M_r$ then (x_1, \dots, x_n) is a **point** in the ***n*-dimensional space** $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ and each x_r is a **coordinate**. A **region** is a set of points. $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ has **dimensions** $\langle M_1, d_1 \rangle, \dots, \langle M_n, d_n \rangle$.*

In order to generalise the properties discussed in section 4.9 to extend to distance functions over *n*-dimensional spaces, \subseteq must be extended over *n*-tuples.

\subseteq **for *n*-tuples definition.** $(X_1, \dots, X_n) \subseteq (Y_1, \dots, Y_n)$ *iff, for $1 \leq r \leq n$, $X_r \subseteq Y_r$ or $X_r = Y_r$.*

Note that, from this definition, $(x_1, \dots, x_n) \subseteq (x_1, \dots, x_n)$ even if each x_r is not a set.

The order of dimensions in an n -dimensional space carries significance for two reasons.

1. Order may be used to ascribe a *context of meaning* to dimensions.
2. The sets M_1, \dots, M_n should appear in the same order as the distance functions d_1, \dots, d_n .

So a space $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ in one context of meaning may be effectively the same as a another space where the sets M_1, \dots, M_n —along with the distance functions d_1, \dots, d_n —are combined in some other order.

Example 4.7. Consider the 3-dimensional space $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$ made up of a “width” dimension $\langle M_1, d_1 \rangle$, a “depth” dimension $\langle M_2, d_2 \rangle$ and a “height” dimension $\langle M_3, d_3 \rangle$. Order ascribes a context of meaning: for any point $(x_1, x_2, x_3) \in M_1 \times M_2 \times M_3$ we know that x_1 is a “width” value, x_2 is a “depth” value and x_3 is a “height” value.

This space is effectively the same as a space $\langle M_3 \times M_2 \times M_1, G_2^{d_3 d_2 d_1} \rangle$ where, as before, $\langle M_3, d_3 \rangle$ is a “height” dimension, $\langle M_2, d_2 \rangle$ is a “depth” dimension and $\langle M_1, d_1 \rangle$ is a “width” dimension. For any point $(x_3, x_2, x_1) \in M_3 \times M_2 \times M_1$ we know that x_3 is a “height” value, x_2 is a “depth” value and x_1 is a “width” value.

In this research, the order in which the sets M_1, \dots, M_n are combined in an n -ary Cartesian Product is considered to be essentially arbitrary—as long as the distance functions are combined in the same order and the context of meaning is appropriate. Indeed a *selection* operator can be used to permute dimensions and project spaces just as Codd uses π_T to permute domains and project relations.

Dimension Selection definition. *If:*

1. $T = t_1 \dots t_k$ where $t_1 \leq n, \dots, t_k \leq n$ and $k \leq n$; and

2. $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ is an n dimensional space

then $\pi_T(M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n}) = \langle M_{t_1} \times M_{t_2} \times \dots \times M_{t_k}, G_p^{d_{t_1} d_{t_2} \dots d_{t_k}} \rangle$.

Permutation definition. *A space $\langle X_1 \times \dots \times X_n, G_p^{d_1 \dots d_n} \rangle$ is a **permutation** of a space $\langle Y_1 \times \dots \times Y_n, G_p^{d_1^* \dots d_n^*} \rangle$ if for each $1 \leq i \leq n$ there is a $1 \leq j \leq n$, not paired with any other i , where $X_i = Y_j$ and $d_i = d_j^*$.*

Less formally, a space x is a permutation of a space y iff y can be transformed into x by reordering, but not adding or removing, dimensions.

Example 4.8. $\langle M_3 \times M_2 \times M_1, G_2^{d_3 d_2 d_1} \rangle = \pi_{321} \langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$. The space $\langle M_3 \times M_2 \times M_1, G_2^{d_3 d_2 d_1} \rangle$ is a permutation of the space $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$.

Projection definition. *A space $\langle X_1 \times \dots \times X_k, G_p^{d_1 \dots d_k} \rangle$ is a **projection** of a space $\langle Y_1 \times \dots \times Y_n, G_p^{d_1^* \dots d_n^*} \rangle$ if there is a monotonic (order preserving) nondecreasing function $f : \{1, \dots, k\} \rightarrow \{1, \dots, n\}$, defined for each $1 \leq i \leq k$, where if $f(i) = j$, $X_i = Y_j$ and $d_i = d_j^*$.*

Less formally, a space x is a projection of a space y iff y can be transformed into x by removing and/or repeating, but not reordering, any of the dimensions. A space is a projection of itself.

Example 4.9. $\langle M_1 \times M_3, G_2^{d_1 d_3} \rangle = \pi_{13} \langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$. The space $\langle M_1 \times M_3, G_2^{d_1 d_3} \rangle$ is a projection of the space $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$.

Note that the space $\langle M_3 \times M_1, G_2^{d_3 d_1} \rangle = \pi_{31} \langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$ is *not* a projection of the space $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$ as the dimensions are out of order. We can say that $\langle M_3 \times M_1, G_2^{d_3 d_1} \rangle$ is a **permuted projection**—or a **projected permutation**—of $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$.

Each point in a projected space **corresponds to** a number of points in the original space whereas each point in a permuted space corresponds to precisely one point in the original space.

Example 4.10. A space $\langle X \times Y \times Z, G_2^{d_1 d_2 d_3} \rangle$ has a projection $\langle X \times Y, G_2^{d_1 d_2} \rangle$ and a permutation $\langle Z \times X \times Y, G_2^{d_3 d_1 d_2} \rangle$. A point (x, y) in the projected space corresponds to any point (x, y, z) , for any $z \in Z$, in the original space. A point (z, x, y) in the permuted space corresponds to the point (x, y, z) in the original space.

4.5 Nested Spaces

There are a number of ways in which information spaces can be contained or *nested* in other information spaces. Most obviously, information spaces can be information elements of other information spaces. Similarly, dimensions or even coordinates of information spaces can be information elements of other information spaces. Other types of nesting are also possible.

Dimension Nesting definition. A k -dimensional space $\langle M_i, d_i \rangle$, where $M_i = Y_1 \times \dots \times Y_k$ and $d_i = G_q^{d_{n+1} \dots d_{n+k}}$, is **dimension nested** in another space $\langle M, G_p^{d_1 \dots d_n} \rangle$ if $\langle M_i, d_i \rangle$ is a dimension of $\langle M, G_p^{d_1 \dots d_n} \rangle$.

Spaces can also be subspaces of other spaces.

Subspace definition. A space $\langle X, d \rangle$ is a **subspace** of another space $\langle Y, d^* \rangle$ iff $X \subseteq Y$ and, for all $x, y \in X$, $d(x, y) = d^*(x, y)$.

Note that, if $\langle X_1 \times \dots \times X_n, d \rangle$ is a subspace of $\langle Y_1 \times \dots \times Y_n, d^* \rangle$, it follows from the definition of \subseteq for n -tuples in section 4.4 that $X_1 \subseteq Y_1, \dots, X_n \subseteq Y_n$.

Another type of nesting occurs when a distance function d between elements of M is used to define a distance function d' between subsets of M . Indeed, it is often appropriate to use a distance function d over a set M of properties of information units to define a distance function d' between in-

formation units, where each information unit is represented by a set $Y \subseteq M$ of properties. For this reason, this research discusses using spaces $\langle M, d \rangle$ to define spaces $\langle M', d' \rangle$ where $M' \subseteq \mathcal{P}(M)$. Normally it is appropriate if $d'(\{x\}, \{y\}) = d(x, y)$.

Based on definition. *If*

1. $d'(X, Y)$ is a function of $\{d(x, y) | x \in X, y \in Y\}$ or
2. $d'(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$, is a function of $d(x_i, y_i)$ (for $0 \leq i \leq n$),

d' is **based on** d .

Example 4.11. For all $X, Y \subseteq M$, $d'(X, Y) = \min_{x \in X, y \in Y} d(x, y)$. So d' is based on d .

Because of fundamental characteristics of information, this type of nesting can be many levelled.

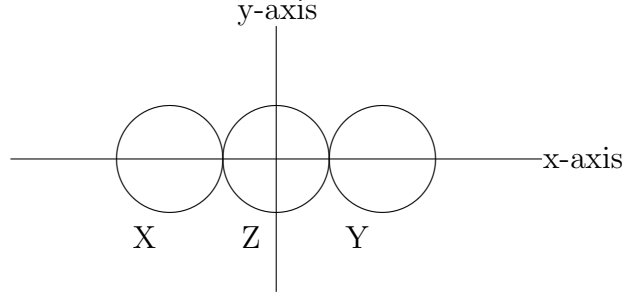
Example 4.12. A distance function d'''' , between Universities, is based on a distance function d''' , between research groups, is based on a distance function d'' , between researchers, is based on a distance function d' , between research papers, is based on a distance function d , between topics...

It is important that this type of many levelled nesting is allowed for.

Set distance definition. *If d is a distance function over a set M of sets then d is a set distance function.*

4.6 Span

Sensible “spatial” set distance functions tend to require more general properties than metrics. The simplest way to illustrate this is with an example.

Figure 4.2: Three balls X, Z, Y in \mathbb{R}^2 .

Example 4.13. If $d_1(x_1, x_2) = |x_1 - x_2|$ for all $x_1, x_2 \in \mathbb{R}$ and $d_2(y_1, y_2) = |y_1 - y_2|$ for all $y_1, y_2 \in \mathbb{R}$ then $\langle \mathbb{R} \times \mathbb{R}, G_2^{d_1 d_2} \rangle$ is a two dimensional Euclidean space where $G_2^{d_1 d_2}((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Clearly d_1, d_2 and $G_2^{d_1 d_2}$ all satisfy $\triangle I$. We define the distance between any two regions $X, Y \subseteq \mathbb{R} \times \mathbb{R}$ as

$$d(X, Y) = \min_{(x_1, y_1) \in X, (x_2, y_2) \in Y} G_2^{d_1 d_2}((x_1, y_1), (x_2, y_2)).$$

We have three balls:

$$X = \{(x, y) | x, y \in \mathbb{R}, (x + 2)^2 + y^2 \leq 1\},$$

$$Z = \{(x, y) | x, y \in \mathbb{R}, x^2 + y^2 \leq 1\} \text{ and}$$

$$Y = \{(x, y) | x, y \in \mathbb{R}, (x - 2)^2 + y^2 \leq 1\}.$$

As illustrated in figure 4.2. Now $d(X, Y) = 2$ while $d(X, Z) = d(Z, Y) = 0$. So the distance function d does not satisfy $\triangle I$.

This example shows that sensible set distance functions may not satisfy $\triangle I$, even when they are based on distance functions that do satisfy $\triangle I$. From the example it should be clear that the *diameter* of the intermediate set Z should be taken into account. Doing so gives the inequality:

$$d(X, Y) \leq d(X, Z) + \max_{(x_1, y_1) \in Z, (x_2, y_2) \in Z} G_2^{d_1 d_2}((x_1, x_2), (y_1, y_2)) + d(Z, Y).$$

From the example, $d(X, Y) = 2$, $d(X, Z) = d(Z, Y) = 0$ as before and $\max_{(x_1, y_1) \in Z, (x_2, y_2) \in Z} G_2^{d_1 d_2}((x_1, x_2), (y_1, y_2)) = 2$. So, as $2 \leq 0 + 2 + 0$, the

generalised triangle inequality that takes the diameter of the intermediate set into account is satisfied in this case.

It can be verified that this inequality is satisfied for *any* $X, Z, Y \subseteq \mathbb{R} \times \mathbb{R}$. Indeed, just as simpler distance functions should satisfy ΔI , sensible set distance functions should satisfy a generalised triangle inequality, such as the one above.

Some caution is required however. For a generalised triangle inequality is to be a test that discriminates between sensible and absurd distance functions, it must be ensured that the “diameters” added to the RHS are reasonable. Without this constraint absurd distance functions can be made to satisfy the inequality simply by defining equally absurd “diameters”.

If d is a non-set distance function over M and d' is a set distance function—based on d —over $\mathcal{P}(M)$ then the diameter of $X \in \mathcal{P}(M)$ should be something like

$$\text{diam}_{d'}(X) = \max_{x,y \in X} d(x, y).$$

However if the elements of M are sets and d is a set distance function then the “diameter”s of x and y should also be taken into account. This “generalised diameter” will be called a set’s **span**. The d' -span of X will be denoted by $s_{d'}(X)$. The function $s_{d'}$ will be defined differently for different d' s. However, in general, $s_{d'}(X)$ should be something like

$$s_{d'}(X) = \max_{x,y \in X} \{0, s_d(x) + d(x, y) + s_d(y)\}.$$

If $s_d(x) = 0$ for all $x \in M$, as might be the case if d is a non-set distance function, then $s_{d'}(X)$ simplifies to $\max_{x,y \in X} d(x, y)$ —the diameter of X . If $s_d(x)$ —the d -**span** of an object x —is not otherwise defined, $s_d(x) = 0$ is assumed. As will be seen later, it is important that all spans are ≥ 0 .

Note that one tempting “solution” to example 4.6 is to define $d(X, Y) = G_2^{d_1 d_2}((x_1, x_2), (y_1, y_2))$ where (x_1, x_2) is the centroid of X and (y_1, y_2) is the centroid of Y . With this definition, following the example, $d(X, Z) = d(Z, Y) = 2$, $d(X, Y) = 4$ and d would clearly satisfy ΔI . Unfortunately this “centroid distance” is not \subseteq -reflexive, or $\not\subseteq$ -strict positive (see section 4.9),

which are desirable properties. Briefly, we want $d(X, Y) = 0$ iff $X \subseteq Y$ and $d(X, Y) > 0$ iff $X \not\subseteq Y$.

4.7 Spans of Points in n -Dimensional Spaces

Example 4.13 took two non-set distance functions d_1 and d_2 , which were used to define a third— $G_2^{d_1 d_2}$ over $\mathbb{R} \times \mathbb{R}$. The function $G_2^{d_1 d_2}$ was used to define a set distance function d . So in the example, d was a distance function between two *sets of pairs* of reals—coordinates in $\mathbb{R} \times \mathbb{R}$. Alternatively d_1 and d_2 could have been used to define two set distance functions d_1' and d_2' , both over $\mathcal{P}(\mathbb{R})$ and then used these distance functions combined to give $G_2^{d_1' d_2'}$ —a distance function between two *pairs of sets* of reals.

Example 4.14. For all $x_1, x_2 \in \mathbb{R}$, $d_1(x_1, x_2) = |x_1 - x_2|$. For all $y_1, y_2 \in \mathbb{R}$, $d_2(y_1, y_2) = |y_1 - y_2|$. For all $X_1, X_2 \in A$ where $A = \mathcal{P}(\mathbb{R})$,

$$d_1'(X_1, X_2) = \min_{x_1 \in X_1, x_2 \in X_2} d_1(x_1, x_2).$$

For all $Y_1, Y_2 \in B$ where $B = \mathcal{P}(\mathbb{R})$,

$$d_2'(Y_1, Y_2) = \min_{y_1 \in Y_1, y_2 \in Y_2} d_2(y_1, y_2).$$

Now $G_2^{d_1' d_2'}((X_1, Y_1), (X_2, Y_2)) = \sqrt{d_1'(X_1, X_2)^2 + d_2'(Y_1, Y_2)^2}$.

$G_2^{d_1' d_2'}(X, Y)$ gives the distance between $X, Y \in \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})$ —pairs of sets (rather than sets of pairs). Note that each of the regions $X, Y \in \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})$ consist of one or more “rectangular” components—it is impossible to define spheres this way.

Example 4.15. We want to define $X, Z, Y \in \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R})$ to be as similar as possible to the corresponding regions in example 4.13. The best we can do is:

$X = (\{-3, -1\}, \{-1, 1\})$, $Z = (\{-1, 1\}, \{-1, 1\})$, $Y = (\{1, 3\}, \{-1, 1\})$.

These examples should clarify the contrast between spaces of sets of n -tuples and spaces of n -tuples of sets. If the elements of an n -tuple can be sets, the n -tuple should have a span that is influenced by the span of each of its elements.

$s_{G_p^{d_1 \dots d_n}}(\mathbf{x})$ (**the $G_p^{d_1 \dots d_n}$ -span of \mathbf{x}**) **definition.** If $\mathbf{x} = (x_1, \dots, x_n)$ and, for $1 \leq r \leq n$, $x_r \in X_r$ and d_r is a distance function over X_r then

$$s_{G_p^{d_1 \dots d_n}}(\mathbf{x}) = \left(\sum_{r=1}^n (s_{d_r}(x_r))^p \right)^{\frac{1}{p}}.$$

In this definition, spans in each of the n spaces—treated in the same way as distance functions—give a span in the n -dimensional space. If $s_d(x_r) = 0$ for $1 \leq r \leq n$ —as is sensible if each d_r were a non-set distance function—then $s_{G_p^{d_1 \dots d_n}}(\mathbf{x}) = 0$.

4.8 The Generalised Triangle Inequality and Set Space

The notion of span allows us to introduce a generalisation of Δ I.

The Generalised Triangle Inequality (G Δ I) definition. If d is a distance function over a set M , $d(x, y) \leq d(x, z) + s_d(z) + d(z, y)$ for each $x, y, z \in M$.

The definition of G Δ I allows the definition of a generalised version of metric space: set space.

Set space definition. $\langle M, d \rangle$ is a **set space** if M is a set and d is a distance function that satisfies G Δ I over M .

Note that if $\langle M, d \rangle$ is a set space, d is only a set distance function if the elements of M are themselves sets and d is defined in terms of distances

between the elements of the elements of M .

Note also that a *set space distance function*—which is a distance function that satisfies $G\Delta I$ —is *not* the same thing as a *set distance function*—which is a distance function d over a set of sets. Because of the deceptive similarity of these two terms, the term *set space distance function* is avoided in this research.

4.9 Other Properties of Set Distance Functions

Just as ΔI is not general enough for set distance functions, neither is reflexivity, symmetry, nor strict positiveness!

Example 4.16. Let X be a set of *desired* attributes and Y be the set of attributes of some object (each chosen from a universe set of potential object attributes). A distance of 0 between X and Y signifies that Y has all the attributes X . We would like $d(X, Y) = 0$ whenever $X \subseteq Y$ —*not* only when $X = Y$ —so d more than meets the reflexive requirements but does not meet the strict positive requirements. If $X \subseteq Y$ and $X \neq Y$ we would like $d(X, Y) = 0$ but $d(Y, X) > 0$ —so d does not meet the symmetry requirements.

4.9.1 \subseteq -Reflexivity

Example 4.16 shows that reflexivity ($d(x, x) = 0$) is not strong enough for set distance functions and suggests the following *strengthened* form of reflexivity.

\subseteq -Reflexivity definition. *If, for all $X, Y \in M$ where $(X \subseteq Y$ or $X = Y)$, $d(X, Y) = 0$ then $\langle M, d \rangle$ is \subseteq -reflexive. If, for all M over which d is defined, $\langle M, d \rangle$ is \subseteq -reflexive, d is \subseteq -reflexive.*

Note that, if \subseteq is *not* defined over the elements of M (the elements of M are *not* sets or n -tuples) and for all $x \in M$, $d(x, x) = 0$ then $\langle M, d \rangle$ is

\subseteq -reflexive. Thus if the elements of M are *not* sets or n -tuples $\langle M, d \rangle$ is \subseteq -reflexive iff $\langle M, d \rangle$ is reflexive!

Also note that, if d is \subseteq -reflexive, d is reflexive.

Example 4.17. For all $x, y \in \mathbb{R}$, $d(x, y) = |x - y|$. For all $X, Y \in \mathcal{P}(\mathbb{R}) - \{\emptyset\}$ $d'(X, Y) = \min_{x \in X, y \in Y} d(x, y)$. If $X \subseteq Y$ then $x \in X \Rightarrow x \in Y$ and $d(x, x) = 0$ so $d'(X, Y) = 0$ and $\langle \mathcal{P}(\mathbb{R}) - \{\emptyset\}, d' \rangle$ is \subseteq -reflexive.

Note that if:

1. for all $x, y \in \mathbb{R}$, $d(x, y) = |x - y|$ and
2. for all $X, Y \in \mathcal{P}(\mathbb{R}) - \{\emptyset\}$, $d'(X, Y) = \max_{x \in X, y \in Y} d(x, y)$ and
3. $X = \{1\}$ and $Y = \{1, 2\}$

then $X \subseteq Y$ but $d'(X, Y) = 1$, so $\langle \mathcal{P}(\mathbb{R}) - \{\emptyset\}, d' \rangle$ is *not* \subseteq -reflexive.

4.9.2 $\not\subseteq$ -Strict Positiveness

As illustrated in example 4.16, a version of strict positiveness that requires $d(X, Y) > 0$ whenever $X \not\subseteq Y$ is required.

$\not\subseteq$ -Strict Positiveness definition. *If for all $X, Y \in M$ where $X \not\subseteq Y$ and $X \neq Y$, $d(X, Y) > 0$ then $\langle M, d \rangle$ is $\not\subseteq$ -strict positive. If, for all M over which d is defined, $\langle M, d \rangle$ is $\not\subseteq$ -strict positive, d is $\not\subseteq$ -strict positive.*

Note that if $d(x, y) > 0$ whenever $x \neq y$ then d is $\not\subseteq$ -strict positive—no matter how $\not\subseteq$ is defined!

Example 4.18. For all $x, y \in \mathbb{R}$, $d(x, y) = |x - y|$. As $d(x, y) > 0$ whenever $x \neq y$ (and so $d(x, y) > 0$ whenever $x \not\subseteq y$ and $x \neq y$ —no matter how $\not\subseteq$ is defined), $\langle \mathbb{R}, d \rangle$ is $\not\subseteq$ -strict positive.

Example 4.19. For all $x, y \in \mathbb{R}$, $d(x, y) = |x - y|$. For all $X, Y \in \mathcal{P}(\mathbb{R}) - \{\emptyset\}$ $d'(X, Y) = \max_{x \in X, y \in Y} d(x, y)$. If $X \not\subseteq Y$ there is an $x \in X$ and a $y \in Y$ where $x \neq y$. $x \neq y \Rightarrow d(x, y) > 0$ and $d'(X, Y) > 0$ so $\langle \mathcal{P}(\mathbb{R}) - \{\emptyset\}, d' \rangle$ is

$\not\subseteq$ -strict positive.

Note that if:

1. for all $x, y \in \mathbb{R}$, $d(x, y) = |x - y|$ and
2. for all $X, Y \in \mathcal{P}(\mathbb{R}) - \{\emptyset\}$, $d'(X, Y) = \min_{x \in X, y \in Y} d(x, y)$ and
3. $X = \{1, 2\}$ and $b = \{1\}$

then $X \not\subseteq Y$ but $d'(X, Y) = 0$ so $\langle \mathcal{P}(\mathbb{R}) - \{\emptyset\}, d' \rangle$ is *not* $\not\subseteq$ -strict positive.

There is a problem with $\not\subseteq$ -strict positiveness however.

Example 4.20. $X = \{1\}$, $Y = \{1, 2\}$. For \subseteq -reflexive d' , $d'(X, Y) = 0$ as $X \subseteq Y$. Now we want $d''(\{X\}, \{Y\}) = 0$ because $\{1\} \subseteq \{1, 2\} \in \{Y\}$ and so $\{Y\}$ (ultimately) contains the desired attribute. But because $\{X\} \not\subseteq \{Y\}$ and $\{X\} \neq \{Y\}$, $\not\subseteq$ -strict positiveness requires $d''(\{X\}, \{Y\}) > 0$.

4.9.3 $\not\subseteq^d$ -strict positiveness

In order to correct the problem, illustrated in example 4.20 above, with $\not\subseteq$ -strict positiveness, a “recursive” form of \subseteq is defined.

$\subseteq^{d'}$ definition. $X \subseteq^{d'} Y$ iff $X = Y$ or $X \subseteq Y$ or d' is based on d and for all $x \in X$, there is a $y \in Y$ where $x \subseteq^d y$. If $\mathbf{x} = (x_1, \dots, x_n)$, $\mathbf{y} = (y_1, \dots, y_n)$ are n -tuples and $d = G_p^{d_1 \dots d_n}$ then $\mathbf{x} \subseteq^d \mathbf{y}$ iff $\mathbf{x} = \mathbf{y}$ or, for $1 \leq i \leq n$, $x_i \subseteq^{d_i} y_i$.

This allows the definition of a suitable recursive form of strict positiveness.

$\not\subseteq^d$ -strict positiveness definition. If for all $X, Y \in M$ where $X \not\subseteq^d Y$, $d(X, Y) > 0$ then $\langle M, d \rangle$ is **$\not\subseteq^d$ -strict positive**. If, for all M over which d is defined, $\langle M, d \rangle$ is $\not\subseteq^d$ -strict positive, d is $\not\subseteq^d$ -strict positive.

Example 4.21. Following example 4.20, because $\{\{1\}\} \subseteq^{d''} \{\{1, 2\}\}$, we do not require $d''(\{\{1\}\}, \{\{1, 2\}\}) > 0$ for d'' to be $\not\subseteq^{d''}$ -strict positive.

It is tempting to define a corresponding recursive form of reflexivity, \subseteq^d -reflexivity. However the author prefers to remain with \subseteq -reflexivity because \subseteq^d -reflexivity leads problems with the $\mathcal{M}_k d$ distance function, introduced in chapter 7, where things work better if the definition of \subseteq^d is slightly altered⁸. As they stand, \subseteq -reflexivity and \mathcal{Q}^d -strict positiveness are sufficiently general to be applied to all the distance functions we discuss. The theorems and proofs provided can be readily adapted for more specific forms of reflexivity and strict positiveness.

4.10 Signed Distances

As will be seen in chapter 9, sometimes a **signed distance function** is required to adequately express the distance between elements.

Example 4.22. M is a set of “heights” $d(x, y) = y - x$ gives the distance between the height x and the height y . A negative distance indicates that x is greater than y , so the distance is “downwards”. A positive number indicates that x is less than y , so the distance is “upwards”.

All signed distance functions $d(x, y)$ have a corresponding (unsigned) distance function $|d(x, y)|$. When assessing the suitability of a signed distance function, this research only requires that the corresponding (unsigned) distance function satisfies $G\Delta I$ and is \mathcal{Q} -strict positive and singleton symmetric. The property \subseteq -reflexivity is always satisfied by all signed distance functions with a corresponding (unsigned) distance function that satisfies \subseteq -reflexivity.

4.11 What this Chapter Achieved

This chapter initiated the development of the mathematical basis for Knowledge Libraries. A review of the relevant literature justified the choice of metric space as the starting point for this development. The metric space

⁸See section 7.5.2.

properties were identified as being too strong, and suitable weaker properties were defined. The concept of “span” was introduced. The function $G_p^{d_1 \dots d_n}$, which is useful for multi dimensional spaces was defined. Notions such as dimension nesting, projected spaces and subspace were developed.

Dimension nesting will be used to create “multi faceted” spaces, as discussed in section 2.3.1. Projected spaces will be used to allow users to select only those dimensions of interest, when querying Knowledge Libraries. Subspaces will be useful when implementing Knowledge Library Security, as discussed in section 3.6.

Chapter 5

Set Spaces

5.1 Overview

Section 5.2 opens this chapter with a discussion of various ways in which set spaces can be manipulated while retaining their properties. Significantly, n set spaces can be combined into an **n -dimensional set space**. The combined space preserves any ΔI , $G\Delta I$, reflexive, strict positive, symmetric, \subseteq -reflexive or $\not\subseteq^d$ -strict positive properties common to all of the component spaces. This is useful because we want our information spaces to be n -dimensional spaces.

The section shows that n -dimensional set spaces with a k -dimensional set space dimension can be “flattened out” into $(n + k - 1)$ -dimensional set spaces.

Set spaces can also **dilated** by multiplying all the distances by a weight w . The dilated space preserves any ΔI , $G\Delta I$, reflexive, strict positive, symmetric, \subseteq -reflexive or $\not\subseteq^d$ -strict positive properties. This means that different weights can be assigned to each of the n dimensions of an n -dimensional space as these weighted dimensions are themselves spaces. Practically, this allows users to assign greater or lesser importance to individual dimensions of an information space.

It is also possible to **translate** a set space by adding a handicap h to all the distances. However, if this is done, reflexive and \subseteq -reflexive properties

are not preserved.

Section 5.3 discusses how set operations can be used to define distance functions over $\mathcal{P}(M)$. The distance function $d(X, Y) = |X| - |X \cap Y|$ for all $X, Y \subseteq M$ satisfies $\triangle I$, is \subseteq -reflexive and $\not\subseteq^d$ -strict positive.

Section 5.4 discusses how a distance function d over a set M can be used to define a distance function d_{ij}^M over a set $\mathcal{P}(M)$. Here i and j can be any number between 0 and 100—which works like a percentage—or “av” which takes an average distance. If d satisfies $\triangle I$ over M then $d_{100\ 0}^M, d_{0\ 100}^M, d_{100\ 100}^M, d_{av\ 100}^M, d_{100\ av}^M$ and $d_{av\ av}^M$ satisfy $\triangle I$ over $\mathcal{P}(M)$. The d_{ij}^M -**span** is defined. If d satisfies $G\triangle I$, all of the d_{ij}^M distance functions satisfy $G\triangle I$. If d is reflexive and strict positive, only $d_{av\ 0}^M(x, y)$ and $d_{100\ 0}^M(x, y)$ are both \subseteq -reflexive and $\not\subseteq^{d_{ij}}$ -strict positive.

Section 5.5 briefly summarises what has been achieved in this chapter.

5.2 Manipulating Set Space

This section discusses how n set spaces can be combined to form an n -dimensional set space, how k -dimensional spaces can be dimensions in n -dimensional spaces, and how set spaces can be dilated and translated.

5.2.1 n -Dimensional Set Spaces

It is possible to form a single n -dimensional set space from n dimensions, each of which is a set space. The proof of this relies on Minkowsky’s inequality which states that if $t_r \leq u_r + v_r$ and $t_r, u_r, v_r \geq 0$ then

$$\left(\sum_{r=1}^n t_r^p \right)^{\frac{1}{p}} \leq \left(\sum_{r=1}^n u_r^p \right)^{\frac{1}{p}} + \left(\sum_{r=1}^n v_r^p \right)^{\frac{1}{p}}.$$

Theorem 5.1 *If $\langle M_1, d_1 \rangle, \dots, \langle M_n, d_n \rangle$ are set spaces and $M \subseteq M_1 \times \dots \times M_n$ then $\langle M, G_p^{d_1 \dots d_n} \rangle$ is a set space.*

PROOF. Let $(x_1, \dots, x_n), (y_1, \dots, y_n), (z_1, \dots, z_n) \in M$ and, for $1 \leq r \leq n$

$$t_r = d_r(x_r, y_r),$$

$$u_r = d_r(x_r, z_r),$$

$$v_r = d_r(z_r, y_r),$$

$$s_r = s_{d_r}(z_r)$$

By the G Δ I

$$t_r \leq u_r + s_r + v_r$$

and by Minkowsky's inequality, applied twice

$$\begin{aligned} \left(\sum_{r=1}^n t_r^p \right)^{\frac{1}{p}} &\leq \left(\sum_{r=1}^n u_r^p \right)^{\frac{1}{p}} + \left(\sum_{r=1}^n (s_r + v_r)^p \right)^{\frac{1}{p}} \\ &\leq \left(\sum_{r=1}^n u_r^p \right)^{\frac{1}{p}} + \left(\sum_{r=1}^n s_r^p \right)^{\frac{1}{p}} + \left(\sum_{r=1}^n v_r^p \right)^{\frac{1}{p}}. \end{aligned}$$

That is

$$G_p^{d_1 \dots d_n}((x_1, \dots, x_n), (y_1, \dots, y_n)) \leq$$

$$G_p^{d_1 \dots d_n}((x_1, \dots, x_n), (z_1, \dots, z_n)) + s_{G_p^{d_1 \dots d_n}}((z_1, \dots, z_n)) + G_p^{d_1 \dots d_n}((z_1, \dots, z_n), (y_1, \dots, y_n))$$

so G Δ I holds for $\langle M, G_p^{d_1 \dots d_n} \rangle$.

5.2.2 Other Properties of n -Dimensional Spaces

If $d = G_p^{d_1 \dots d_n}$ and $M \subseteq M_1 \times \dots \times M_n$, it is easily shown that if the set spaces $\langle M_i, d_i \rangle$, for $1 \leq i \leq n$:

1. are reflexive, $\langle M, d \rangle$ is reflexive.
2. are symmetric, $\langle M, d \rangle$ is symmetric.
3. are strict positive, $\langle M, d \rangle$ is strict positive.
4. satisfy Δ I, then $\langle M, d \rangle$ satisfies Δ I. (Theorem 5.1 with each $s_r = 0$.)

5. are \subseteq -reflexive, $\langle M, d \rangle$ is \subseteq -reflexive.
6. are \mathcal{Z}^d -strict positive, $\langle M, d \rangle$ is \mathcal{Z}^d -strict positive.

Note that the \subseteq -reflexive and \mathcal{Z}^d -strict positive properties apply because \subseteq and \subseteq^d has been defined over n -tuples (see sections 4.4 and 4.9.3).

5.2.3 Dimension Nesting

The following theorem shows that a number of dimensions of a space can be formed into a *dimension nested* space without altering the functional characteristics of the space.

Theorem 5.2 *If the k -dimensional space $\langle M_1, d_{k+1} \rangle$, where $M_1 = Y_1 \times \dots \times Y_k$ and $d_{k+1} = G_p^{d_1 \dots d_k}$ is dimension nested in $\langle M, G_p^{d_{k+1} \dots d_{k+n}} \rangle$, where $M = M_1 \times \dots \times M_n$, then $\langle M, G_p^{d_{k+1} \dots d_{k+n}} \rangle$ is isomorphic to a $(k+n-1)$ -dimensional space $\langle Z, G_p^{d_1 \dots d_k d_{k+2} \dots d_{n+k}} \rangle$.*

PROOF. If $Z = Y_1 \times \dots \times Y_k \times M_2 \times \dots \times M_n$ and $x_1 = (y_1, \dots, y_k)$ then it is clear that each point $\mathbf{x} = (x_1, \dots, x_n) \in M$ has a corresponding point $\mathbf{z} = (y_1, \dots, y_k, x_2, \dots, x_n) \in Z$.

Furthermore, if $\mathbf{x}' = (x'_1, \dots, x'_n) \in M$ etc.

$$G_p^{d_1 \dots d_n}(\mathbf{x}, \mathbf{x}') = G_p^{d_1 \dots d_k d_{k+2} \dots d_{k+n}}(\mathbf{z}, \mathbf{z}') = \left(\sum_{r=1}^k (d_r(y_r, y'_r))^p + \sum_{r=2}^n (d_{k+r}(x_r, x'_r))^p \right)^{\frac{1}{p}}.$$

Note that the above proof shows that an n -dimensional space where one dimension, $\langle M_1, d_{k+1} \rangle$, is itself a k -dimensional space ($d_{k+1} = G_p^{d_1 \dots d_k}$ and $M_1 = Y_1 \times \dots \times Y_n$), is in fact isomorphic to a $(k+n-1)$ -dimensional space $\langle Z, G_p^{d_1 \dots d_k d_{k+2} \dots d_{k+n}} \rangle$. In effect the n -dimensional space can be “expanded” or “normalised” into a corresponding isomorphic $(n+k-1)$ -dimensional space. Equally, each $(k+n-1)$ -dimensional space has a corresponding isomorphic n -dimensional space (with a nested k -dimensional space).

5.2.4 Dilated and Translated Spaces

Set spaces can be dilated and/or translated.

Dilated Space definition. *If $w \in \mathbb{R}^{>0}$ then $\langle M, wd \rangle$ is the space $\langle M, d \rangle$ after it has been **dilated** by w .*

Translated Space definition. *If $h \in \mathbb{R}^{>0}$ then $\langle M, d + h \rangle$ is the space $\langle M, d \rangle$ after it has been **translated** by h .*

Fairly obviously, the span of each element in a space that has been dilated by w should be weighted by w . Less obviously, the spans of translated spaces do not need to be altered.

$s_{wd+h}(x)$ (**the $wd + h$ -span of x**) **definition.**

$$s_{wd+h}(x) = ws_d(x).$$

Note that, in this definition, x can be any object for which $s_d(x)$ is defined.

Now if $d(X, Y) \leq d(X, Z) + s_d(Z) + d(Z, Y)$ and $w, h \in \mathbb{R}^{>0}$,

$$wd(X, Y) + h \leq wd(X, Z) + h + ws_d(Z) + wd(Z, Y) + h$$

and so $G\Delta I$ holds for dilated and translated spaces¹.

It can also be easily shown that any reflexive, symmetric, strict positive, ΔI , \subseteq -reflexive or $\not\subseteq^d$ -strict positive properties of $\langle M, d \rangle$ will be preserved in $\langle M, wd \rangle$.

Similarly, $\langle M, d + h \rangle$ preserves any symmetric, strict positive, ΔI or $\not\subseteq^{d+h}$ -strict positive properties of $\langle M, d \rangle$. Significantly, reflexive and \subseteq -reflexive properties are *not* preserved.

Different weights can be assigned to each of the n dimensions of an n -dimensional set space to reflect the greater or lesser importance of individual

¹Clearly, ΔI also holds for dilated and translated spaces.

dimensions.

Theorem 5.3 *If each dimension $\langle M_r, d_r \rangle$ of a set space $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ is dilated by w_r , this results in a set space $\langle M_1 \times \dots \times M_n, G_p^{w_1 d_1 \dots w_n d_n} \rangle$.*

PROOF. $G\Delta I$ follows as in the proof of Theorem 5.1 but with each d_r replaced by $w_r d_r$.

5.3 Set Distance Functions Based on Set Operations

Appealing set distance functions can be based on set operations. Consider:

$$d(X, Y) = |X \cup Y| - |X \cap Y|$$

where d is defined over any pair of sets.

In words this is “to calculate the distance between X and Y , count all the elements of X or Y , not in X and Y .”

Although it can be shown that $|X \cup Y| - |X \cap Y|$ satisfies ΔI and is $\not\subseteq^d$ -strict positive, it is not \subseteq -reflexive.

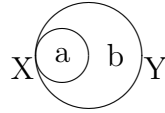


Figure 5.1: Subset element counts for \subseteq -reflexive proofs. $a = |X|$, $b = |Y - X|$.

Following the figure, if $d(X, Y) = |X \cup Y| - |X \cap Y|$, $X \subseteq Y$ then $d(X, Y) = a + b - a = b$. So \subseteq -reflexivity *fails* for $|X \cup Y| - |X \cap Y|$ because b may be greater than 0.

This suggests that a \subseteq -reflexive distance function of this type can be created by not counting elements of Y . Indeed, it can be readily verified that

$$d(X, Y) = |X| - |X \cap Y|$$

is \subseteq -reflexive as, following figure 5.1, $d - d = 0$. It can also be shown that $|X| - |X \cap Y|$ satisfies ΔI . Following figure 5.2, $b + f \leq (f + g) + (a + b)$. Similarly, following figure 5.3, it can be shown that $|X| - |X \cap Y|$ is \mathcal{Z}^d -strict

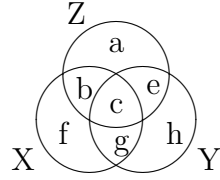


Figure 5.2: Subset element counts for ΔI proofs. $a = |Z - X - Y|$, $b = |Z \cap X - Y|$, $c = |Z \cap X \cap Y|$...

positive as $a + b - b > 0$.

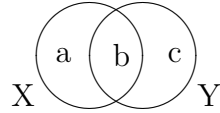


Figure 5.3: Subset element counts for \mathcal{Z}^d -strict positive proofs. $a = |X - Y|$, $b = |X \cap Y|$ and $c = |Y - X|$.

Note that, as d is *not* based on any other function, the \mathcal{Z}^d -strict positive and \mathcal{Z} -strict positive properties are equivalent.

Normalising by dividing through by $|X|$ seems quite reasonable, but

$$d(X, Y) = \frac{|X| - |X \cap Y|}{|X|} = 1 - \frac{|X \cap Y|}{|X|}$$

fails to satisfy ΔI and so is unsuitable for use as a set space distance function.

Example 5.1. $d(X, Y) = 1 - \frac{|X \cap Y|}{|X|}$. We want $d(X, Y) \leq d(X, Z) + d(Z, Y)$. If $|X| = 1$, $|Y| = 1$, $|X \cap Y| = 0$ and $Z = X \cup Y$ we have $1 \not\leq 0 + \frac{1}{2}$. So $1 - \frac{|X \cap Y|}{|X|}$ does not satisfy ΔI .

5.4 The d_{ij}^M Set Distance function

Dubuisson and Jain [32] define the “distance” from a point x to a set of points Y as $\min_{y \in Y} \|x - y\|$, where $\|x - y\|$ is the Euclidian distance between x and y .

They then define six distance functions on pairs of sets as follows.

$$\begin{aligned} d_1(X, Y) &= \min_{x \in X} \min_{y \in Y} \|x - y\|, & d_2(X, Y) &= {}^{50}K_{x \in X}^{th} \min_{y \in Y} \|x - y\|, \\ d_3(X, Y) &= {}^{75}K_{x \in X}^{th} \min_{y \in Y} \|x - y\|, & d_4(X, Y) &= {}^{90}K_{x \in X}^{th} \min_{y \in Y} \|x - y\|, \\ d_5(X, Y) &= \max_{x \in X} \min_{y \in Y} \|x - y\|, & d_6(X, Y) &= \sum_{x \in X} \min_{y \in Y} \frac{\|x - y\|}{|X|}. \end{aligned}$$

Dubuisson and Jain discuss four means of converting these directed distance functions into undirected distance functions such as

$$d(X, Y) = \min\{d_i(X, Y), d_i(Y, X)\},$$

thus creating twenty four undirected distance functions altogether. This research is focused mainly on the directed distance functions, however it is useful to correct, generalise and simplify the six distance functions above.

For Dubuisson and Jain ${}^jK_{x \in X}^{th}$ represents the K^{th} largest value of $\min_{y \in Y} \|x - y\|$, where $\frac{K}{|X|} = \frac{j}{100}$. If $j|X|$ is divisible by 100, K is the number of elements in the $j\%$ of X nearest to Y . If not, $\frac{K}{|X|} = \frac{j}{100}$ is not meaningful. Furthermore, if X or Y is the empty set, this is not defined.

To generalise, replace $\|x - y\|$ in the six distance function definitions above by $d(x, y)$ —an arbitrary distance function over $X \cup Y$.

To generalise and simplify distance functions d_{ij}^M are defined so that $d_{ij}^M(X, Y)$ can be not only $\min_{x \in X} \min_{y \in Y} d(x, y)$, ${}^{50}K_{x \in X}^{th} \min_{y \in Y} d(x, y)$ etc. but also $\min_{x \in X} \max_{y \in Y} d(x, y)$, ${}^{50}K_{x \in X}^{th} {}^{90}K_{x \in X}^{th} d(x, y)$ etc.²

²Chapter 7 defines and discuss d_{ij}^M —a distance function which gives the distance be-

Definitions for xY , d_j^M and X^Y are required in order to define d_{ij}^M . For these definitions the convention used is that $x \in X$, $y \in Y$ and $z \in Z$.

xy and xY definition. *Given a distance function d over a set $Y \cup \{x\}$ and a number $0 \leq j \leq 100$, ${}^xY \subseteq Y$ is any set where:*

1. *if $j \neq 0$, $|{}^xY| = \lceil \frac{j|Y|}{100} \rceil$,*
2. *if $j = 0$, ${}^xY = \{y\}$ and*
3. *for each $y \in {}^xY$, $d(x, y) \leq d(x, y')$ for all $y' \in Y - {}^xY$.*

xy is any element of xY where $d(x, {}^xy) = \max_{y \in {}^xY} d(x, y)$.

Informally, xY can be built-up by starting from \emptyset and adding elements $y \in Y$ in turn—beginning with those having the smallest $d(x, y)$ to those with the largest—until $|{}^xY| > \frac{j}{100}$. xy is the last element we add.

Informally, xY can be read as ‘the $j\%$ of Y that x is d -closest to’.

Note that, while purists may prefer notation like ${}^xY_{dj}$ and ${}^xy_{dj}$ to indicate that d and j are independent variables, a less formal, simplified notation has been used in this research as d and j are always clear from the context.

Similarly, although there may be more than one xY for the same Y , it happens that $\max_{y \in {}^xY} d(x, y)$ is always the same for the same Y, d, j, x , so—for our purposes—all elements xy and sets xY produce the same result.

tween L -collections, rather than just sets. This is a further generalisation.

$d_j^M(x, Y)$ (**where** $M \neq \emptyset$, $Y \subseteq M$ **and** $d(x, y)$ **is defined for all** $y \in M$) **definition.**

1. $d_j^M(x, \emptyset) = \max_{y \in M} d(x, y).$

If $Y \neq \emptyset$:

2. *If* $0 \leq j \leq 100$, $d_j^M(x, Y) = d(x, {}^x y).$

3. $d_{\text{av}}^M(x, Y) = \frac{1}{|Y|} \sum_{y \in Y} d(x, y).$

This definition ensures that there is no $y \in M$ where $d_j^M(x, \emptyset) < d(x, y)$. It will be seen in section 5.4.1 that this will be important if ΔI , or $G\Delta I$ is to hold over a set of subsets of Z which may include the empty set.

Note that, if $j = \frac{100K}{|Y|}$ then $d_j^M(x, Y)$ is a K^{th} smallest $d(x, y)$ for $y \in Y$. That is $d_j^M(x, Y) = d(x, y_K)$ where y_K is the K^{th} element of Y —ranked by $d(x, y)$ for all $y \in Y$. Of course such a y_K need not be unique. In particular $d_0^M(x, Y) = \min_{y \in Y} d(x, y) = d(x, y_1)$ and $d_{100}^M(x, Y) = \max_{y \in Y} d(x, y) = d(x, y_{|Y|})$.

If $|Y|$ is odd, i.e. $|Y| = 2K + 1$, then $d_{50}^M(x, Y)$ is the $K + 1^{\text{th}}$ largest value of $d(x, y)$ for $y \in Y$ —i.e. the median value. If $|Y|$ is even, i.e. $|Y| = 2K$, then $d_{50}^M(x, Y)$ is the K^{th} largest value—usually just below the median value.

$d_{\text{av}}^M(x, Y)$ is the average distance between x and Y .

x^Y and X^Y definition. *Given a distance function d over $X \cup Y$, numbers $0 \leq i \leq 100$ and $0 \leq j \leq 100$, $X^Y \subseteq X$ is any set where:*

1. *if* $i \neq 0$, $|X^Y| = \lceil \frac{i|X|}{100} \rceil$,

2. *if* $i = 0$, $X^Y = \{x\}$ and

3. $(\forall x \in X^Y)(\forall x' \in X - X^Y) d_j^M(x, Y) \leq d_j^M(x', Y).$

x^Y is any element of X^Y where $d_j^M(x^Y, Y) = \max_{x \in X^Y} d_j^M(x, Y)$

Similarly to ${}^x Y$ the set X^Y can be built-up by starting from \emptyset and adding

elements $x \in X$ in turn—beginning with those having the smallest $d_j^M(x, y)$ to those with the largest—until $|X^Y| > \frac{i|X|}{100}$. Again, x^Y is the last element added.

Read X^Y as ‘the $i\%$ of X d_j^M -closest to Y .’ Again, just as for xY , there may be a number of X^Y s for the same X, d, i, j, Y , but $\max_{x \in X^Y} d_j^M(x, Y)$ will remain the same, so this is of no concern. Purists may prefer the notation X_{dij}^Y and x_{dij}^Y to indicate that d, i and j are independent variables.

$d_{ij}^M(X, Y)$ **definition.**

1. $d_{ij}^M(\emptyset, Y) = 0$.

If $X \neq \emptyset$:

2. If $0 \leq i \leq 100$ and j is av or $0 \leq j \leq 100$ then $d_{ij}^M(X, Y) = d_j^M(x^Y, Y)$.

3. $d_{avj}^M(X, Y) = \sum_{x \in X} \frac{d_j^M(x, Y)}{|X|}$.

A notation now exists that can represent all six of Dubuisson and Jain’s directed distance functions³ and infinitely many other possibilities.

$$\begin{aligned}
 d_{00}^M(X, Y) &= d_1(X, Y) = \min_{x \in X} \min_{y \in Y} d(x, y), \\
 d_{500}^M(X, Y) &= d_2(X, Y), \\
 d_{750}^M(X, Y) &= d_3(X, Y), \\
 d_{900}^M(X, Y) &= d_4(X, Y), \\
 d_{1000}^M(X, Y) &= d_5(X, Y) = \max_{x \in X} \min_{y \in Y} d(x, y), \\
 d_{av0}^M(X, Y) &= d_6(X, Y), \\
 &\dots \\
 d_{0100}^M(X, Y) &= \min_{x \in X} \max_{y \in Y} d(x, y), \\
 &\dots \\
 d_{100100}^M(X, Y) &= \max_{x \in X} \max_{y \in Y} d(x, y)
 \end{aligned}$$

The undirected version of d_{1000}^M , given by $\max\{d_{1000}^M(X, Y), d_{1000}^M(Y, X)\}$,

³whenever they are defined. The distance functions defined above are also defined for all other $X, Y \subseteq M$.

is the well known *Hausdorff* distance function, which is a metric if d is.

The undirected version of $d_{av\ 0}^M(X, Y)$, given by $\max\{d_{av\ 0}^M(X, Y), d_{av\ 0}^M(Y, X)\}$ is called the modified Hausdorff distance in [32] and [33]⁴. However neither $d_{av\ 0}^M$ nor the modified Hausdorff distance are metrics as ΔI fails. ΔI also fails for all the K^{th} ranked distance functions $d_{i\ 0}^M$ where $i < 100$.

Example 5.2. $d(x, y) = |x - y|$, $X = \{1\}$, $Y = \{2, 3, \dots, 100\}$, $Z = \{1, 2, \dots, 100\}$. Now $d_{i\ 0}^M(X, Y) = 1$, $d_{i\ 0}^M(X, Z) = 0$ and, if $i \leq 99$, $d_{i\ 0}^M(Z, Y) = 0$ and so ΔI fails. By increasing the size of Y and Z similar examples can be contrived where ΔI fails for any $i < 100$.

Some other d_{ij}^M distance functions do satisfy ΔI . This is shown below.

5.4.1 The Triangle Inequality (ΔI)

This section examines ΔI for d_{ij}^M distance functions.

Theorem 5.4 *If $\langle M, d \rangle$ is such that d satisfies ΔI over M then $d_{100\ 0}^M$, $d_{0\ 100}^M$, $d_{100\ 100}^M$, $d_{av\ 100}^M$, $d_{100\ av}^M$ and $d_{av\ av}^M$ satisfy ΔI over $M' \subseteq \mathcal{P}(M)$.*

PROOF. This theorem will be proven for $d_{100\ 0}^M$ and $d_{av\ av}^M$, the other cases are similar.

If $x \in X$, $y \in Y$ and $z \in Z$ where $X, Y, Z \in M'$ then:

$$d(x, y) \leq d(x, z) + d(z, y)$$

Case: $100\ 0$

$$\begin{aligned} \min_{y \in Y} d(x, y) &\leq d(x, z) + \min_{y \in Y} d(z, y), \\ &\leq \min_{z \in Z} d(x, z) + \max_{z \in Z} \min_{y \in Y} d(z, y). \end{aligned}$$

$$\therefore \max_{x \in X} \min_{y \in Y} d(x, y) \leq \max_{x \in X} \min_{z \in Z} d(x, z) + \max_{z \in Z} \min_{y \in Y} d(z, y)$$

$$\text{and } d_{100\ 0}^M(X, Y) \leq d_{100\ 0}^M(X, Z) + d_{100\ 0}^M(Z, Y).$$

⁴Various other “modified Hausdorff” distance functions have been proposed for determining the distance between two digital images. See [58]

Case: $av\ av$

$$\sum_{y \in Y} d(x, y) \leq |Y|d(x, z) + \sum_{y \in Y} d(z, y),$$

$$\text{so } |Z| \sum_{y \in Y} d(x, y) \leq |Y| \sum_{z \in Z} d(x, z) + \sum_{z \in Z} \sum_{y \in Y} d(z, y)$$

$$\text{and } |Z| \sum_{x \in X} \sum_{y \in Y} d(x, y) \leq |Y| \sum_{x \in X} \sum_{z \in Z} d(x, z) + |X| \sum_{z \in Z} \sum_{y \in Y} d(z, y).$$

$$\therefore d_{av\ av}^M(X, Y) \leq d_{av\ av}^M(X, Z) + d_{av\ av}^M(Z, Y).$$

It is easy to check that this holds if any of X, Y, Z are empty.

$\triangle I$ does not hold for d_{ij}^M in general.

Example 5.3. $M = \mathcal{P}(\mathbb{N}_1)$, $d(x, y) = |x - y|$, $X = \{1, 3\}$, $Y = \{5, 9\}$ and $Z = \{3, 5\}$. For each row in table 1 below we can see that $d_{ij}^{\mathbb{N}_1}(X, Y) \not\leq d_{ij}^{\mathbb{N}_1}(X, Z) + d_{ij}^{\mathbb{N}_1}(Z, Y)$ and so $\triangle I$ fails for $d_{0\ 0}^{\mathbb{N}_1}$, $d_{av\ 0}^{\mathbb{N}_1}$ and $d_{0\ av}^{\mathbb{N}_1}$.

	$d_{ij}^{\mathbb{N}_1}(X, Y)$	$d_{ij}^{\mathbb{N}_1}(X, Z)$	$d_{ij}^{\mathbb{N}_1}(Z, X)$
$d_{0\ 0}^{\mathbb{N}_1}$	2	0	0
$d_{av\ 0}^{\mathbb{N}_1}$	3	1	1
$d_{0\ av}^{\mathbb{N}_1}$	4	1	2

Table 5.1: Distances for three distance functions over $X = \{1, 3\}$, $Y = \{5, 9\}$ and $Z = \{3, 5\}$.

5.4.2 Span

The reason why the distance functions $d_{0\ 0}^{\mathbb{N}_1}$ and $d_{av\ 0}^{\mathbb{N}_1}$ fail to satisfy $\triangle I$, in the case illustrated in table 1 above, is that the 3 in $\{3, 5\}$ is close to $\{1, 3\}$ while the 5 is close to $\{5, 9\}$. So here, as for $d_{0\ av}^{\mathbb{N}_1}$, the problem is the “diameter” of $\{3, 5\}$.

The diameter of a set X is given as $\max_{x, y \in X} d(x, y)$ in [17]. A generalised version of $\triangle I$ holds for these examples if the diameter of the intermediate

set Z , $\max_{x,y \in Z} d(x,y) = 2$, is added to the RHS of the inequality. However even that does not work when the elements of X, Y, Z are themselves sets.

Example 5.4.

1. $X, Y, Z \in M$, $X = \{\{1\}, \{2\}\}$, $Y = \{\{3\}, \{4\}\}$, $Z = \{\{2, 5\}, \{5, 4\}\}$,
2. $d(x, y) = |x - y|$, d is a distance function over \mathbb{N}_1 ,
3. $d' = d_{00}^{\mathbb{N}_1}$ is a distance function over $X \cup Y \cup Z$,
4. $d'' = d_{00}^{\mathcal{P}(\mathbb{N}_1)}$ is a distance function over M .

We want $d''(X, Y) \leq d''(X, Z) + \max_{x,y \in Z} d'(x, y) + d''(Z, Y)$. However

$$d''(X, Y) = d''(\{\{1\}, \{2\}\}, \{\{3\}, \{4\}\}) = d'(\{2\}, \{3\}) = d(2, 3) = 1.$$

Whereas

$$d''(X, Z) = d''(\{\{1\}, \{2\}\}, \{\{2, 5\}, \{5, 4\}\}) = d'(\{2\}, \{2, 5\}) = d(2, 2) = 0,$$

$$\max_{x,y \in Z} d'(x, y) = d'(\{2, 5\}, \{5, 4\}) = d(5, 5) = 0,$$

$$d''(Z, Y) = d''(\{\{2, 5\}, \{5, 4\}\}, \{\{3\}, \{4\}\}) = d'(\{5, 4\}, \{4\}) = d(4, 4) = 0.$$

So the inequality does not hold in this case.

What is missing is a consideration of the diameters of $\{2, 5\}$ and $\{5, 4\}$. Taking this into account gives us d_{ij}^M -**span**.

$s_{d_{ij}^M}(X)$ (**the d_{ij}^M -span of X**) **definition.**

If, for some y , $X = \{y\}$ then $s_{d_{ij}^M}(X) = s_d(y)$.

If, for all y , $X \neq \{y\}$ then $s_{d_{ij}^M}(X) = \max_{x,y \in X} \{s_d(x) + d(x, y) + s_d(y)\}$.

Example 5.5. Using $d(x, y) = |x - y|$, $d' = d_{00}^{\mathbb{N}_1}$, $d'' = d_{00}^{\mathcal{P}\mathbb{N}_1}$ as in the

example above, we will calculate $s_{d''}(Z)$ where $Z = \{\{2, 5\}, \{5, 4\}\}$:

$$s_{d''}(Z) = s_{d'}(\{2, 5\}) + d'(\{2, 5\}, \{5, 4\}) + s_{d'}(\{5, 4\}).$$

$$s_{d'}(\{2, 5\}) = s_d(2) + d(2, 5) + s_d(5) = 0 + 3 + 0 = 3,$$

$$d'(\{2, 5\}, \{5, 4\}) = d(5, 5) = 0,$$

$$s_{d'}(\{5, 4\}) = s_d(5) + d(5, 4) + s_d(4) = 0 + 1 + 0 = 1.$$

So

$$s_{d''}(Z) = 3 + 0 + 1 = 4.$$

5.4.3 The Generalised Triangle Inequality ($G\Delta I$)

Lemma 5.5 *If M is a set and $\langle M, d \rangle$, is such that $G\Delta I$ holds and $M' \subseteq \mathcal{P}(M)$ then d_{ij}^M , where $0 \leq i \leq 100$ and $0 \leq j \leq 100$, satisfies $G\Delta I$ over $M' - \{\emptyset\}$.*

PROOF. Let $X, Y, Z \in M$. Consider any pair xY and zY .

If ${}^xY = {}^zY$, $d(z, {}^xy) \leq d(z, {}^zy)$.

If ${}^xY \neq {}^zY$, as $|{}^xY| = |{}^zY|$, there is a $y \in {}^zY - {}^xY$ where $d(x, {}^xy) \leq d(x, y)$.

Of course $d(z, y) \leq d(z, {}^zy)$.

In either case, from the $G\Delta I$,

$$d(x, {}^xy) \leq d(x, z) + s_d(z) + d(z, {}^zy)$$

which is

$$d_j^M(x, Y) \leq d(x, z) + s_d(z) + d_j^M(z, Y). \quad (5.1)$$

Let $z \in Z^Y$ and $z' \in {}^xZ$. Then by $G\Delta I$

$$d_j^M(x, Y) \leq d(x, z') + s_d(z') + d(z', z) + s_d(z) + d_j^M(z, Y)$$

$$\leq d(x, z') + \max_{z, z' \in Z} \{s_d(z') + d(z', z) + s_d(z)\} + d_j^M(z, Y)$$

and using the definition for $s_{d_{ij}^M}(Z)$

$$d_j^M(x, Y) \leq d(x, z') + s_{d_{ij}^M}(Z) + d_j^M(z, Y). \quad (5.2)$$

As $d(x, z') \leq d(x, {}^x z)$ and $d_j^M(z, Y) \leq d_j^M(z^Y, Y)$,

$$d_j^M(x, Y) \leq d(x, {}^x z) + s_{d_{ij}^M}(Z) + d_j^M(z^Y, Y).$$

which is

$$d_j^M(x, Y) \leq d_j^M(x, Z) + s_{d_{ij}^M}(Z) + d_{ij}^M(Z, Y). \quad (5.3)$$

Consider any pair X^Z and X^Y . Substituting any x^Y for x in (5.3) gives:

$$d_j^M(x^Y, Y) \leq d_j^M(x^Y, Z) + s_{d_{ij}^M}(Z) + d_{ij}^M(Z, Y).$$

If $X^Z = X^Y$, $d_j^M(x^Y, Z) \leq d_j^M(x^Z, Z)$ so

$$d_j^M(x^Y, Y) \leq d_j^M(x^Z, Z) + s_{d_{ij}^M}(Z) + d_{ij}^M(Z, Y). \quad (5.4)$$

If $X^Z \neq X^Y$ let $x \in X^Z - X^Y$ so $d_j^M(x^Y, Y) \leq d_j^M(x, Y)$ and, from (5.3),

$$d_j^M(x^Y, Y) \leq d_j^M(x, Z) + s_{d_{ij}^M}(Z) + d_{ij}^M(Z, Y).$$

Of course $d_j^M(x, Z) \leq d_j^M(x^Z, Z)$ so again this is (5.4), which is $G\Delta I$.

Lemma 5.6 *If $\langle M, d \rangle$ is such that $G\Delta I$ holds and $M' \subseteq \mathcal{P}(M)$ then $d_{av\ j}^M$, where $0 \leq j \leq 100$, satisfies $G\Delta I$ over $M' - \{\emptyset\}$.*

PROOF. From (5.2), with ${}^x z$ for z' ,

$$d_j^M(x, Y) \leq d_j^M(x, Z) + s_{d_{ij}^M}(Z) + d_j^M(z, Y).$$

Summing over all $x \in X$ and $z \in Z$ and dividing by $|X||Z|$ gives

$$d_{\text{av}j}^M(X, Y) \leq d_{\text{av}j}^M(X, Z) + s_{d_{ij}^M}(Z) + d_{\text{av}j}^M(Z, Y).$$

Lemma 5.7 *If $\langle M, d \rangle$ is such that $G\Delta I$ holds and $M' \subseteq \mathcal{P}(M)$ then $d_{\text{av av}}^M$ satisfies $G\Delta I$ over $M' - \{\emptyset\}$.*

PROOF. By $G\Delta I$ applied twice

$$d(x, y) \leq d(x, z') + s_d(z') + d(z', z) + s_d(z) + d(z, y).$$

From the definition for $s_{d_{ij}^M}(Z)$

$$d(x, y) \leq d(x, z') + s_{d_{ij}^M}(Z) + d(z, y).$$

Summing over all $y \in Y$ and $z' \in Z$ and dividing by $|Y||Z|$ gives

$$d_{\text{av}}^M(x, Y) \leq d_{\text{av}}^M(x, Z) + s_{d_{ij}^M}(Z) + d_{\text{av}}^M(z, Y). \quad (5.5)$$

Summing over all $x \in X$ and $z \in Z$ and dividing by $|Y||Z|$ gives

$$d_{\text{av av}}^M(X, Y) \leq d_{\text{av av}}^M(X, Z) + s_{d_{ij}^M}(Z) + d_{\text{av av}}^M(Z, Y).$$

Lemma 5.8 *If $\langle M, d \rangle$ is such that $G\Delta I$ holds and $M' \subseteq \mathcal{P}(M)$ then $d_{i \text{av}}^M$, where $0 \leq i \leq 100$, satisfies $G\Delta I$ over $M' - \{\emptyset\}$.*

PROOF.

For any pair X^Y and X^Z .

If $X^Y = X^Z$, $d_{\text{av}}^M(x^Y, Z) \leq d_{\text{av}}^M(x^Z, Z)$.

If $X^Y \neq X^Z$, as $|X^Y| = |X^Z|$, there is an $x \in X^Z - X^Y$. So $d_{\text{av}}^M(x, Z) \leq d_{\text{av}}^M(x^Z, Z)$

and $d_{av}^M(x^Y, Y) \leq d_{av}^M(x, Y)$.

In either case, from (5.5),

$$d_{av}^M(x^Y, Y) \leq d_{av}^M(x^Z, Z) + s_{d_{ij}^M}(Z) + d_{av}^M(z', Y).$$

With z^Y for z' this is

$$d_{iav}^M(X, Y) \leq d_{iav}^M(X, Z) + s_{d_{ij}^M}(Z) + d_{iav}^M(Z, Y).$$

Theorem 5.9 *If M is a set and $\langle M, d \rangle$, is such that $G\Delta I$ holds and $M' \subseteq \mathcal{P}(M)$ then d_{ij}^M , where $0 \leq i \leq 100$ or $i = av$, and $0 \leq j \leq 100$ or $j = av$, satisfies $G\Delta I$ over M' .*

PROOF.

Case $X = \emptyset$. $d_{ij}^M(\emptyset, Y) = 0$ so

$$d_{ij}^M(\emptyset, Y) \leq d_{ij}^M(\emptyset, Z) + s_{d_{ij}^M}(Z) + d_{ij}^M(\emptyset, Z).$$

Case $Y = \emptyset$. From the $G\Delta I$,

$$\max_{y \in M} d(x, y) \leq d(x, z) + s_d(z) + \max_{y' \in M} d(z, y').$$

This is

$$d_j^M(x, \emptyset) \leq d(x, z) + s_d(z) + d_j^M(z, \emptyset)$$

which is (5.1) above with $Y = \emptyset$. The remainder of the proof for this special case follows directly from there.

Case $Z = \emptyset$. We have $d_j^M(x, Y) \leq \max_{y \in M} d(x, y) = d_j^M(x, \emptyset)$ for all $x \in M$. So $d_{ij}^M(X, Y) \leq d_{ij}^M(X, \emptyset)$ and

$$d_{ij}^M(X, Y) \leq d_{ij}^M(X, \emptyset) + s_{d_{ij}^M}(\emptyset) + d_{ij}^M(\emptyset, Y).$$

The remainder of the proof holds due to lemmas 5.5, 5.6, 5.7 and 5.8.

By theorem 5.9 all the d_{ij}^M distance functions satisfy G Δ I.

5.4.4 \subseteq -reflexivity

If $\langle \mathbb{N}_1, d \rangle$ is $\not\subseteq^d$ -strict positive, d_{ij}^M with $j \neq 0$ is *not* \subseteq -reflexive.

Example 5.6. With $j = \text{av}$ or $1 < j \leq 100$, $d_{ij}^M(\{1\}, \{1, 2, \dots, 100\}) > 0$ as $d(1, y) > 0$ for all $y \in \{2, \dots, 100\}$.

Following this example it should be clear that, for any $0 < j \leq 100$, a $Y \subseteq \mathbb{N}_1$ can be found to make $d_{ij}^M(\{1\}, Y) > 0$.

Theorem 5.10 *If $\langle M, d \rangle$ is $(\subseteq -)$ reflexive, d_{i0}^M is \subseteq -reflexive.*

PROOF. For all $X, Y \in \mathcal{P}(M)$ where $X \subseteq Y$, for all $x \in X$, $d_0^M(x, Y) = 0$ as $x \in Y$.

Note that this theorem only requires that $\langle M, d \rangle$ is reflexive—a *weaker* property than \subseteq -reflexivity. Because the theorem holds for reflexive $\langle M, d \rangle$, it automatically holds for \subseteq -reflexive $\langle M, d \rangle$.

5.4.5 $\not\subseteq^d$ -strict positiveness

If $|Z| \geq 7$, $M = \mathcal{P}(Z) - \{\emptyset\}$, d is \subseteq -reflexive over M and $0 \leq i \leq 99$ then d_{ij}^M is *not* $\not\subseteq^{d_{ij}^M}$ -strict positive.

Example 5.7. Let $Y = \{Z\}$, $X = \{X_1, \dots, X_{100}\}$, $X \not\subseteq^{d_{ij}^M} Y$ and, for $1 \leq k \leq 100$, $X_k \neq \emptyset$ and $X_k \subseteq Z$ (X_1, \dots, X_{100} can be distinct only if $|Z| \geq 7$). Now, for $0 \leq i \leq 99$, $d_{ij}^M(X, Y) = 0$.

Following this example it should be clear that, for sufficiently large Z , an X can be chosen to show that, for any $0 \leq i < 100$, d_{ij}^M is *not* $\not\subseteq^{d_{ij}^M}$ -strict positive.

Theorem 5.11 *If d is \mathcal{Z}^d -strict positive, and $d' = d_{100j}^M$ or $d' = d_{avj}^M$ then d' is $\mathcal{Z}^{d'}$ -strict positive.*

PROOF. As d is \mathcal{Z}^d -strict positive, $d_{avj}^M(\emptyset, Y) > 0$ and $d_{100j}^M(\emptyset, Y) > 0$ if $Y \neq \emptyset$. If $X, Y \in \mathcal{P}(M) - \{\emptyset\}$ and $X \mathcal{Z}^{d_{ij}^M} Y$ then there is an $x \in X$ where, for all $y \in Y$, $x \mathcal{Z}^d y$. As d is \mathcal{Z}^d -strict positive, $d_j^M(x, Y) > 0$ so $d_{avj}^M(X, Y) > 0$. Similarly, as $d_{100j}^M(X, Y) \geq d_j^M(x, Y)$ for any $x \in X$, $d_{100j}^M(X, Y) > 0$.

From these examples and theorems, this research concludes that, in general, of the d_{ij}^M distance functions, only d_{av0}^M and d_{1000}^M are suitable distance functions for $\mathcal{Z}^{d_{ij}^M}$ -strict positive and \subseteq -reflexive spaces.

5.5 What this Chapter Achieved

This chapter includes many theorems and proofs that show how spaces, with desirable properties, can be formed and manipulated.

Importantly, a single n -dimensional space, with properties common to all of its dimensions, can be formed from n dimensions, each of which is a set space. There are a number of ways of forming a multidimensional space from k , n -dimensional spaces. For example, one way is to form a kn -dimensional space. At the opposite extreme, a k -dimensional space, where each dimension its self has n dimensions, can be formed. These results will be useful when forming multidimensional information spaces.

Example 5.8. Research papers can be “attached” to points in the space $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$ where: M_1 is a set of “authors” and d_1 an appropriate distance function between authors; M_2 is a set of topic areas and d_2 a distance function between topics; M_3 is a set of paper titles and d_3 a distance function between titles. If the spaces $\langle M_1, d_1 \rangle$, $\langle M_2, d_2 \rangle$, $\langle M_3, d_3 \rangle$ all have the desirable properties, so does $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$. If $\langle M_1, d_1 \rangle$ is an m -dimensional space $\langle N_1 \times \dots \times N_m, G_2^{d_1^* \dots d_m^*} \rangle$ then $\langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$ is equivalent to $\langle N_1 \times \dots \times N_m \times M_2 \times M_3, G_2^{d_1^* \dots d_m^* d_2 d_3} \rangle$.

This chapter also showed how the distance function of individual dimensions of a multidimensional space can be weighted without altering properties of the space. This result can be applied to allow users of multidimensional information spaces to adjust the relative significance of individual dimensions when forming range queries.

Example 5.9. Following the previous example above, it may be that a user is interested in a particular paper. They know the topic they want, but are less certain about the author and title. The space can be tuned to increase the significance of the topic by weighting the topic distance function (with a weight > 1). If the “query point” was $\mathbf{x} = (\text{Smith}, \text{Metric Space}, \text{Searching Metric Spaces})$, the user is interested in papers attached to points \mathbf{y} where $G_2^{d_1 2d_2 d_3}(\mathbf{x}, \mathbf{y})$ is small. Note the $2d_2$, increasing the significance of the topic dimension in this query.

The set distance function $d(X, Y) = |X| - |X \cap Y|$ was shown to have the desirable properties. Similarly, $d_{av\ 0}^M$ and $d_{100\ 0}^M$ have the desirable properties if d does. These results will be useful when forming the dimensions of multidimensional information spaces.

Example 5.10. The utility of the space used in the two previous examples above is limited as only papers with a single author can be correctly “classified”. The space could be improved by replacing the author dimension $\langle M_1, d_1 \rangle$ with the set space $\langle \mathcal{P}(M_1), d \rangle$, where d is a set distance function such as $d(X, Y) = |X| - |X \cap Y|$. This would allow papers with multiple authors to be correctly classified.

Chapter 6

L -Collections

6.1 Overview

This chapter begins with a review of sets and set like generalisations in the literature. **Multisets** allow each element of a set to be associated with a natural number (a positive integer). This is useful, but does not provide everything needed to define information space. **Indexed families** provide a notation that allows the elements of a set to be enumerated. **Rough sets** could have some use representing uncertainty when “probabilities” are not known. **Fuzzy sets** allow fractional membership, but not multiple membership. **L -fuzzy sets** are more general, but a merge form of union—which information space requires—is not defined.

None of these mathematical objects provides all that is required for the definition of information space.

Section 6.3 defines a new form of generalised set, the **L -collection**. The standard set operators are extended over L -collections where $L \subseteq \mathbb{R}^{>0}$ is a **maximizable set**. L -collections provide a consistent template for generalizing sets.

Section 6.4 shows that $\{1\}$ -collections are equivalent to sets, \mathbb{N}_1 -collections are equivalent to multisets, $(0, 1]$ -collections are equivalent to fuzzy sets and $\{0.5, 1\}$ -collections are equivalent to rough sets.

Section 6.5 discusses how proofs over sets and multisets may be extended

to $\{1\}$, \mathbb{N}_1 and $\mathbb{Q}^{>0}$ -collections.

Section 6.6 briefly summarises what has been achieved in this chapter.

6.2 Background

Although distance functions between sets of points, such as d_{ij}^M come close, they fall short of providing all that is required to define information space. For example, we want to compare two sets of information elements where a number of the information elements are “attached” to the same point in the information space. A set of points could have one information element attached to each point, or it could have 1000 information elements attached to one point and 1 information element attached to all others. Because each element of a set must be distinct, each point can only be a member of the set once. So, if an information space is a set space, the multiplicity of information elements attached to each point does not effect the distance between sets of points. For many applications this does not matter, but for some this is a significant failing.

Another problem occurs when we are not sure to what point an information element should be attached. Perhaps we are 40% sure it should be attached to one point and 60% sure it should be attached to another; perhaps we want to assign the authorship of a paper 70% to one author and 30% to another. There is no effective method to do this using set space.

A generalised form of set is required that allows multiple, fractional and even multiple fractional membership.

6.2.1 Sets

Georg Cantor(1845-1918), through papers published between 1874 and 1884, is recognized as the originator of set theory and even modern mathematics.

By a set we understand any M of definite, distinct objects m of our perception or of our thought (which will be called the elements of M) into a whole [translated from German][20].

Cantor's work was highly controversial in his day as constructivist mathematicians were uncomfortable with the notion of infinite sets. In 1901, Bertrand Russell showed that the set containing exactly the sets that are not members of themselves, formally $B = \{A \mid A \notin A\}$, is not well defined as, logically, $B \in B \iff B \notin B$ (Russell's Paradox or Russell's Antinomy). This and other paradoxes lead to the development of axiomatic set theories—such as Zermelo set theory (1908, Ernst Zermelo) and Fraenkel set theory (1922, Abraham Fraenkel and Thoralf Skolem). Currently the Zermelo-Fraenkel set theory along with the axiom of choice—abbreviated ZFC—is the most common choice of axiomatic set theory. Sets are a common choice for the *foundation of mathematics* whereby all mathematical objects are defined in terms of sets and set membership.

6.2.2 Multisets

The requirement that each element of a set be distinct from all other elements (that is—be unique in the set) make sets unsuitable as the mathematical basis for many practical applications. Most usually, multisets are described as sets without the requirement that all elements are distinct. A more formal definition is provided below.

Multiset definition. *A multiset \mathcal{M} is a pair (M, m) where M is a set and $m : M \rightarrow \mathbb{N}_1$. M is called the **underlying set**. For each $x \in M$ the **multiplicity** (that is, the number of occurrences) of x is the number $m(x)$. $x \in \mathcal{M}$ stands for $x \in M$.*

So, by pairing a set M with a function $m : x \rightarrow \mathbb{N}_1$, multisets generalize sets by associating elements of M with a “multiplicity” $m(x)$. Note that, as $m(x) \in \mathbb{N}_1$, $m(x) \neq 0$ for all $x \in M$.

Example 6.1. Given a set M of fruit {apple, orange, banana,...} and a function m where $m(\text{apple}) = 10$, $m(\text{orange}) = 8$, $m(\text{banana}) = 15$, ... we have a multiset (M, m) that might indicate how much of each type of fruit is presently available.

Note a set can be thought of as a type of multiset where the multiplicity of each element is 1.

Multisets are a useful generalisation, but they do not allow fractional membership.

6.2.3 Merges and Joins

When sets are generalised so that each element x has an associated multiplicity or “membership” $m(x)$ (not necessarily in \mathbb{N}_1), union can be generalized in two ways to produce two different operators which are herein called **merge** and **join**.

The merge operator, is often represented by the symbol \uplus . When two “generalized sets” are merged, the multiplicity of each element common to both generalized sets is a function of the sum of the multiplicities of the element in each operand generalized set. More precisely, if $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are generalized sets, where $m_X(x)$ and $m_Y(y)$ are the multiplicities of each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively, and $(M, m) = \mathcal{X} \uplus \mathcal{Y}$ then $M = X \cup Y$ and, for each $x \in X \cap Y$, $m(x)$ is a function of $m_X(x) + m_Y(x)$.

When two (generalized) sets are joined, the multiplicity of each element common to both (generalized) sets is usually the *maximum* of the multiplicities of the element in each operand (generalized) set. If the operands are sets, join is equivalent to union. Because of this, join operators are often represented by the symbol \cup . More precisely, if $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are generalized sets, where $m_X(x)$ and $m_Y(y)$ are the multiplicities of each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively, and $(M, m) = \mathcal{X} \cup \mathcal{Y}$ (where \cup is a join operator) then $M = X \cup Y$ and, for each $x \in X \cap Y$, $m(x)$ is usually $\max\{m_X(x), m_Y(x)\}$.

The term “join” has been used with different meanings in a number of closely related fields.

If (M, \leq) is a partially ordered set, the join of two elements $x, y \in M$ is the least upper bound (the supremum) of x and y and is denoted $x \vee y$. Formally, $z = x \vee y$ iff:

1. $x \leq z$ and $y \leq z$,

2. for any $z' \in Z$, if $x \leq z'$ and $y \leq z'$, $z \leq z'$.

As will be seen in section 6.2.6, the join of two elements in a partially ordered set can be used to define a join type union (as discussed above).

In relational databases “join” refers to the union of two databases. This is defined in relational algebra, where the join of two relations R_1 and R_2 is the *composition* $R_2 \circ R_1$ of R_1 and R_2 [24].

Example 6.2. If $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ then

$$R_2 \circ R_1 = \{(x, z) | (x, y) \in R_1, (y, z) \in R_2\}.$$

6.2.4 Indexed Families

Indexed families are quite common in the literature, although definitions are much harder to find.

Indexed Families are closely related to sets, multisets and tuples.

Indexed Family definition. Depending on the notation used, an **indexed family** can be a multiset or a tuple. In either case, given an (implicit) surjective function $f : Z \rightarrow M$, Z is the **index set** and M is **indexed by** Z . For any $z \in Z$, $f(z)$ is denoted x_z . We say x_z **belongs** to the **key** z .

The indexed family $\{x_z\}_{z \in Z}$ (or simply $\{x_z\}$) is the multiset (M, m) where $m(x)$ equals the number of times, for $z \in Z$, that $f(z) = x$.

Given some (implicit) ordering relation \leq on $Z = \{z_1, \dots, z_{|Z|}\}$ so that $z_i \leq z_j$ for all $1 \leq i \leq j \leq |Z|$, the indexed family $(x_z)_{z \in Z}$ (or (x_z)) is the tuple $(x_{z_1}, \dots, x_{z_{|Z|}})$.

Note that $\{x_z\}_{z \in Z}$ is a set if f is injective. A function $f : Z \rightarrow M$ is *injective* iff, for each $z_1, z_2 \in Z$ where $z_1 \neq z_2$, $f(z_1) \neq f(z_2)$. If f is *not* injective, a single element of M may *belong* to different keys. f is *surjective* iff, for every $z \in M$, there is a $z_i \in Z$ where $f(z_i) = z$.

Example 6.3. If $Z = \{1, \dots, n\}$ and $M = \{x_1, \dots, x_n\}$ we can also write $M = \{x_z\}_{z \in Z}$. M is *indexed* by $\{1, \dots, n\}$ which is the *index set* of M . For any $z \in Z$, x_z *belongs* to the *key* z .

Example 6.4. If $Z = \{1, \dots, n\}$ and $M = \{x_1, \dots, x_n\}$, $(x_z)_{z \in Z} = (x_1, \dots, x_n)$.

Indexed families provide a way of succinctly defining a set (multiset or tuple) in terms of a previously given set.

6.2.5 Rough Sets

The term “rough set” was coined by Zdzislaw Pawlak in 1982 [71, 72]. While it appears that Pawlak does not give an explicit definition of rough sets—Pawlak does not make a clear distinction between rough sets and their use to represent, and derive rules from, “information systems” (relations)—the following definition is consistent with his usage.

Rough Set definition. *A rough set is a pair of sets (A, B) . A is the lower approximation and B is the upper approximation of some set M . $A \subseteq M \subseteq B$.*

As will be seen, rough sets can be used to approximate “imaginary” sets that can not be explicitly defined.

Rough sets are becoming a popular alternative to fuzzy sets when “probabilities” are not readily obtainable or simply not of interest. Perhaps Pawlak’s idiosyncratic use of mathematics is a barrier to the greater acceptance and use of rough sets.

Example 6.5. Given a “database” that associates patients (sets of symptoms) with diagnoses: We have *definite* diagnoses (*all* patients in the database with the set of symptoms X have a particular disease) and we have *possible* diagnoses (*some* patients in the database with the set of symptoms Y have the disease.)

If A is the set of sets of symptoms *definitely* linked with the disease, that is

$$A = \{X | \text{all patients with symptoms } X \text{ have the disease}\},$$

and B is the set of sets of symptoms *possibly* linked with the disease, that is

$$B = \{Y | \text{some patients with symptoms } Y \text{ have the disease}\},$$

then (A, B) is a rough set of sets of symptoms of the disease.

In this example M —the “set” approximated by A and B —is the set of sets of symptoms *actually caused by* the disease. That is

$$M = \{V | V \text{ is a set of symptoms actually caused by the disease}\}.$$

Practically, in some patients the set of symptoms V *is* caused by the disease, while other patients might have the same set of symptoms, some or all of which are caused by other factors. If this is possible then V should be (simultaneously!) an element of and not an element of M . In this case M is an “imaginary” set that can never be explicitly defined—yet (A, B) can still be used to approximate M .

Rough sets could be useful for certain knowledge libraries, such as a patient knowledge library as in the example.

6.2.6 Fuzzy sets

Fuzzy sets were introduced in 1965 by Lotfi Zadeh in [104] as a way of handling uncertainty. Fuzzy sets, just like (infinite) sets, were initially highly controversial. However the vast number practical applications of fuzzy sets and fuzzy-logic have made Zadeh a very highly cited researcher [30, 52].

Fuzzy set definition. *A fuzzy set \mathcal{M} is a set M with a function $m : M \rightarrow [0, 1]$. For each $x \in M$, $m(x)$ is the membership **grade** of x . If $M_+ = \{x \in M | m(x) > 0\} = \{x_1, \dots, x_n\}$, \mathcal{M} can be denoted by $\{m_1(x_1)/x_1, \dots, m_1(x_n)/x_n\}$.*

The set M is the **universe set**—the set of all objects currently of interest. Importantly, the universe set is distinct from the *universal set* (the set of all

objects, including itself) which, as Cantor's paradox shows, does not exist. For the fuzzy set operators \subseteq , \cup , \cap and $-$, it is implicit that the operands (fuzzy sets) all have the same universe set.

Note that the fuzzy sets $\mathcal{M}_1 = (M_1, m_1)$ and $\mathcal{M}_2 = (M_2, m_2)$, where:

1. for all $x \in M_1 \cap M_2$, $m_1(x) = m_2(x)$;
2. for all $x \in M_1 - M_2$, $m_1(x) = 0$; and
3. for all $x \in M_2 - M_1$, $m_2(x) = 0$;

can both be denoted by $\{m(x_1)/x_1, \dots, m(x_n)/x_n\}$ where $M_1 \cap M_2 = \{x_1, \dots, x_n\}$. This is because elements of a universe set with a membership grade of zero, are not considered to be “in” the fuzzy set. So, practically, no distinction is made between \mathcal{M}_1 and \mathcal{M}_2 .

Zadeh defines fuzzy set union in the following way.

Fuzzy set Union definition. *If \mathcal{X} and \mathcal{Y} are fuzzy sets with membership grade functions m_X and m_Y respectively, $\mathcal{X} \cup \mathcal{Y}$ is the fuzzy set \mathcal{M} , with membership grade function m where, for all $x \in M$, $m(x) = \max\{m_X(x), m_Y(x)\}$.*

So fuzzy set union is the *join* form of union.

Zadeh has chosen a notation that, while making it easy to define fuzzy set operators, makes it difficult to describe certain generic fuzzy sets. Rather than a single universe set and membership grades that may be equal to zero, a number of underlying sets M_+ and only non zero membership grades, is preferred in this research. This makes the notation for fuzzy sets used in this research, (M_+, m_+) where $m_+(x) = m(x)$ for all $x \in M_+$, consistent with the notation used for multisets.

Zadeh does not define an “element of” unary relation for fuzzy sets. In this research it is useful, and consistent with how fuzzy sets are used, to define $x \in \mathcal{M}$ by $x \in M_+$.

For Zadeh, the *merge* union of fuzzy sets \mathcal{X} and \mathcal{Y} is the “algebraic sum” $\mathcal{X} + \mathcal{Y}$.

Fuzzy Set Algebraic Sum definition. *If \mathcal{X} and \mathcal{Y} are fuzzy sets with membership grade functions m_X and m_Y respectively, $\mathcal{X} + \mathcal{Y}$ is the fuzzy set \mathcal{M} , with membership grade function m where, for all $x \in M$, $m(x) = m_X(x) + m_Y(x)$.*

Of course this is only meaningful if $m_X(x) + m_Y(x) \leq 1$. Because of this limitation, fuzzy sets cannot be used to define information spaces that allow more than one information element to be attached to the same point.

***L*-fuzzy sets**

Although Zadeh noted the possibility of “fuzzy sets” with more general membership functions, he left it to Joseph Goguen to investigate the idea further in [42], where *L*-fuzzy sets—also called *L*-sets—are defined.

***L*-fuzzy set definition.** *An *L*-fuzzy set \mathcal{M} on a set M is a function $\mathcal{M} : M \rightarrow L$.*

This is a simpler notation than that used for fuzzy sets—achieved without the loss of any precision.

***L*-fuzzy set Union definition.** *If \mathcal{X}, \mathcal{Y} are *L*-fuzzy sets (with the same domain M), and $\mathcal{X}(x) \vee \mathcal{Y}(x)$ is defined for all $x \in \mathcal{X}$ and all $x \in \mathcal{Y}$ then $\mathcal{X} \cup \mathcal{Y}$ is the *L*-fuzzy set \mathcal{M} where $\mathcal{M}(x) = \mathcal{X}(x) \vee \mathcal{Y}(x)$ for all $x \in M$.*

Here $\mathcal{X}(x) \vee \mathcal{Y}(x)$ is the *supremum* of $\mathcal{X}(x)$ and $\mathcal{Y}(x)$ in L (see section 6.2.3).

Example 6.6. Assuming \leq is defined as it normally is for numbers, as the supremum of any two numbers a, b is $\max\{a, b\}$, the union $\mathcal{M} = \mathcal{X} \cup \mathcal{Y}$ of $[0, 1]$ -fuzzy sets \mathcal{X} and \mathcal{Y} is, for all $x \in X$, $\mathcal{M}(x) = \max\{\mathcal{X}(x), \mathcal{Y}(x)\}$. This corresponds to fuzzy set union.

So *L*-fuzzy set union is necessarily the *join* type of union. Merge union is not defined.

Note that, if (L, \leq) is a lattice and $x \vee y \in L$ and $x \wedge y \in L$ for all $x, y \in L$

then union (and its dual, intersection) is defined between all such L -fuzzy sets (with the same domain).

6.3 L -Collections

This section extends L -fuzzy sets by adding a merge type union operator. Because this research is directed towards developing the theory to support a specific application, generality is not particularly important. As it is useful to use a different notation, to avoid confusion, the generalised sets developed in this chapter will be called L -collections. Note however, that the results in [42] also hold for L -collections.

The name “ L -collection” is taken from Cantor’s original definition of sets—where sets are defined as a type of “collection”. Informally, L -collections are a type of collection where each element of a set is associated with an element of the set L .

L -collections are more general than sets, multisets, fuzzy sets and rough sets in the sense that any set has a corresponding $\{1\}$ -collection, any multiset has a corresponding \mathbb{N}_1 -collection, any fuzzy set has a corresponding $(0, 1]$ -collection and any rough set has a corresponding $\{0.5, 1\}$ -collection.

L -Collection definition. *An L -collection \mathcal{M} is a pair (M, m) where M is a set and $m : M \rightarrow L$. m is called the **membership function** and M is called the **underlying set**. For each $x \in M$ the **membership grade** of x is $m(x)$. If $M = \{x_1, \dots, x_n\}$ an alternative notation for \mathcal{M} is $\{x_1|m(x_1), \dots, x_n|m(x_n)\}$.*

Note that, if the multiplicity $m(x)$ of an element x is 1, the multiplicity may be omitted. Also standard set notation may be used to denote a $\{1\}$ -collection¹ so, for example, $\{x_1|1, x_2|1, x_3|1\} = \{x_1, x_2, x_3\} = \{x_1, x_2, x_3\}$.

L -Collection Membership definition. *If $\mathcal{M} = (M, m)$ is an L -collection, $x \in \mathcal{M}$ stands for $x \in M$. If $x \in \mathcal{M}$ we say x is an **element** (or a **member**) of \mathcal{M} .*

¹As they are isomorphic, see section 6.4.

Empty L -collection definition. L -collections with the underlying set \emptyset are denoted by \emptyset_L .

Note that the membership function of \emptyset_L is irrelevant. In this research, sometimes \emptyset is used as shorthand for \emptyset_L when this does not introduce ambiguity.

6.3.1 L -collection operators

For the purposes of this research, L is a set of numbers such as $\mathbb{R}^{>0}$, $(0, 1]$, $\mathbb{Q}^{>0}$, \mathbb{N}_1 and $\{1\}$ etc.

Maximizable set definition. L is a **maximizable set** if, for all $x, y \in \mathbb{R}^{>0}$, $\max\{z \in L \mid z \leq x + y\}$ is defined and (for $y < x$), $\max\{z \in L \mid z \leq x - y\}$ is defined.

Note $\mathbb{R}^{>0}$, $(0, 1]$, $\mathbb{Q}^{>0}$, \mathbb{N}_1 and $\{1\}$ are all maximizable sets.

In this section intersection, merge type union and join type union operators are defined where both operands are L -collections and $L \subseteq \mathbb{R}^{>0}$ is a maximizable set. If the intersection, merge or join of an L_1 -collection and an L_2 -collection is required, where $L_1 \neq L_2$, one or other collection must first be *cast* so both are of the same “type”.

Many of the definitions below can be generalised to the case where (L, \leq) is a lattice.

L -Collection Intersection definition. If $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are L -collections, $\mathcal{X} \cap \mathcal{Y}$ is the L -collection $(X \cap Y, m)$ where, for all $x \in X \cap Y$, $m(x) = \min\{m_X(x), m_Y(x)\}$.

L -Collection Union (Merge) definition. If $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are L -collections, $\mathcal{X} \cup \mathcal{Y}$ is the L -collection $(X \cup Y, m)$ where, if:
 $x \in X \cap Y$, $m(x) = \max\{y \in L \mid y \leq m_X(x) + m_Y(x)\}$.
 $x \in X - Y$, $m(x) = m_X(x)$.
 $x \in Y - X$, $m(x) = m_Y(x)$.

Example 6.7. If $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are $(0, 1]$ -collections then $\mathcal{X} \cup \mathcal{Y} = (X \cup Y, m)$ will also be a $(0, 1]$ -collection. If there is a $x \in X \cap Y$ where $m_X(x) = 1$ and $m_Y(x) = 1$ then $m(x) = 1$ as $\max\{y \in (0, 1] \mid y \leq 2\} = 1$.

Note that, if $y_1, y_2 \in L \Rightarrow y_1 + y_2 \in L$ and $x \in X \cap Y$, $m(x) = m_X(x) + m_Y(x)$. The condition $y_1, y_2 \in L \Rightarrow y_1 + y_2 \in L$ is always satisfied if L is \mathbb{N}_1 , $\mathbb{Q}^{>0}$ or $\mathbb{R}^{>0}$, but is not if L is $\{1\}$ or $(0, 1]$.

If \mathcal{X} and \mathcal{Y} are $\{1\}$ -collections, $\mathcal{X} \cup \mathcal{Y}$ works the same way as $X \cup Y$. In this research, union is used to add elements to L -collections—when theorems over sets are generalised into theorems over L -collections, this form of union is required. For these reasons, \cup rather than \oplus is used to represent the merge form of L -collection union.

L -Collection Union (Join) definition. If $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are L -collections, $\mathcal{X} \vee \mathcal{Y}$ is the L -collection $(X \cup Y, m)$ where, if:

- $x \in X \cap Y$, $m(x) = \max\{m_X(x), m_Y(x)\}$.
- $x \in X - Y$, $m(x) = m_X(x)$.
- $x \in Y - X$, $m(x) = m_Y(x)$.

This definition for the join form of union is included for completeness only. If the operands are $\{1\}$ -collections, this form of union produces the same result as set union.

***L*-Collection Complement definition.** Given the *L*-collections

$\mathcal{X} = (X, m_X)$, $\mathcal{Y} = (Y, m_Y)$ and $\mathcal{Z} = (Z, m_Z)$.

$\mathcal{X} - \mathcal{Y}$: $\mathcal{X} - \mathcal{Y} = \mathcal{Z}$ where $Z = \{x \in X \mid \exists z \in L \text{ where } m_X(x) - m_Y(x) \geq z\}$ and, for each $x \in Z$, $m_Z(x) = \max\{y \in L \mid y \leq m_X(x) - m_Y(x)\}$.

$\mathcal{X} - \{x\}$: If $x \in X$, $\mathcal{X} - \{x\} = (X - \{x\}, m_X)$.

$\mathcal{X} - (x, r)$: If $x \in X$, $r \in L$ and

1. $\exists z \in L$ where $m_X(x) - r \geq z$ then $\mathcal{X} - (x, r) = \mathcal{Y}$ where $Y = X$, $m_Y(x) = \max\{y \in L \mid y \leq m_X(x) - r\}$ and, for all $z \in X - \{x\}$, $m_Y(z) = m_X(z)$.
2. $\forall z \in L$, $m_X(x) - r < z$ then $\mathcal{X} - (x, r) = \mathcal{X} - \{x\}$.

Sub *L*-Collection definition. If $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are *L*-collections, $\mathcal{X} \subseteq \mathcal{Y}$ iff $X \subseteq Y$ and, for each $x \in X$, $m_X(x) \leq m_Y(x)$.

***L*-Collection Power Set definition.** If \mathcal{M} is an *L*-collection, $\mathcal{P}(\mathcal{M}) = \{\mathcal{X} \mid \mathcal{X} \subseteq \mathcal{M}\}$.

Note that $\mathcal{P}(\mathcal{M})$ is a set—not an *L*-collection—of *L*-collections. Power sets of $\mathbb{Q}^{>0}$, $\mathbb{R}^{>0}$ or $(0, 1]$ -collections will be infinite.

Example 6.8. If $\mathcal{M} = \{^c a \mid 1\}$ is a $(0, 1]$ -collection and

$$\mathcal{P}(\mathcal{M}) = \{\{^c a \mid 1\}, \{^c a \mid x_1\}, \{^c a \mid x_2\}, \dots, \emptyset\},$$

(where $1 > x_1 > x_2 > \dots$) is infinite.

***L*-Collection Cardinality definition.** If $\mathcal{M} = (M, m)$ is an *L*-collection, for finite M , $|\mathcal{M}| = \sum_{x \in M} m(x)$.

There are two “cardinalities” associated with an L -collection $\mathcal{M} = (M, m)$. We call $|\mathcal{M}|$ the cardinality of \mathcal{M} whereas $|M|$ is the number of distinct elements in \mathcal{M} .

Singleton definition. *If $\mathcal{M} = (M, m)$ is an L -collection where $|M| = 1$ (there is only 1 distinct element in \mathcal{M}) then \mathcal{M} is a **singleton**.*

Note that $\mathcal{M} = (M, m)$ is a singleton whenever M is a singleton set. So $(\{x\}, m)$ is a singleton, even though it may be that $m(x) = 1000$.

L -Collection Scaling definition. *If $\mathcal{X} = (X, m_X)$ is an L -collection and $w \in \mathbb{R}^{\geq 0}$, $w\mathcal{X}$ is an L -collection (M, m) where $M = \{x \in X \mid \exists u \in L, u \leq wm_X(x)\}$ and, for all $x \in M$, $m(x) = \max\{y \in L \mid y \leq wm_X(x)\}$. $w\mathcal{M}$ is the L -collection \mathcal{M} scaled by w .*

Scaling a $\{1\}$, \mathbb{N}_1 , $\mathbb{Q}^{>0}$, $\mathbb{R}^{>0}$ or $(0, 1]$ -collection (M, m_X) by 0 will result in the L -collection \emptyset_L as $0 \notin L$. Indeed, scaling a $\{1\}$ -collection by any $w < 1$ will result in $\emptyset_{\{1\}}$. For certain L -collections \mathcal{M} , $\mathcal{M} \neq \frac{1}{w}(w\mathcal{M})$.

Example 6.9. Let $\mathcal{M} = \{x|3\}$ be an \mathbb{N}_1 -collection.

Because $\max\{y \in \mathbb{N}_1 \mid y \leq 1.5\} = 1$, $0.5\mathcal{M} = \{x|1\}$.

Because $\max\{y \in \mathbb{N}_1 \mid y \leq \frac{1}{0.5}\} = 2$, $\frac{1}{0.5}(0.5\mathcal{M}) = \{x|2\}$.

Note that, if \mathcal{M} is an L -collection and $wy \in L$ for all $y \in L$ then $\mathcal{M} = \frac{1}{w}(w\mathcal{M})$.

L -Collection Casting definition. *If $\mathcal{X} = (X, m_X)$ is an L_1 -collection, $\mathcal{X}_{L_2} = (Y, m)$ is an L_2 -collection where $Y = \{x \in X \mid m_X(x) \geq y \text{ for some } y \in L\}$ and for all $x \in Y$, $m(x) = \max\{r \in L_2 \mid r \leq m_X(x)\}$.*

This definition allows us to resolve union, complement and subset operations between an L_1 -collection and an L_2 -collection (or between an L_2 -collection and an L_1 -collection) where $L_1 \subseteq L_2$ by casting the L_1 -collection to an L_2 -collection.

This method only works if $L_1 \subseteq L_2$. In the more general case where $L_1 \neq L_2$, casting operands to $L_1 \cup L_2$ -collections can give unexpected results.

Example 6.10. Let $\mathcal{X} = \{x|0.9\}$ be a $(0, 1]$ -collection and $\mathcal{Y} = \{x|2\}$ an \mathbb{N}_1 -collection. Now, because $\max\{x \in (0, 1] \cup \mathbb{N}_1 | x \leq 0.9 + 2\} = 2$,

$$\mathcal{X}_{(0,1] \cup \mathbb{N}_1} \cup \mathcal{Y}_{(0,1] \cup \mathbb{N}_1} = \{x|2\},$$

as $2.9 \notin (0, 1] \cup \mathbb{N}_1$, rather than $\{x|2.9\}$ as we might have expected.

It can be useful to cast an L -collection before scaling it. It has already been discussed that, if \mathcal{X} is a $\{1\}$ -collection, $0.5\mathcal{X}$ is $\emptyset_{\{1\}}$. In contrast, $0.5(\mathcal{X}_{(0,1]})$ is a, potentially useful, $(0, 1]$ -collection.

6.4 Sets, Multisets, Fuzzy Sets, Rough Sets and L -Collections

6.4.1 Sets and L -Collections

For any $\{1\}$ -collection $\mathcal{X} = (X, 1)$:

1. as $x \in \mathcal{X} \iff x \in X$, X is isomorphic to \mathcal{X} ;
2. $\mathcal{P}(\mathcal{X})$ is the set of all $\{1\}$ -collections $(Z, 1)$ where $Z \subseteq X$, so $\mathcal{P}(\mathcal{X})$ is isomorphic to $\mathcal{P}(X)$;
3. $|\mathcal{X}| = \sum_{x \in X} 1 = |X|$;

Furthermore, if $\mathcal{Y} = (Y, 1)$ is a $\{1\}$ -collection:

1. $\mathcal{X} \cap \mathcal{Y} = (X \cap Y, 1)$, so $\mathcal{X} \cap \mathcal{Y}$ is isomorphic to $X \cap Y$.
2. $\mathcal{X} \cup \mathcal{Y} = (X \cup Y, 1)$, so $\mathcal{X} \cup \mathcal{Y}$ is isomorphic to $X \cup Y$.
3. $\mathcal{X} - \mathcal{Y} = (X - Y, 1)$, so $\mathcal{X} - \mathcal{Y}$ is isomorphic to $X - Y$.
4. As the membership function is 1 for all $x \in X$, $\mathcal{X} \subseteq \mathcal{Y} \iff X \subseteq Y$.

Scale is not defined for sets so cX has no meaning. So, in terms of the operations defined in this chapter, the $\{1\}$ -collection $\mathcal{X} = (X, 1)$ is isomorphic to the set X —with additional scaling and casting operators. Because of this, \emptyset is used, rather than $\emptyset_{\{1\}}$ to represent the empty $\{1\}$ -collection.

6.4.2 Multisets and *L*-Collections

Comparing the definition of an *L*-collection with the definition of a multiset, it is easy to see that a multiset (M, m) is isomorphic to an \mathbb{N}_1 -collection (M, m) .

6.4.3 Fuzzy Sets and *L*-Collections

It should be clear that, any fuzzy set $\{m(x_1)/x_1, \dots, m(x_n)/x_n\}$ has an isomorphic $(0, 1]$ -collection $\{x_1|m(x_1), \dots, x_n|m(x_n)\}$.

6.4.4 Rough Sets and *L*-Collections

Every rough set (A, B) can be assigned a corresponding $\{0.5, 1\}$ -collection (B, m) where for each $x \in A$, $m(x) = 1$ and for each $x \in B - A$, $m(x) = 0.5$. For practical purposes, rough sets are always defined with respect to a relation (Pawlak’s “information system”) so union, intersection operators etc. are not important.

6.5 Extending Proofs Over Sets and Multisets to $\{1\}$, \mathbb{N}_1 and $\mathbb{Q}^{>0}$ -Collections

Because, as is discussed in section 6.4.1 sets and 1-collections are isomorphic, proofs that hold over sets should automatically hold over corresponding $\{1\}$ -collections. Sets and corresponding $\{1\}$ -collections can be substituted for one another without altering any result (that does not involve scale or casting). Similarly, multisets—with appropriately defined operators—are effectively just \mathbb{N}_1 -collections.

6.5. EXTENDING PROOFS OVER SETS AND MULTISSETS TO $\{1\}$, \mathbb{N}_1 AND $\mathbb{Q}^{>0}$ -COLLECTIONS

Consider the set $M = \{x_1, \dots, x_n\}$ where, for each $1 \leq i \leq n$, $p(x_i)$ is the set of properties of x_i . Because each element of M must be distinct, it is the case that, for any $1 \leq j \leq n$, $p(x_i) \neq p(x_j)$ if $i \neq j$. But what if the set $X = \{x_{1\ 1}, \dots, x_{1\ n_1}, \dots, x_{n\ 1}, \dots, x_{n\ n_n}\}$ is required where n_1, \dots, n_n are $\in \mathbb{N}_1$ and, for each $1 \leq k \leq n_i$, x_{ik} has the properties $p(x_i)$?

One approach is to use an \mathbb{N}_1 -collection $\mathcal{M} = (M, m)$ where each $m(x_i) = n_i$. Another approach is to use a *labeled* set X where $x_{ik} \in X$ and x_{ik} has the properties $p(x_i) \cup l(k)$ where $l(k)$ is the property of having the label k . As long as the label of an element is not otherwise germane to our investigation, X and \mathcal{M} should serve equally well.

In later chapters, mappings between \mathcal{M} and X will be used to extend theorems, proven over sets, to \mathbb{N}_1 -collections. These proofs have the following general form:

1. A certain result holds over the set X .
2. This result still holds after adding further elements x_{ik} to X , where each additional element has the properties $p(x_i) \cup l(k)$.
3. But there is a property preserving mapping from \mathcal{M} to X , so the result holds over \mathcal{M} .

If $\mathcal{M} = (M, m)$ is a finite $\mathbb{Q}^{>0}$ -collection then there is a $w \in \mathbb{N}_1$ such that $w\mathcal{M}$ is isomorphic to an \mathbb{N}_1 -collection. In other words, there is a $w \in \mathbb{N}_1$ such that $wm(x) \in \mathbb{N}_1$ for all $x \in M$. Because the scale of \mathcal{M} does not effect certain properties (such as distance between, and spans of, elements of \mathcal{M}), this can be used to extend theorems, proven over \mathbb{N}_1 -collections, to $\mathbb{Q}^{>0}$ -collections. The general form for such proofs is as follows:

1. For any finite $\mathbb{Q}^{>0}$ -collection $\mathcal{M} = (M, m)$ where $m : M \rightarrow \mathbb{Q}^{>0}$ there is a w such that $w\mathcal{M}$ is isomorphic to an \mathbb{N}_1 -collection.
2. Due to this isomorphism, a certain result holds over $w\mathcal{M}$.
3. But certain properties of $w\mathcal{M}$ are unaffected by scale, so the result holds over \mathcal{M} .

Importantly, this form of proof is only valid for *finite* $\mathbb{Q}^{>0}$ -collections. As infinite $\mathbb{Q}^{>0}$ -collections are not required to define information space, this limitation is not of concern.

6.6 What this Chapter Achieved

This chapter reviewed the literature on “set like” generalisations. As a suitable generalisation was not found, L -collections were defined. L -collections are a type of collection where each element of a set is associated with an element of the set $L \subseteq \mathbb{R}^{>0}$. L -collections are closely related to L -fuzzy sets, though different notation is used and different set operators are defined.

In the context of this research, L -collections have two main uses. First, L -collections enable the representation of multiple identical information elements. Second, L -collections allow “partial” and/or “uncertain” classifications to be made. Further theoretical development is required before examples of these uses can be given (see chapter 8).

Chapter 7

L -Collection Space

7.1 Overview

This chapter is about L -collection distance functions—distance functions over sets of L -collections.

Section 7.2 defines **L -collection distance function**.

Section 7.3 discusses $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$ —a set distance function introduced in chapter 5. It is shown that $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$ satisfies ΔI , is $\not\subseteq^d$ -strict positive and \subseteq -reflexive when \mathcal{X} and \mathcal{Y} are L -collections.

Section 7.4 generalises the set space distance function d_{ij}^M so that it is defined over sets of L -Collections, rather than just sets of sets. The generalised distance function is called $d_{ij}^{\mathcal{M}}$ —where \mathcal{M} is an L -collection. Unfortunately no $d_{ij}^{\mathcal{M}}$ distance function is $\not\subseteq^{d^{\mathcal{M}}}$ -strict positive. The reason for this is that we could have a pair of L -collections $\mathcal{X} = (X, m_1)$ and $\mathcal{Y} = (Y, m_2)$ where $X \subseteq Y$ but $m_1(x) > m_2(x)$ for some $x \in X$. Because $X \subseteq Y$, $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y}) = 0$. But because $\mathcal{X} \not\subseteq \mathcal{Y}$, this is not what we want.

Section 7.5 solves this problem by defining a new distance function ${}_k^{\mathcal{M}}d$ where $0 \leq k \leq 1$ and \mathcal{M} is an L -collection. Informally, when determining ${}_k^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y})$, k gives a proportion of \mathcal{X} close to \mathcal{Y} , similarly to how, when determining $d_{ij}^M(X, Y)$, i gives a percentage of X close to Y .

Although ${}_k^{\mathcal{M}}d$ where $k < 1$ does *not* satisfy $G\Delta I$, this section shows that if d satisfies $G\Delta I$ over a set M , ${}_1^M d$ satisfies $G\Delta I$ over $\mathcal{P}(M)$. The section then

shows that $\overset{\mathcal{M}}{1}d$ satisfies $G\Delta I$, over $\mathcal{P}(\mathcal{M})$ where \mathcal{M} is a finite L -collection with $L \subseteq \mathbb{Q}^{>0}$. Greater generality is not required and so is not pursued.

The section goes on to show that, if d is (\subseteq) -reflexive and \mathcal{Z}^d -strict positive over M then $\overset{\mathcal{M}}{k}d$ is \subseteq -reflexive, and $\overset{\mathcal{M}}{1}d$ is $\mathcal{Z}^{\overset{\mathcal{M}}{1}d}$ -strict positive, over $\mathcal{P}(\mathcal{M})$ where $\mathcal{M} = (M, m)$ is any finite L -collection.

Section 7.6 defines $\overset{\mathcal{M}}{\text{av}}d$. Following a similar pattern of proofs as for $\overset{\mathcal{M}}{k}d$ we show that, if d satisfies $G\Delta I$, is (\subseteq) -reflexive and \mathcal{Z}^d -strict positive over M then $\overset{\mathcal{M}}{\text{av}}d$ satisfies $G\Delta I$, is \subseteq -reflexive and $\mathcal{Z}^{\overset{\mathcal{M}}{\text{av}}d}$ -strict positive over $\mathcal{P}(\mathcal{M})$ where $\mathcal{M} = (M, m)$ is a finite L -collection with $L \subseteq \mathbb{Q}^{>0}$.

Section 7.7 briefly summarises what has been achieved in this chapter.

7.2 L -Collection Distance Functions

L -collection distance function definition. *If d is a distance function over a set M of L -collections then d is an L -collection distance function.*

Example 7.1. Following the definition of L -collection distance function, if M is a set of \mathbb{N}_1 -collections, d is an \mathbb{N}_1 -collection distance function.

Note that, as previously defined, if M is a set of sets (effectively $\{1\}$ -collections) d is a **set distance function**.

Section 6.3 defined \subseteq for L -collections. Thus the definitions of \subseteq -reflexivity (section 4.9.1) and \mathcal{Z}^d -strict positiveness (section 4.9.3) can be applied to distance functions over sets of L -collections.

An L -Collection Distance Function Based on L -Collection Operations

The straight-forward distance function

$$d(\mathcal{X}, \mathcal{Y}) = |\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$$

introduced in section 5.3 as a set distance function can also be used as a *L*-collection distance function. To verify that $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$ satisfies ΔI , is \subseteq -reflexive and \mathcal{Q}^d -strict positive, when \mathcal{X} and \mathcal{Y} are *L*-collections this result is reiterated.

7.2.1 ΔI for $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$

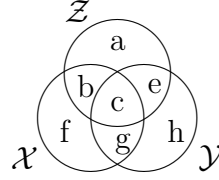


Figure 7.1: Sub *L*-collection element counts for ΔI proofs. $a = |\mathcal{Z} - \mathcal{X} - \mathcal{Y}|$, $b = |\mathcal{Z} \cap \mathcal{X} - \mathcal{Y}|$, $c = |\mathcal{Z} \cap \mathcal{X} \cap \mathcal{Y}|$, etc.

We want to show that

$$|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}| \leq (|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Z}|) + (|\mathcal{Z}| - |\mathcal{Z} \cap \mathcal{Y}|).$$

But this is $b + f \leq (f + g) + (a + b)$, which holds as $0 \leq g + a$.

7.2.2 \mathcal{Q}^d -strict positiveness for $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$

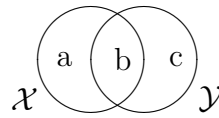


Figure 7.2: Sub *L*-collection element counts for \mathcal{Q} -strict positive proofs. $a = |\mathcal{X} - \mathcal{Y}|$, $b = |\mathcal{X} \cap \mathcal{Y}|$ and $c = |\mathcal{Y} - \mathcal{X}|$.

We want to show that $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}| > 0$ if $\mathcal{X} \not\subseteq \mathcal{Y}$. That is $a - b + b > 0$.

Note that, as d is *not* based on any other function, the \mathcal{Z}^d -strict positive and \mathcal{Z} -strict positive properties are equivalent.

7.2.3 \subseteq -reflexivity for $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$

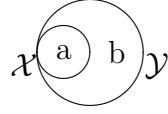


Figure 7.3: Sub L -collection element counts for \subseteq -reflexive proofs. $a = |\mathcal{X}|$, $b = |\mathcal{Y} - \mathcal{X}|$

We want to show that $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}| = 0$ if $\mathcal{X} \subseteq \mathcal{Y}$. That is $a - a = 0$.

7.3 The $d_{ij}^{\mathcal{M}}$ L -Collection Distance Function

This section defines $d_{ij}^{\mathcal{M}}$ which gives distances between L -collections $\subseteq \mathcal{M}$. So $d_{ij}^{\mathcal{M}}$ can be used as a distance function over any subset of $\mathcal{P}(\mathcal{M})$ of L -collections. If \mathcal{M} is a $\{1\}$ -collection (a set) then the definitions of ${}^x\mathcal{Y}$, $d_j^{\mathcal{M}}(x, \mathcal{Y})$, $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y})$ and $\mathcal{X}^{\mathcal{Y}}$, below, are semantically identical to the definitions of xY , $d_j^M(x, Y)$, X^Y and $d_{ij}^M(X, Y)$ (respectively) in section 5.4.

Definition of ${}^x\mathcal{Y}$. ${}^x\mathcal{Y} = (M, m)$ is any L -collection $\subseteq \mathcal{Y} = (Y, m_Y)$ where, for each $y \in {}^x\mathcal{Y}$:

1. $m(y) = m_Y(y)$;
2. $d(x, y) \leq d(x, y')$ for all $y' \in \mathcal{Y} - {}^x\mathcal{Y}$;
3. if $j = 0$, ${}^x\mathcal{Y} = (\{y\}, m_Y(y))$,
if $j \neq 0$, $|{}^x\mathcal{Y}| \geq \frac{j|\mathcal{Y}|}{100}$ and if, for all $y' \in {}^x\mathcal{Y}$, $d(x, y) \geq d(x, y')$ then
 $|{}^x\mathcal{Y} - \{y\}| < \frac{j|\mathcal{Y}|}{100}$.

Just as for xY , an L -collection ${}^x\mathcal{Y}$ can be built-up by starting from \emptyset and adding elements $y \in Y$ —from the smallest $d(x, y)$ to the largest—until $|{}^x\mathcal{Y}| > \frac{j|Y|}{100}$.

Definition of $d_j^{\mathcal{M}}(x, \mathcal{Y})$ where $\mathcal{M} \neq \emptyset$, $\mathcal{Y} = (Y, m_Y) \subseteq \mathcal{M}$ and $d(x, y)$ is defined for all $y \in \mathcal{M}$.

1. $d_j^{\mathcal{M}}(x, \emptyset) = \max_{y \in \mathcal{M}} d(x, y)$.

If $\mathcal{Y} \neq \emptyset$:

2. If $0 \leq j \leq 100$, $d_j^{\mathcal{M}}(x, \mathcal{Y}) = \max_{y \in {}^x\mathcal{Y}} d(x, y)$.

3. $d_{\text{av}}^{\mathcal{M}}(x, \mathcal{Y}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} m_Y(y) d(x, y)$.

Note that $d_j^{\mathcal{M}}(x, \mathcal{Y})$ and $d_{\text{av}}^{\mathcal{M}}(x, \mathcal{Y})$ are independent of the choice of ${}^x\mathcal{Y}$.

Definition of $\mathcal{X}^{\mathcal{Y}}$. $\mathcal{X}^{\mathcal{Y}} = (M, m)$ is any L -collection $\subseteq \mathcal{X} = (X, m_X)$ where, for each $x \in \mathcal{X}^{\mathcal{Y}}$:

1. $m(x) = m_X(x)$;

2. $d_j(x, \mathcal{Y}) \leq d_j(x', \mathcal{Y})$ for all $x' \in \mathcal{X} - \mathcal{X}^{\mathcal{Y}}$;

3. if $i = 0$, $\mathcal{X}^{\mathcal{Y}} = (\{x\}, m_X(x))$,
if $i \neq 0$, $|\mathcal{X}^{\mathcal{Y}}| \geq \frac{i|\mathcal{X}|}{100}$ and if, for all $x' \in \mathcal{X}^{\mathcal{Y}}$, $d_j(x, \mathcal{Y}) \geq d_j(x', \mathcal{Y})$ then
 $|\mathcal{X}^{\mathcal{Y}} - \{x\}| < \frac{i|\mathcal{X}|}{100}$.

Similarly to ${}^x\mathcal{Y}$ (and X^Y) an L -collection $\mathcal{X}^{\mathcal{Y}}$ can be built-up by starting from \emptyset and adding elements $x \in X$ in turn—from the smallest $d_j^{\mathcal{M}}(x, Y)$ to the largest $d_j^{\mathcal{M}}(x, Y)$ —until $|\mathcal{X}^{\mathcal{Y}}| > \frac{i|X|}{100}$.

Definition of $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y})$ where $\mathcal{X} = (X, m_X)$.

1. $d_{ij}^{\mathcal{M}}(\emptyset, \mathcal{Y}) = 0$.

If $\mathcal{X} \neq \emptyset$:

2. If $0 \leq i \leq 100$, $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y}) = \max_{x \in \mathcal{X}^{\mathcal{Y}}} d_j^{\mathcal{M}}(x, \mathcal{Y})$.

3. $d_{\text{avj}}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} m_X(x) d_j^{\mathcal{M}}(x, \mathcal{Y})$.

Note that $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y})$ and $d_{\text{avj}}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y})$ are independent of the choice of $\mathcal{X}^{\mathcal{Y}}$.

The problem with $d_{ij}^{\mathcal{M}}$ is that, if \mathcal{M} can be an L -collection, no $d_{ij}^{\mathcal{M}}$ distance function is $\not\subseteq^{d_{ij}^{\mathcal{M}}}$ -strict positive.

Example 7.2. Let $\mathcal{X} = (\{x\}, m_X)$, $\mathcal{Y} = (\{x\}, m_Y)$, $m_X(x) > m_Y(x)$, $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$. Now $\mathcal{X} \not\subseteq \mathcal{Y}$ but, for any i, j , if $d(x, x) = 0$, $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y}) = 0$.

7.4 The $\frac{\mathcal{M}}{k}d$ L -Collection Distance Function

Informally $\frac{\mathcal{M}}{k}d(X, Y)$ is the k^{th} proportion of X that is nearest to Y . This is similar to the $i^{\text{th}}\%$ of X nearest to a $y \in Y$ in $d_{ij}^{\mathcal{M}}(X, Y)$. If M is a set, to find the distance $\frac{\mathcal{M}}{k}d(X, Y)$ between two sets $X, Y \subseteq M$:

1. Choose $x \in X$ and $y \in Y$ so that $d(x, y)$ is minimised—this is the first distance.
2. Remove x from X and y from Y giving $X^- = X - \{x\}$ and $Y^- = Y - \{y\}$.
3. Now choose $x \in X^-$, $y \in Y^-$ so that $d(x, y)$ is minimised—this is the second distance.
4. Repeat until each $x \in X$ has been chosen (pairing residual elements of X with the empty set, with a distance = $\max_{x, y \in M} d(x, y)$, in the event that Y is exhausted).

5. $\mathcal{M}_k^D d(X, Y)$, where $k \in [0, 1]$, is the $\lceil k|X| \rceil^{th}$ such distance.

Example 7.3. $d(x, y) = |x - y|$, $X = \{2, 4, 5\}$, $Y = \{1, 4, 7\}$. The first distance is $d(4, 4) = 0$, the second is $d(2, 1) = 1$ and the third is $d(5, 7) = 2$. We have $|X| = 3$ so $\lceil 0.5|X| \rceil = 2$ and $\mathcal{M}_{0.5}^D d(X, Y) = d(2, 1) = 1$.

If \mathcal{M} is an L -collection and $\mathcal{X} = (X, m_X)$, $\mathcal{Y} = (Y, m_Y)$ are L -collections, choose $x \in X$ and $y \in Y$ so that $d(x, y)$ is minimised. This distance is given a weight of

$$p = \frac{\min\{m_X(x), m_Y(y)\}}{|\mathcal{X}|}$$

and $(x, p|\mathcal{X}|)$ is removed from \mathcal{X} and $(y, p|\mathcal{Y}|)$ from \mathcal{Y} before repeating, as in the algorithm above.

Definition of $\mathcal{M}_k^D d(\mathcal{X}, \mathcal{Y})$ where $\mathcal{M} = (M, m)$ is an L -collection, $k \in [0, 1]$, d is a distance function over M and $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$ definition.

$$\mathcal{M}_k^D d(\emptyset, \mathcal{Y}) = 0.$$

$$\text{If } \mathcal{X} \neq \emptyset, \mathcal{M}_k^D d(\mathcal{X}, \emptyset) = \max_{x, y \in M} d(x, y).$$

If $\mathcal{X}, \mathcal{Y} \neq \emptyset$: Let $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$. Choose $x \in X$ and $y \in Y$ where $d(x, y) = \min_{x_1 \in X, y_1 \in Y} d(x_1, y_1)$. If there is more than one such pair, choose the pair that minimises $\mathcal{M}_k^D d(\mathcal{X}, \mathcal{Y})$.

$$\text{Let } p = \frac{\min\{m_X(x), m_Y(y)\}}{|\mathcal{X}|}.$$

$$1. \text{ If } k \leq p, \mathcal{M}_k^D d(\mathcal{X}, \mathcal{Y}) = d(x, y).$$

$$2. \text{ If } k > p, \mathcal{M}_k^D d(\mathcal{X}, \mathcal{Y}) = \mathcal{M}_{k^-}^D d(\mathcal{X}^-, \mathcal{Y}^-).$$

Where $\mathcal{X}^- = \mathcal{X} - (x, p|\mathcal{X}|)$, $\mathcal{Y}^- = \mathcal{Y} - (y, p|\mathcal{Y}|)$ and $k^- = \frac{k-p}{1-p}$.

Note that this definition for $\mathcal{M}_k^D d$ is *recursive*, with (1) being a *base case* and (2) being the *general case*. $\mathcal{M}_k^D d$ is a distance function over $\mathcal{P}(\mathcal{M})$.

Because this is a procedural definition, the following, equivalent, (naive recursive) algorithm is provided for greater clarity.

Preconditions. $c = \max_{x, y \in M} d(x, y)$ is a predefined constant, $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ are L -collections and $0 \leq k \leq 1$.

$\mathbf{d}(k, \mathcal{X}, \mathcal{Y})$

if \mathcal{X} is \emptyset , return 0

if \mathcal{Y} is \emptyset , return c

$candidates = \{(x, y) | x \in X, y \in Y, d(x, y) \leq d(x_1, y_1) \text{ for all } x_1 \in X, y_1 \in Y\}$

$answer = c$

for each $(x, y) \in candidates$

$$p = \frac{\min\{m_X(x), m_Y(y)\}}{|\mathcal{X}|}$$

if $k \leq p$ and $d(x, y) < answer$, $answer = d(x, y)$

if $k > p$

$$\mathcal{X}^- = \mathcal{X} - (x, p|\mathcal{X}|)$$

$$\mathcal{Y}^- = \mathcal{Y} - (y, p|\mathcal{X}|)$$

$$k^- = \frac{k-p}{1-p}$$

if $\mathbf{d}(k^-, \mathcal{X}^-, \mathcal{Y}^-) < answer$, $answer = \mathbf{d}(k^-, \mathcal{X}^-, \mathcal{Y}^-)$

return $answer$

Note that k is used to “select” a proportion k of \mathcal{X} , just as k^- selects a proportion k^- of \mathcal{X}^- . The proportion of \mathcal{X}^- that is selected by k^- should equal the proportion of \mathcal{X} selected by k , minus that part of \mathcal{X} that was removed to get \mathcal{X}^- . Numerically, we want $|\mathcal{X}^-|k^- = |\mathcal{X}|k - \min\{m_X(x), m_Y(y)\}$. This is $k^- = \frac{|\mathcal{X}|(k-p)}{|\mathcal{X}^-|} = \frac{k-p}{1-p}$ as $|\mathcal{X}^-| = |\mathcal{X}| - \min\{m_X(x), m_Y(y)\} = |\mathcal{X}| - |\mathcal{X}|p$.

Note that, for finite \mathcal{X} , the algorithm converges as $|\mathcal{X}^-| < |\mathcal{X}|$ and $0 < k^- \leq k$ (as $k > p$). If $k = 1$ then $k^- = 1$, otherwise $k^- < k$.

Example 7.4. $d(x, y) = |x - y|$, $\mathcal{X} = \{1|1, 3|2, 4|2\}$ and $\mathcal{Y} = \{1|2, 5|2\}$.

To find $\mathcal{M}_{0.5}d(\mathcal{X}, \mathcal{Y})$ we choose $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ so that $d(x, y)$ is minimised.

We choose $x = 1$, $y = 1$ as $d(1, 1) = 0$.

Now, noting that $|\mathcal{X}| = 1 + 2 + 2 = 5$, we see that $k = 0.5$ is larger than $p = \frac{\min\{1, 2\}}{5} = \frac{1}{5} = 0.2$.

We have $\mathcal{X}^- = \{3|2, 4|2\}$, $\mathcal{Y}^- = \{1|1, 5|2\}$ and $k^- = \frac{5(0.5-0.2)}{4} = \frac{3}{8} = 0.375$.

To find $\mathcal{M}_{0.5}d(\mathcal{X}, \mathcal{Y})$ we find $\mathcal{M}_{0.375}d(\mathcal{X}^-, \mathcal{Y}^-)$, choosing $x = 4$, $y = 5$ as $d(4, 5) = 1$.

Now, as $|\mathcal{X}^-| = 4$, we see that $p = \frac{\min\{2, 2\}}{4} = 0.5$ is larger than $k^- = 0.375$ so $\mathcal{M}_{0.5}d(\mathcal{X}, \mathcal{Y}) = \mathcal{M}_{0.375}d(\mathcal{X}^-, \mathcal{Y}^-) = d(4, 5) = 1$.

The following example shows why, if $d(x_1, y_1) = d(x_2, y_2) = \min_{x \in X, y \in Y} d(x, y)$ it is important to choose the pair that minimises $\mathcal{M}_k d(\mathcal{X}, \mathcal{Y})$.

Example 7.5. $\mathcal{X} = \{x_1, x_2\}$, $\mathcal{Y} = \{y_1, y_2\}$, $d(x_1, y_1) = d(x_2, y_1) = 1$, $d(x_1, y_2) = 100$, $d(x_2, y_2) = 2$. Now If x_1 is paired with y_1 , $\mathcal{M}_k d(\mathcal{X}, \mathcal{Y}) = 2$, but if x_2 is paired with y_1 , $\mathcal{M}_k d(\mathcal{X}, \mathcal{Y}) = 100$. So, without the requirement to choose the $x \in \mathcal{X}, y \in \mathcal{Y}$ that *minimises* $\mathcal{M}_k d(\mathcal{X}, \mathcal{Y})$, the definition for $\mathcal{M}_k d$ would be ambiguous.

7.4.1 Span and $G\Delta I$ for $\mathcal{M}_k d$

$s_{\mathcal{M}_k d}(\mathcal{X})$ —the $\mathcal{M}_k d$ —*span* of \mathcal{X} definition.

If, for some y , $\mathcal{X} = \{y\}$ then $s_{\mathcal{M}_k d}(\mathcal{X}) = s_d(y)$.

If, for all y , $\mathcal{X} \neq \{y\}$ then $s_{\mathcal{M}_k d}(\mathcal{X}) = \max_{x, y \in \mathcal{X}} \{s_d(x) + d(x, y) + s_d(y)\}$.

This is the same definition of span as that for d_{ij}^M (see section 5.2.4)—except $\mathcal{M}_k d$ replaces d_{ij}^M . This definition of span is also used for $\mathcal{M}_{av} d$ —an L -collection distance function discussed in section 7.6.

Note that, even if d satisfies ΔI over M , $\mathcal{M}_k d$ where $\mathcal{M} = (M, m)$ and $k \neq 1$ may *not* satisfy $G\Delta I$ over $\mathcal{P}(\mathcal{M})$. In particular, $\mathcal{M}_k d(\mathcal{X}, \mathcal{Y})$ may be greater than $\mathcal{M}_k d(\mathcal{X}, \mathcal{Z}) + s(\mathcal{M}_k d, \mathcal{Z}) + \mathcal{M}_k d(\mathcal{Z}, \mathcal{Y})$ when $|\mathcal{Y}| < k|\mathcal{X}| \leq |\mathcal{Z}|$, but $k|\mathcal{Z}| \leq |\mathcal{Y}|$ (for $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \subseteq \mathcal{M}$).

Example 7.6. $d(x, y) = |x - y|$, $\mathcal{X} = \{^c 1 | 1\}^{\natural}$, $\mathcal{Z} = \{^c 1 | k\}^{\natural}$, $\mathcal{Y} = \{^c 1 | k^2\}^{\natural}$, $\max_{x, y \in M} d(x, y) = 1000$. Now $\mathcal{M}_k d(\mathcal{X}, \mathcal{Z}) = d(1, 1) = 0$, $\mathcal{M}_k d(\mathcal{Z}, \mathcal{Y}) = d(1, 1) = 0$ but, for $k < 1$, $\mathcal{M}_k d(\mathcal{X}, \mathcal{Y}) = \mathcal{M}_{k^-} d(\mathcal{X}^-, \emptyset) = \max_{x, y \in M} d(x, y) = 1000$.

Lemma 7.1 *If d satisfies $G\Delta I$ over a set M , $X, Y, Z \subseteq M$ and $|X| \leq |Z| \leq |Y|$ then $\mathcal{M}_k d(X, Y) \leq \mathcal{M}_k d(X, Z) + s_{\mathcal{M}_k d}(Z) + \mathcal{M}_k d(Z, Y)$.*

PROOF. If $X = \emptyset$ the result is obvious. Assuming $X \neq \emptyset$, label each element of X and each element of Y so that x_i^Y is the i^{th} element of X and ${}^X y_i$ is the i^{th} element of Y chosen when determining ${}_k^M d(X, Y)$. So, if

$$n = f_k(X) = \begin{cases} 1, & k|X|=0 \\ \lceil k|X| \rceil, & k|X| \neq 0 \end{cases}$$

then ${}_k^M d(X, Y) = d(x_n^Y, {}^X y_n)$, ${}_k^M d(X, Z) = d(x_n^Z, {}^X z_n)$.

If $m = f_k(Z)$, ${}_k^M d(Z, Y) = d(z_m^Y, {}^Z y_m)$. Note that $n \leq m$ as $|X| \leq |Z|$.

There are n distances

$$d(x_1^Z, {}^X z_1) \leq d(x_2^Z, {}^X z_2) \leq \dots \leq d(x_n^Z, {}^X z_n)$$

and m distances

$$d(z_1^Y, {}^Z y_1) \leq d(z_2^Y, {}^Z y_2) \leq \dots \leq d(z_m^Y, {}^Z y_m)$$

giving nm distances

$$d(x_1^Z, {}^Z y_1), \dots, d(x_1^Z, {}^Z y_m), \dots, d(x_n^Z, {}^Z y_m)$$

where, from $G\Delta I$, (for each $0 < i \leq n$, $0 < j \leq m$)

$$d(x_i^Z, {}^Z y_j) \leq d(x_i^Z, z_j^Y) + s_d(z_j^Y) + d(z_j^Y, {}^Z y_j),$$

$$d(x_i^Z, z_j^Y) \leq d(x_i^Z, {}^X z_i) + s_d({}^X z_i) + d({}^X z_i, z_j^Y)$$

and so

$$d(x_i^Z, {}^Z y_j) \leq d(x_i^Z, {}^X z_i) + s_d({}^X z_i) + d({}^X z_i, z_j^Y) + s_d(z_j^Y) + d(z_j^Y, {}^Z y_j).$$

From this and the definition of $s_k^M d$ -span

$$d(x_i^Z, {}^Z y_j) \leq d(x_i^Z, {}^X z_i) + s_k^M d(Z) + d(z_j^Y, {}^Z y_j).$$

So there are at least mn $x \in X, y \in Y$ combinations such that

$$d(x, y) \leq d(x_n^Z, x_n^X) + s_{M_d}(Z) + d(z_m^Y, z_m^Z).$$

A maximum of $(n - 1)$ x s and $(m - 1)$ y s, may have been chosen before x_n^Y, x_n^X leaving at least one combination available.

Theorem 7.2 *If d satisfies $G\Delta I$ over a set M then $\mathcal{M}_1^M d$ satisfies $G\Delta I$ over $\mathcal{P}(M)$.*

PROOF. For $X, Y, Z \subseteq M$. If $|X| > |Z|$ or $|Z| > |Y|$ then $\mathcal{M}_1^M d(X, Z)$ or $\mathcal{M}_1^M d(Z, Y)$ is $\max_{x, y \in M} d(x, y)$. So ΔI is satisfied when $|X| > |Z|$ or $|Z| > |Y|$. The only remaining case is $|X| \leq |Z| \leq |Y|$ which is covered in lemma 7.1.

Lemma 7.3 *d^+ satisfies $G\Delta I$ over $M \cup \{v\}$ if d satisfies $G\Delta I$ over M and there is a $u \in M$ such that, for all $x, y \in M$, $d^+(x, y) = d(x, y)$, $d^+(x, v) = d(x, u)$, $d^+(v, y) = d(u, y)$, $s_{d^+}(v) = s_d(u)$ and $d^+(v, v) = d(u, u)$.*

PROOF. For all $x, y, z \in M$:

$$\begin{aligned} d^+(v, y) &\leq d^+(v, z) + s_{d^+}(z) + d^+(z, y) \text{ as } d(u, y) \leq d(u, z) + s_d(z) + d(z, y), \\ d^+(x, y) &\leq d^+(x, v) + s_{d^+}(v) + d^+(v, y) \text{ as } d(x, y) \leq d(x, u) + s_d(u) + d(u, y), \\ d^+(x, v) &\leq d^+(x, z) + s_{d^+}(z) + d^+(z, v) \text{ as } d(x, u) \leq d(x, z) + s_d(z) + d(z, u). \end{aligned}$$

Lemma 7.4 *If d satisfies $G\Delta I$ over a finite set M and $\mathcal{M} = (M, m)$ is an \mathbb{N}_1 -collection then $\mathcal{M}_1^M d$ satisfies $G\Delta I$ over $\mathcal{P}(\mathcal{M})$.*

PROOF. Let $M = \{x_1, \dots, x_n\}$, $M^+ = \{x_{1\ 1}, \dots, x_{1\ n_1}, \dots, x_{n\ 1}, \dots, x_{n\ n_n}\}$ and d^+ be a distance function over M^+ where $d^+(x_{ir}, x_{ir'}) = d(x_i, x_i)$, $s_{d^+}(x_{ir}) = s_d(x_i)$ and $d^+(x_{ir}, x_{jr''}) = d(x_i, x_j)$ for any $1 \leq i, j \leq n$, $1 \leq r, r' \leq n_i$ and $1 \leq r'' \leq n_j$.

From lemma 7.3, d^+ satisfies $G\Delta I$.

From theorem 7.2, $\mathcal{M}_1^{M^+} d^+$ satisfies $G\Delta I$.

For each $1 \leq i \leq n$, let $m(x_i) = n_i$. Each $\mathcal{X} = (X, m_X) \subseteq \mathcal{M}$ has a corresponding $X^+ \subseteq M^+$ where $x_i \in \mathcal{X} \Leftrightarrow x_{i1}, \dots, x_{i m_X(x_i)} \in X^+$.

Now, for any $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$, ${}_1^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}) = {}_1^{M^+}d^+(X^+, Y^+)$ and $s_{{}_1^{\mathcal{M}}d}(\mathcal{X}) = s_{{}_1^{M^+}d^+}(X^+)$.

Theorem 7.5 *If d satisfies $G\Delta I$ over a finite set M and $\mathcal{M} = (M, m)$ is an L -collection, where $L \subseteq \mathbb{Q}^{>0}$, then ${}_1^{\mathcal{M}}d$ satisfies $G\Delta I$ over any finite subset of $\mathcal{P}(\mathcal{M})$.*

PROOF. Let $M' = \{\mathcal{X}_1, \dots, \mathcal{X}_n\}$ be a finite subset of $\mathcal{P}(\mathcal{M})$. Because M' and each \mathcal{X}_i is finite, there is a $w \in \mathbb{N}_1$ where, for each $1 \leq i \leq n$, $w\mathcal{X}_i$ is an \mathbb{N}_1 -collection. From lemma 7.4, ${}_1^{w\mathcal{M}}d$ satisfies $G\Delta I$ over $\mathcal{P}(w\mathcal{M})$. For any $1 \leq i \leq n$, $1 \leq j \leq n$ and any $w \in \mathbb{R}^{>0}$ at all, ${}_1^{w\mathcal{M}}d(w\mathcal{X}_i, w\mathcal{X}_j) = {}_1^{\mathcal{M}}d(\mathcal{X}_i, \mathcal{X}_j)$ and $s_{{}_1^{w\mathcal{M}}d}(w\mathcal{X}_i) = s_{{}_1^{\mathcal{M}}d}(\mathcal{X}_i)$.

7.4.2 \subseteq -reflexivity for ${}_k^{\mathcal{M}}d$ distance functions

Theorem 7.6 *If the distance function d is (\subseteq) -reflexive then ${}_k^{\mathcal{M}}d$ is \subseteq -reflexive.*

PROOF. If $\mathcal{X} \subseteq \mathcal{Y}$ then $\min_{x \in \mathcal{X}, y \in \mathcal{Y}} d(x, y) = d(x, x) = 0$ so:

1. If $k \leq p$, ${}_k^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}) = 0$.
2. If $k > p$, ${}_k^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}) = {}_k^{\mathcal{M}}d(\mathcal{X}^-, \mathcal{Y}^-)$ where $\mathcal{X}^- \subseteq \mathcal{Y}^-$ as $\mathcal{X} \subseteq \mathcal{Y}$ and so $m_X(x) \leq m_Y(x)$. So ${}_k^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}) = \dots = {}_k^{\mathcal{M}}d(\mathcal{X}^*, \mathcal{Y}^*)$ where $k^* \leq p$ and $\mathcal{X}^* \subseteq \mathcal{Y}^*$.

Note that, if \subseteq^d -reflexivity was defined to mirror \mathcal{Z}^d -strict positiveness, ${}_k^{\mathcal{M}}d$ distance functions where $0 < k \leq 1$ would *not* be \subseteq^d -reflexive.

Example 7.7. d is some \mathcal{Z}^d -strict positive distance function over \mathbb{N}_1 . d' is some $\mathcal{Z}^{d'}$ -strict positive distance function, based on d , over $M = \mathcal{P}(\mathbb{N}_1)$. $d'' =$

$\mathcal{M}_k^M(d')$ is a distance function over $\mathcal{P}(M)$. $X = \{\{1\}, \{1, 2\}, \{1, 2, 3\}, \dots, \mathbb{N}_1\}$ and $Y = \{\mathbb{N}_1\}$. As $X \subseteq^{d''} Y$, for d'' to be $\subseteq^{d''}$ -reflexive we would require $d''(X, Y) = 0$ but, for $0 < k \leq 1$, $d''(X, Y) > 0$.

To fix this, \subseteq^d would need to be defined so, when evaluating $X \subseteq^d Y$, the elements of (the elements of...) X are “paired off” with the elements of (the elements of...) Y —similarly to how elements are paired off when evaluating $\mathcal{M}_k^M d$. Not only would this be quite complex, but then d_{ij}^M distance functions, which work best with the existing definition of \subseteq^d , would not be \mathcal{Z}^d -strict positive.

The complete solution then, would be to define two versions of \subseteq^d : the existing version, for d_{ij}^M distance functions; and a “pairing off” version, for $\mathcal{M}_k^M d$ distance functions. These different versions would then be used to define different versions of the \subseteq^d -reflexive and \mathcal{Z}^d -strict positive properties for d_{ij}^M and $\mathcal{M}_k^M d$ distance functions.

Unfortunately, the technical complexity of all this would obscure the main argument of this research, while adding little of value. If greater precision is required, the existing theorems can easily be adapted to the more specific forms of \subseteq^d -reflexivity and \mathcal{Z}^d -strict positiveness.

7.4.3 \mathcal{Z}^d -strict positiveness for $\mathcal{M}_k^M d$ distance functions

Even if d is strict positive, $\mathcal{M}_k^M d$ where $k \neq 1$ may *not* be \mathcal{Z} -strict positive. Indeed whenever $\mathcal{X} \supseteq \mathcal{Y}$, $k|\mathcal{X}| \leq |\mathcal{Y}|$ and d is (\subseteq^-) -reflexive, $\mathcal{M}_k^M d(\mathcal{X}, \mathcal{Y}) = 0$.

Example 7.8. $\mathcal{X} = \{1, 2, \dots, 10\}$, $\mathcal{Y} = \{1, 2, \dots, 9\}$, $d(x, y) = |x - y|$. To find $\mathcal{M}_{0.9}^M d(\mathcal{X}, \mathcal{Y})$ we can pair off $x = y = 1, x = y = 2, \dots, x = y = 9$ to get $\mathcal{M}_{0.9}^M d(\mathcal{X}, \mathcal{Y}) = d(9, 9) = 0$. By increasing the size of \mathcal{X} and \mathcal{Y} , similar examples can be found where $\mathcal{M}_k^M d(\mathcal{X}, \mathcal{Y}) = 0$ for any $\mathcal{M}_k^M d$ where $k < 1$.

Theorem 7.7 *If the distance function d is \mathcal{Z}^d -strict positive over M and $\mathcal{M} = (M, m)$ then, for $d' = \mathcal{M}_1^M d$, d' is $\mathcal{Z}^{d'}$ -strict positive over $\mathcal{P}(\mathcal{M})$.*

PROOF. For any $\mathcal{X} = (X, m_X), \mathcal{Y} = (Y, m_Y)$, when determining ${}_1^M d(\mathcal{X}, \mathcal{Y})$ distances $d(x, y)$ ($x \in X, y \in Y$) are chosen in nondecreasing order. If $|\mathcal{X}| > |\mathcal{Y}|$, ${}_1^M d(\mathcal{X}, \mathcal{Y}) > 0$. If $|\mathcal{X}| \leq |\mathcal{Y}|$, ${}_1^M d(\mathcal{X}, \mathcal{Y})$ is the maximum of these chosen $d(x, y)$ s. If $\mathcal{X} \not\subseteq^d \mathcal{Y}$ there is at least one $x \in X$ such that $x \not\subseteq^d y$ for all y . As d is \subseteq^d -strict positive, $d(x, y) > 0$ and so ${}_1^M d(\mathcal{X}, \mathcal{Y}) > 0$.

7.5 The ${}_av^M d$ L -Collection Distance Function

Informally, ${}_k^M d(X, Y)$, where $0 \leq k \leq 1$, chooses pairs elements $x \in X$ with elements $y \in Y$ —removing chosen pairs—so that $d(x, y)$ is nondecreasing with each choice. ${}_k^M d(X, Y) = d(x, y)$ for the $k|X|^{th}$ x, y pair. If Y is exhausted before the $k|X|^{th}$ pair is chosen, ${}_k^M d(X, Y) = \max_{x, y \in M} d(x, y)$.

Equally informally, ${}_av^M d(X, Y)$ is the average of these $d(x, y)$ distances. If $|X| > |Y|$ then $(|X| - |Y|) \max_{x, y \in M} d(x, y)$ is added to the sum of the $d(x, y)$ distances before dividing this by $|X|$ —as if each unpaired element of X where paired with \emptyset .

The L -collection distance function ${}_av^M d$ essentially works in the same way, but assigns weights to each $d(x, y)$ that reflect the membership grade of the paired elements.

Definition of ${}_av^M d(\mathcal{X}, \mathcal{Y})$ where $\mathcal{M} = (M, m)$ is an L -collection, d is a distance function over M and $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$.

$${}_av^M d(\emptyset, \mathcal{Y}) = 0.$$

$$\text{If } \mathcal{X} \neq \emptyset, {}_av^M d(\mathcal{X}, \emptyset) = \max_{x, y \in M} d(x, y).$$

If $\mathcal{X}, \mathcal{Y} \neq \emptyset$: Let $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$. Choose $x \in X$ and $y \in Y$ where $d(x, y) = \min_{x_1 \in X, y_1 \in Y} d(x_1, y_1)$. If there is more than one such pair, choose the pair that minimises ${}_av^M d(\mathcal{X}, \mathcal{Y})$.

$$\text{Let } p = \frac{\min\{m_X(x), m_Y(y)\}}{|\mathcal{X}|}.$$

$${}_av^M d(\mathcal{X}, \mathcal{Y}) = pd(x, y) + (1 - p){}_av^M d(\mathcal{X}^-, \mathcal{Y}^-)$$

where $\mathcal{X}^- = \mathcal{X} - (x, p|\mathcal{X}|)$ and $\mathcal{Y}^- = \mathcal{Y} - (y, p|\mathcal{X}|)$.

Example 7.9. $\mathcal{X} = \{9, 19, 29\}$, $\mathcal{Y} = \{10, 20\}$, $\mathcal{M} = \{1, \dots, 30\}$, $d(x, y) = |x - y|$.

As sets can be treated as $\{1\}$ -collections we have

$$\begin{aligned}\mathcal{M}_{av}d(\mathcal{X}, \mathcal{Y}) &= \frac{1}{3}d(9, 10) + \frac{2}{3} \left(\frac{1}{2}d(19, 20) + \frac{1}{2}d(29, \emptyset) \right) \\ &= \frac{1}{3}d(9, 10) + \frac{1}{3}d(19, 20) + \frac{1}{3}d(1, 30) = 10\frac{1}{3}.\end{aligned}$$

Note that, from the definition, if $\mathcal{X} \neq \emptyset$, $\mathcal{M}_{av}d(\mathcal{X}, \emptyset) = \max_{x, y \in M} d(x, y)$, so in the example $d(29, \emptyset) = d(1, 30)$. Also, if $q_i = \min\{m_X(x), m_Y(y)\}$ and $d_i = d(x, y)$ on the i^{th} recursion then

$$\begin{aligned}\mathcal{M}_{av}d(\mathcal{X}, \mathcal{Y}) &= \frac{q_1}{|\mathcal{X}|}d_1 + \left(1 - \frac{q_1}{|\mathcal{X}|}\right) \left(\frac{q_2}{|\mathcal{X}| - q_1}d_2 + \left(1 - \frac{q_2}{|\mathcal{X}| - q_1}\right) \left(\frac{q_3}{|\mathcal{X}| - q_1 - q_2}d_3 + \dots \right) \right) \\ &= \frac{q_1}{|\mathcal{X}|}d_1 + \frac{|\mathcal{X}| - q_1}{|\mathcal{X}|} \left(\frac{q_2}{|\mathcal{X}| - q_1}d_2 + \frac{(|\mathcal{X}| - q_1) - q_2}{|\mathcal{X}| - q_1} \left(\frac{q_3}{|\mathcal{X}| - q_1 - q_2}d_3 + \dots \right) \right) \\ &= \frac{q_1}{|\mathcal{X}|}d_1 + \frac{q_2}{|\mathcal{X}|}d_2 + \frac{q_3}{|\mathcal{X}|}d_3 + \dots\end{aligned}$$

If r is the final recursion and $|\mathcal{X}| \leq |\mathcal{Y}|$ then the last term in the series will be $\frac{q_r}{|\mathcal{X}|}d_r$ and

$$\sum_{i=1}^r \frac{q_i}{|\mathcal{X}|} = 1.$$

If $|\mathcal{X}| > |\mathcal{Y}|$ then the last term will be $\frac{|\mathcal{X}| - |\mathcal{Y}|}{|\mathcal{X}|} \max_{x, y \in M} d(x, y)$ and

$$\sum_{i=1}^{r-1} \frac{q_i}{|\mathcal{X}|} = 1 - \frac{|\mathcal{X}| - |\mathcal{Y}|}{|\mathcal{X}|} = \frac{|\mathcal{Y}|}{|\mathcal{X}|}.$$

Note that *scaling* $\mathcal{X}, \mathcal{Y} \in \mathcal{M}$ by a constant will not affect the result.

Lemma 7.8 *If \mathcal{M} is an L -collection, where $L \subseteq \mathbb{Q}^{>0}$, $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$ and $w \in \mathbb{Q}^{>0}$ then $s_{\text{av}}^{\mathcal{M}d}(\mathcal{X}) = s_{\text{av}}^{w\mathcal{M}d}(w\mathcal{X})$ and ${}^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}) = {}^{w\mathcal{M}}d(w\mathcal{X}, w\mathcal{Y})$.*

PROOF. Let $\mathcal{M} = (M, m)$, $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$. Now $w\mathcal{M} = (M, wm)$, $w\mathcal{X} = (X, wm_X)$, $w\mathcal{Y} = (Y, wm_Y)$, $w\mathcal{X}, w\mathcal{Y} \subseteq w\mathcal{M}$ and

$$s_{\text{av}}^{w\mathcal{M}d}(w\mathcal{X}) = s_{\text{av}}^{\mathcal{M}d}(X) = s_{\text{av}}^{\mathcal{M}d}(\mathcal{X}).$$

Let $q_i = \min\{m_X(x), m_Y(y)\}$, $d_i = d(x, y)$ on the i^{th} recursion, and $r =$ the final recursion when determining ${}^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y})$.

If $|\mathcal{X}| \leq |\mathcal{Y}|$ then

$${}^{w\mathcal{M}}d(w\mathcal{X}, w\mathcal{Y}) = \sum_{i=1}^r \frac{wq_i}{|w\mathcal{X}|} d_i = \sum_{i=1}^r \frac{q_i}{|\mathcal{X}|} d_i = {}^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}).$$

If $|\mathcal{X}| > |\mathcal{Y}|$ then

$$\begin{aligned} {}^{w\mathcal{M}}d(w\mathcal{X}, w\mathcal{Y}) &= \left(\sum_{i=1}^{r-1} \frac{wq_i}{|w\mathcal{X}|} d_i \right) + \frac{|w\mathcal{X}| - |w\mathcal{Y}|}{|w\mathcal{X}|} \max_{x, y \in M} d(x, y) \\ &= \left(\sum_{i=1}^{r-1} \frac{q_i}{|\mathcal{X}|} d_i \right) + \frac{|\mathcal{X}| - |\mathcal{Y}|}{|\mathcal{X}|} \max_{x, y \in M} d(x, y) = {}^{\mathcal{M}}d(\mathcal{X}, \mathcal{Y}). \end{aligned}$$

7.5.1 $G\Delta I$ for ${}^{\mathcal{M}}d$

Theorem 7.9 *If d satisfies $G\Delta I$ over a finite set M then ${}^{\mathcal{M}}d$ satisfies $G\Delta I$ over $\mathcal{P}(M)$.*

In the following proof, $X^Y \subseteq X$ denotes the set of all points in X , paired with a point in Y when determining ${}^{\mathcal{M}}d(X, Y)$. This defines $X^Y, X^Z \subseteq X$ and $Z^Y \subseteq Z$. Similarly, ${}^X Y \subseteq Y$ denotes the set of all points in Y that a point in X^Y is paired with when determining ${}^{\mathcal{M}}d(X, Y)$. This defines ${}^X Y, {}^Z Y \subseteq Y$ and ${}^X Z \subseteq Z$.

Note that if X is smaller than Y and Z , $X^Z = X^Y = X$. Similarly, if Y is smaller than X and Z , $^XY = ^ZY = Y$.

The sets X and Y (and so their subsets) are (fully) indexed with the constraint that (x_i, y_i) is the i^{th} pair chosen when determining $^M_{\text{av}}d(X, Y)$, so if $|X| > |Y|$, $x_{|Y|+1}, \dots, x_{|X|}$ are in an arbitrary order. Z is (fully) indexed with the constraint that, for each i where $x_i \in X^Z$, x_i is paired with z_i when determining $^M_{\text{av}}d(X, Z)$. The function f is defined so, for each j where $z_j \in Z^Y$, z_j is paired with $y_{f(j)}$ when determining $^M_{\text{av}}d(Z, Y)$.

Note that, while it is always possible to sequentially index X and Y , so $X = \{x_1, x_2, \dots, x_{|X|}\}$ and $Y = \{y_1, y_2, \dots, y_{|Y|}\}$, it is normally not possible to sequentially index Z when $|Z| < |X|$. That is, Z cannot, generally be represented by $\{z_1, \dots, z_{|Z|}\}$ as the indices are chosen based on the ordering of X .

PROOF. For $X, Y, Z \subseteq M$, there are 6 cases to consider: $|X| \leq |Z| \leq |Y|$, $|X| \leq |Y| \leq |Z|$, $|Z| \leq |X| \leq |Y|$, $|Y| \leq |X| \leq |Z|$, $|Y| \leq |Z| \leq |X|$ and $|Z| \leq |Y| \leq |X|$.

If $|X| \leq |Y|$, for every $Y' \subseteq ^ZY$ of size $\min\{|X|, |Z|\}$, for each $y \in Y'$, there is a distinct $x_i \in X = X^Y$ where:

1. if $y \in ^ZY \cap ^XY$, $y = y_i$ (and so $d(x_i, y_i) = d(x_i, y)$);
2. if $y \in ^ZY - ^XY$, $d(x_i, y_i) \leq d(x_i, y)$ (otherwise y would be $\in ^XY$).

So for each $y \in Y'$ a distinct x_i can be chosen to satisfy $d(x_i, y_i) \leq d(x_i, y)$. From this and $G\Delta I$ applied twice, for $\min\{|X|, |Z|\}$ of the x_i s in X , $z, z' \in Z$ and $(y = y_i \text{ or } y \in ^ZY - ^XY)$:

$$d(x_i, y_i) \leq d(x_i, z) + s_d(z) + d(z, z') + s_d(z') + d(z', y).$$

As $s^M_{\text{av}}d(Z) \geq s_d(z) + d(z, z') + s_d(z')$ for all $z, z' \in Z$,

$$d(x_i, y_i) \leq d(x_i, z) + s^M_{\text{av}}d(Z) + d(z', y). \quad (7.1)$$

Case 1 $|X| \leq |Z| \leq |Y|$. We have

$$\begin{aligned} {}^M_{\text{av}}d(X, Y) &= \frac{1}{|X|} \sum_{i=1}^{|X|} d(x_i, y_i), \\ {}^M_{\text{av}}d(X, Z) &= \frac{1}{|X|} \sum_{i=1}^{|X|} d(x_i, z_i) \text{ and} \\ {}^M_{\text{av}}d(Z, Y) &= \frac{1}{|Z|} \sum_{j=1}^{|Z|} d(z_j, y_{f(j)}). \end{aligned}$$

As $|Y'| = |X|$, inequality (7.1) can be summed to give, with z_i for z ,

$$\sum_{i=1}^{|X|} d(x_i, y_i) \leq \sum_{i=1}^{|X|} d(x_i, z_i) + |X| s_{\text{av}}^M d(Z) + \sum_{y \in Y'} d(z', y).$$

As this holds for any $Y' \subseteq {}^Z Y$, we can have $|Z|$, not necessarily distinct Y' s, the $(\mathbb{N}_1\text{-collection})$ union of which contains each $y \in {}^Z Y$ exactly $|X|$ times. Adding these $|Z|$ inequalities, with z_j for z' , gives

$$|Z| \left(\sum_{i=1}^{|X|} d(x_i, y_i) \right) \leq \left(\sum_{i=1}^{|X|} \sum_{j=1}^{|Z|} d(x_i, z_i) + s_{\text{av}}^M d(Z) + d(z_j, y_{f(j)}) \right).$$

Dividing through by $|Z||X|$ gives ${}^M_{\text{av}}d(X, Y) \leq {}^M_{\text{av}}d(X, Z) + s_{\text{av}}^M d(Z) + {}^M_{\text{av}}d(X, Y)$.

The other cases are similar, so are presented in less detail.

Case 2 $|X| \leq |Y| \leq |Z|$. We have

$$\begin{aligned} {}^M_{\text{av}}d(X, Y) &= \frac{1}{|X|} \sum_{i=1}^{|X|} d(x_i, y_i), \\ {}^M_{\text{av}}d(X, Z) &= \frac{1}{|X|} \sum_{i=1}^{|X|} d(x_i, z_i) \text{ and} \\ {}^M_{\text{av}}d(Z, Y) &= \frac{1}{|Z|} \left(\sum_{\{f(j)|y_{f(j)} \in Y\}} d(z_j, y_{f(j)}) \right) + \frac{|Z|-|Y|}{|Z|} \max_{x,y \in M} d(x, y). \end{aligned}$$

For any $z_j, y \in M$, $d(z_j, y) \leq \max_{x,y' \in M} \{d(x, y')\}$. Similarly to case 1, from inequality (7.1) applied $|X||Z|$ times, with $\max_{x,y \in M} d(x, y)$ for $d(z_j, y)$ $|X|(|Z| - |Y|)$ times,

$$\begin{aligned}
|Z| \left(\sum_{i=1}^{|X|} d(x_i, y_i) \right) &\leq |Z| \left(\sum_{i=1}^{|X|} d(x_i, z_i) + s_{\text{av}}^M d(Z) \right) \\
&\quad + |X| \left(\sum_{\{f(j)|y_{f(j)} \in Y\}} d(z_j, y_{f(j)}) \right) + |X|(|Z| - |Y|) \max_{x,y \in M} d(x, y).
\end{aligned}$$

Dividing through by $|Z||X|$ gives $\frac{M}{\text{av}} d(X, Y) \leq \frac{M}{\text{av}} d(X, Z) + s_{\text{av}}^M d(Z) + \frac{M}{\text{av}} d(X, Y)$.

Case 3 $|Z| \leq |X| \leq |Y|$.

$$\begin{aligned}
\frac{M}{\text{av}} d(X, Y) &= \frac{1}{|X|} \sum_{i=1}^{|X|} d(x_i, y_i), \\
\frac{M}{\text{av}} d(X, Z) &= \left(\frac{1}{|X|} \sum_{\{i|x_i \in X^Z\}} d(x_i, z_i) \right) + \frac{|X| - |Z|}{|X|} \max_{x,y \in M} d(x, y) \text{ and} \\
\frac{M}{\text{av}} d(Z, Y) &= \frac{1}{|Z|} \sum_{j=1}^{|Z|} d(z_j, y_{f(j)}).
\end{aligned}$$

Similarly to case 2, from inequality (7.1) applied $|Z|^2$ times, with $\max_{x,y \in M} d(x, y)$ for $d(x_i, z_i)$ the remaining $|Z|(|X| - |Z|)$ times,

$$\begin{aligned}
|Z| \left(\sum_{i=1}^{|X|} d(x_i, y_i) \right) &\leq |Z| \left(\sum_{\{i|x_i \in X^Z\}} d(x_i, z_i) \right) + |Z|(|X| - |Z|) \max_{x,y \in M} d(x, y) \\
&\quad + |X| \left(\sum_{j=1}^{|Z|} s_{\text{av}}^M d(Z) + d(z_j, y_{f(j)}) \right).
\end{aligned}$$

Dividing through by $|Z||X|$ gives $\frac{M}{\text{av}} d(X, Y) \leq \frac{M}{\text{av}} d(X, Z) + s_{\text{av}}^M d(Z) + \frac{M}{\text{av}} d(X, Y)$.

If $|Y| \leq |X|$, for every $X' \subseteq X^Z$ of size $\min\{|X|, |Z|\}$, for each $x \in X'$, there is a distinct $y_i \in Y = {}^X Y$ where:

1. if $x \in X^Z \cap X^Y$, $x = x_i$ (and so $d(x_i, y_i) = d(x, y_i)$);
2. if $x \in X^Z - X^Y$, $d(x_i, y_i) \leq d(x, y_i)$ (otherwise x would be $\in X^Y$).

From this and G Δ I applied twice, for $\min\{|Y|, |Z|\}$ of the y_i s in ${}^X Y$, $z, z' \in Z$ and $(x = x_i \text{ or } x \in X^Z - X^Y)$:

$$d(x_i, y_i) \leq d(x, z) + s_d(z) + d(z, z') + s_d(z') + d(z', y_i).$$

As $s_{\text{av}}^M d(Z) \geq s_d(z) + d(z, z') + s_d(z')$ for all $z, z' \in Z$,

$$d(x_i, y_i) \leq d(x, z) + s_{\text{av}}^M d(Z) + d(z', y_i). \quad (7.2)$$

Case 4 $|Y| \leq |X| \leq |Z|$.

$$\begin{aligned} s_{\text{av}}^M d(X, Y) &= \left(\frac{1}{|X|} \sum_{i=1}^{|Y|} d(x_i, y_i) \right) + \frac{|X|-|Y|}{|X|} \max_{x,y \in M} d(x, y), \\ s_{\text{av}}^M d(X, Z) &= \frac{1}{|X|} \sum_{i=1}^{|X|} d(x_i, z_i) \text{ and} \\ s_{\text{av}}^M d(Z, Y) &= \frac{1}{|Z|} \sum_{j=1}^{|Y|} d(z_j, y_{f(j)}) + \frac{|Z|-|Y|}{|Z|} \max_{x,y \in M} d(x, y). \end{aligned}$$

Similarly to cases 2 and 3, as $|Z|(|X| - |Y|) \leq |X|(|Z| - |Y|)$, from inequality (7.2) applied $|X||Z|$ times, with $\max_{x,y \in M} d(x, y)$ for $d(x_i, z_i)$, $|Z|(|X| - |Y|)$ times and for $d(z_j, y_i)$, $|X|(|Z| - |Y|)$ times,

$$\begin{aligned} &|Z| \left(\sum_{i=1}^{|Y|} d(x_i, y_i) \right) + |Z|(|X| - |Y|) \max_{x,y \in M} d(x, y) \\ &\leq |Z| \left(\sum_{i=1}^{|X|} d(x_i, z_i) + s_{\text{av}}^M d(Z) \right) + |X| \left(\sum_{j=1}^{|Y|} d(z_j, y_{f(j)}) \right) + |X|(|Z| - |Y|) \max_{x,y \in M} d(x, y). \end{aligned}$$

Dividing through by $|Z||X|$ gives $s_{\text{av}}^M d(X, Y) \leq s_{\text{av}}^M d(X, Z) + s_{\text{av}}^M d(Z) + s_{\text{av}}^M d(Z, Y)$.

Case 5 $|Y| \leq |Z| \leq |X|$.

$$\begin{aligned} s_{\text{av}}^M d(X, Y) &= \left(\frac{1}{|X|} \sum_{i=1}^{|Y|} d(x_i, y_i) \right) + \frac{|X|-|Y|}{|X|} \max_{x,y \in M} d(x, y), \\ s_{\text{av}}^M d(X, Z) &= \left(\frac{1}{|X|} \sum_{\{i|z_i \in Z\}} d(x_i, z_i) \right) + \frac{|X|-|Z|}{|X|} \max_{x,y \in M} d(x, y) \text{ and} \\ s_{\text{av}}^M d(Z, Y) &= \left(\frac{1}{|Z|} \sum_{f(j)=1}^{|Y|} d(z_j, y_{f(j)}) \right) + \frac{|Z|-|Y|}{|Z|} \max_{x,y \in M} d(x, y). \end{aligned}$$

Similarly to case 4, as $|Z|(|X| - |Y|) \leq |Z|(|X| - |Z|) + |X|(|Z| - |Y|)$, from inequality (7.2) applied $|X||Z|$ times, with $\max_{x,y \in M} d(x, y)$ for:

1. $d(x_i, y_i)$, $|Z|(|X| - |Y|)$ times;
2. for $d(x_i, z_i)$, $|Z|(|X| - |Z|)$ times;
3. for $d(z_j, y_i)$, $|X|(|Z| - |Y|)$ times.

$$\begin{aligned}
& |Z| \left(\sum_{i=1}^{|Y|} d(x_i, y_i) \right) + |Z|(|X| - |Y|) \max_{x, y \in M} d(x, y) \\
& \leq |Z| \left(\sum_{i=1}^{|X|} d(x_i, z_i) \right) + |Z|(|X| - |Y|) \max_{x, y \in M} d(x, y) \\
& + |Z||X| s_{\text{av}}^M(Z) + |X| \left(\sum_{j=1}^{|Y|} d(z_j, y_{f(j)}) \right) + |X|(|Z| - |Y|) \max_{x, y \in M} d(x, y).
\end{aligned}$$

Dividing through by $|Z||X|$ gives $\frac{M}{\text{av}}d(X, Y) \leq \frac{M}{\text{av}}d(X, Z) + s_{\text{av}}^M d(Z) + \frac{M}{\text{av}}d(X, Y)$.

Case 6 $|Z| \leq |Y| \leq |X|$.

$$\begin{aligned}
\frac{M}{\text{av}}d(X, Y) &= \left(\frac{1}{|X|} \sum_{i=1}^{|Y|} d(x_i, y_i) \right) + \frac{|X|-|Y|}{|X|} \max_{x, y \in M} d(x, y), \\
\frac{M}{\text{av}}d(X, Z) &= \left(\frac{1}{|X|} \sum_{\{i|z_i \in Z\}} d(x_i, z_i) \right) + \frac{|X|-|Z|}{|X|} \max_{x, y \in M} d(x, y) \text{ and} \\
\frac{M}{\text{av}}d(Z, Y) &= \frac{1}{|Z|} \sum_{\{j|z_j \in Z\}} d(z_j, y_{f(j)}).
\end{aligned}$$

Similarly to cases 3 and 5, as $|Z|(|X| - |Y|) \leq |Z|(|X| - |Z|)$, from inequality (7.2) applied $|Z|^2$ times, with $\max_{x, y \in M} d(x, y)$ for $d(x, z_i)$, $|Z|(|X| - |Z|)$ times and for $d(x_i, y_i)$ the remaining $|Z|(|X| - |Y|)$ times,

$$\begin{aligned}
& |Z| \left(\sum_{i=1}^{|Y|} d(x_i, y_i) \right) + |Z|(|X| - |Y|) \max_{x, y \in M} d(x, y) \\
& \leq |Z| \left(\sum_{\{i|z_i \in Z\}} d(x_i, z_i) \right) + |Z|(|X| - |Z|) \max_{x, y \in M} d(x, y) \\
& \quad + |X| \left(\sum_{j=1}^{|Y|} s_{\text{av}}^M(Z) + d(z_j, y_{f(j)}) \right).
\end{aligned}$$

Dividing through by $|Z||X|$ gives $\frac{M}{\text{av}}d(X, Y) \leq \frac{M}{\text{av}}d(X, Z) + s_{\text{av}}^M d(Z) + \frac{M}{\text{av}}d(X, Y)$.

Lemma 7.10 *If*

1. d satisfies $G\Delta I$ over $M = \{z_1, \dots, z_{|M|}\}$,
2. $\mathcal{M} = (M, m)$,
3. $M^+ = \{z_{1\ 1}, \dots, z_{1\ m(z_1)}, \dots, z_{|Z|\ 1}, \dots, z_{|Z|\ m(z_{|Z|})}\}$,
4. $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$ are \mathbb{N}_1 -collections where $\mathcal{X} = (X, m_X)$, $\mathcal{Y} = (Y, m_Y)$ and $X = \{x_1, \dots, x_{|X|}\}$, $Y = \{y_1, \dots, y_{|Y|}\}$,
5. $X^+ \subseteq M^+$ where $X^+ = \{x_{1\ 1}, \dots, x_{1\ m_X(x_1)}, \dots, x_{|X|\ 1}, \dots, x_{|X|\ m_X(x_{|X|})}\}$ and, for any $1 \leq i \leq |X|$, $1 \leq j \leq |M|$, $1 \leq r \leq m_X(x_i)$ and $1 \leq r' \leq m(z_j)$, $d^+(x_{ir}, z_{jr'}) = d(x_i, z_j)$, $d^+(z_{jr'}, x_{ir}) = d(z_j, x_i)$ and $s_{d^+}(x_{ir}) = s_d(x_i)$.
6. $Y^+ \subseteq M^+$ where $Y^+ = \{y_{1\ 1}, \dots, y_{1\ m_Y(y_1)}, \dots, y_{|Y|\ 1}, \dots, y_{|Y|\ m_Y(y_{|Y|})}\}$ and, for any $1 \leq i \leq |Y|$, $1 \leq j \leq |M|$, $1 \leq r \leq m_Y(y_i)$ and $1 \leq r' \leq m(z_j)$, $d^+(y_{ir}, z_{jr'}) = d(y_i, z_j)$, $d^+(z_{jr'}, y_{ir}) = d(z_j, y_i)$ and $s_{d^+}(y_{ir}) = s_d(y_i)$.

then $s_{\text{av}}^{\mathcal{M}d}(\mathcal{X}) = s_{\text{av}}^{M^+d^+}(X^+)$ and $\mathcal{M}d(\mathcal{X}, \mathcal{Y}) = \mathcal{M}^{M^+d^+}(X^+, Y^+)$.

PROOF.

$$\begin{aligned} s_{\text{av}}^{\mathcal{M}d}(\mathcal{X}) &= \max_{x, y \in X} \{s_d(x) + d(x, y) + s_d(y)\} \\ &= \max_{x, y \in X^+} \{s_d(x) + d(x, y) + s_d(y)\} = s_{\text{av}}^{M^+d^+}(X^+). \end{aligned}$$

Let x_1 and y_1 be the first elements of X and Y chosen when determining $\mathcal{M}d(\mathcal{X}, \mathcal{Y})$. Let $t = \min\{m_X(x_1), m_Y(y_1)\}$. Note that $|X^+| = \sum_{r=1}^{|X|} m_X(x_r) = |\mathcal{X}|$.

$$\begin{aligned} & \mathcal{M}^{M^+d^+}(X^+, Y^+) \\ &= \frac{1}{|X^+|} d^+(x_{1\ 1}, y_{1\ 1}) + \frac{|X^+| - 1}{|X^+|} \left(\frac{1}{|X^+| - 1} d^+(x_{1\ 2}, y_{1\ 2}) + \dots \right) \\ & \quad + \frac{1-t}{|X^+|} \mathcal{M}^{M^+d^+}(X^+ - \{x_{1\ 1}, \dots, x_{1\ t}\}, Y^+ - \{y_{1\ 1}, \dots, y_{1\ t}\}) \end{aligned}$$

$$\begin{aligned}
&= \left(\frac{1}{|X^+|} \sum_{r=1}^t d^+(x_{1r}, y_{1r}) \right) + \frac{1-t}{|X^+|} \mathcal{M}_{\text{av}}^{M^+} d^+(X^+ - \{x_{11}, \dots, x_{1t}\}, Y^+ - \{y_{11}, \dots, y_{1t}\}) \\
&= \frac{t}{|\mathcal{X}|} d(x_1, y_1) + \frac{1-t}{|\mathcal{X}|} \mathcal{M}_{\text{av}} d(\mathcal{X} - (x_1, t), \mathcal{Y} - (y_1, t)) \\
&= \mathcal{M}_{\text{av}} d(\mathcal{X}, \mathcal{Y}).
\end{aligned}$$

Note, the last line of this proof should be obvious from how $\mathcal{M}_{\text{av}}^{M^+} d^+$ is expanded within the proof— $\mathcal{M}_{\text{av}} d$ can be similarly expanded.

Lemma 7.11 *If d satisfies $G\Delta I$ over M and $\mathcal{M} = (M, m)$ is a finite \mathbb{N}_1 -collection then $\mathcal{M}_{\text{av}} d$ satisfies $G\Delta I$ over $\mathcal{P}(\mathcal{M})$.*

PROOF. If $M = \{z_1, \dots, z_{|M|}\}$ let $M^+ = \{z_{11}, \dots, z_{1m(z_1)}, \dots, z_{|M|1}, \dots, z_{|M|m(z_{|M|})}\}$ where $z_1 = z_{11}, z_2 = z_{21}, \dots, z_{|M|} = z_{|M|1}$. For any $1 \leq i, j \leq |M|$, any $1 \leq r \leq m(z_i)$ and any $1 \leq r' \leq m(z_j)$, let $d^+(z_{ir}, z_{jr'}) = d(z_i, z_j)$ and $s_{d^+}(z_{ir}) = s_d(z_i)$.

As $M^+ = M \cup \{z_{12}, \dots, z_{|M|m(z_{|M|})}\}$, from lemma 7.3, d^+ satisfies $G\Delta I$ over M^+ .

From theorem 7.9 $\mathcal{M}_{\text{av}}^{M^+} d$ satisfies $G\Delta I$ over $\mathcal{P}(M^+)$.

But for every $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{M}$ there is an $X^+, Y^+ \in M^+$ so, from lemma 7.10, $\mathcal{M}_{\text{av}} d$ satisfies $G\Delta I$ over $\mathcal{P}(\mathcal{M})$.

Theorem 7.12 *If d satisfies $G\Delta I$ over a finite set M and $\mathcal{M} = (M, m)$ is a L -collection, where $L \subseteq \mathbb{Q}^{>0}$, then $\mathcal{M}_{\text{av}} d$ satisfies $G\Delta I$ over any finite subset of $\mathcal{P}(\mathcal{M})$.*

PROOF. Let $M' = \{\mathcal{X}_1, \dots, \mathcal{X}_n\}$ be a finite subset of $\mathcal{P}(\mathcal{M})$. Because M' and each \mathcal{X}_i is finite, there is a $w \in \mathbb{N}_1$ where, for each $1 \leq i \leq n$, $w\mathcal{X}_i$ is an \mathbb{N}_1 -collection. From lemma 7.8, $\mathcal{M}_{\text{av}}^{w\mathcal{M}} d$ satisfies $G\Delta I$ over $\mathcal{P}(w\mathcal{M})$. For any

$1 \leq i \leq n$, $1 \leq j \leq n$ and any $w \in \mathbb{R}^{>0}$ at all, ${}^w\mathcal{M}d(w\mathcal{X}_i, w\mathcal{X}_j) = \mathcal{M}d(\mathcal{X}_i, \mathcal{X}_j)$ and $s_{\text{av}}^w\mathcal{M}d(w\mathcal{X}_i) = s_{\text{av}}\mathcal{M}d(\mathcal{X}_i)$.

7.5.2 Other properties of $\mathcal{M}d$

Theorem 7.13 *If d is (\subseteq) -reflexive over M and $\mathcal{M} = (M, m)$, $\mathcal{M}d$ is \subseteq -reflexive over $\mathcal{P}(\mathcal{M})$.*

PROOF. If $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{M}$ then $\min_{x \in \mathcal{X}, y \in \mathcal{Y}} d(x, y) = d(x, x) = 0$ and $\mathcal{X}^- \subseteq \mathcal{Y}^- \subseteq \mathcal{M}$ as $\mathcal{X} \subseteq \mathcal{Y}$ and $m_X(x) \leq m_Y(x)$. Hence the minimum distance each recursion is 0 and $\mathcal{M}d(\mathcal{X}, \mathcal{Y}) = 0$.

Theorem 7.14 *If the distance function d is \mathcal{L}^d -strict positive over M and $\mathcal{M} = (M, m)$ then $\mathcal{M}d$ is $\mathcal{L}^{\mathcal{M}d}$ -strict positive over $\mathcal{P}(\mathcal{M})$.*

As d is non negative, the proof for this theorem follows directly from theorem 7.7.

7.6 What this Chapter Achieved

This chapter generalised the, chapter 5, definitions, theorems and proofs involving set distance functions, forming L -collection distance function definitions, theorems and proofs.

Because the family of set distance functions d_{ij}^M did not generalise well, a new family of L -collection distance function $\mathcal{M}_k d$ was defined. It was shown that $d(\mathcal{X}, \mathcal{Y}) = |\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$ has the desired properties, and $\mathcal{M}_{\text{av}} d$ and $\mathcal{M}_1 d$ have the desired properties (if d does).

L -collections can be used to help form “classification spaces” that allow the contribution proportion of authors to research papers to be recorded.

Example 7.10. $M = \{x_1, \dots, x_{10}\}$ is a set of “research paper authors”, d is a distance function over M , $\mathcal{M} = (M, m)$ is a $(0, 1]$ -collection where $m(z) = 1$ for all $z \in M$. Now each point in the space $\langle \mathcal{P}(\mathcal{M}), \mathcal{M}_k d \rangle$ is a $(0, 1]$ -collection

of research paper authors.

This space makes points available such as $\{x_1|_{\frac{1}{3}}, x_3|_{\frac{1}{6}}, x_7|_{\frac{1}{2}}\} \in \mathcal{P}(\mathcal{M})$ which can be used to “attach” information units (such as research papers).

Further examples and discussion of the use of L -collections in the definition of information space are provided in chapter 8.

Chapter 8

Information Space

8.1 Overview

This chapter discusses how information space can provide the core Knowledge Library functionality identified in section 3.5.

Section 8.3 shows how networks can be used to define basic distance functions. The concepts: **network**, **path**, **ancestor**, **descendant**, ***d*–length**, ***d*–shortest path**, **common descendant**, **common ancestor** and **networked space** are defined. This section also defines the operators: *add_node* and *make_distance_function*. These operators allow distance functions to be progressively “built-up” by adding nodes to networks.

Section 8.4 discusses **classification space**—essentially information space without any attached information elements. This section discuss how extremely large classification spaces can have very compact representations. How classification spaces can be defined that allow “uncertain” or “partial” classifications and “many levelled” classification spaces can be “built-up” is also discussed. The section also defines distance function, point, point set, and *L*-collection projections.

Section 8.5 defines a *create_space* operator to simplify the definition of spaces with points that are *L*-collections. This section also discuss how dimensions, coordinates and points can be added to, and subtracted from, spaces. The section goes on to discuss various methods of point selection.

The ordering of dimensions is also considered.

Section 8.6 discusses how information units can be attached to points in classification space. This section examine how distance functions between points, sets of points, and \mathbb{N}_1 -collections of points can be used to give distances corresponding to pairs, pairs of sets, and pairs of \mathbb{N}_1 -collections of information units. The section defines: **type one index**, **type two index**, **index**, and **attached**.

Section 8.7 defines **information space** and **information element**. This section discusses the relationship between Knowledge Libraries and information space—A Knowledge Library provides the *context of meaning* for an information space, while an information space provides the mathematical basis for a Knowledge Library.

Section 8.8 discusses the creation of information spaces and how information units and coordinates can be added and subtracted from information space. The operators *subtract_coordinate*, *add_dimension* and *subtract_dimension* are defined. The section considers various methods of information unit selection. There is a discussion on how points in information space can be selected and the operator *get_collection* is defined. Finally, it is shown how the information units attached to points in a set of points can be counted.

Section 8.9 briefly outlines what has been achieved in this chapter. The chapter discusses how networks, single and many levelled classification spaces, indexes and multidimensional spaces can form information spaces.

8.2 Introduction

The purpose of this chapter is to discuss how information space can provide the core Knowledge Library functionality identified in section 3.5.

Chapter 9 gives some “paper based examples” of the information space methods developed in this chapter that provide this functionality.

Chapter 10 discusses how this functionality can be implemented, relating the problem to existing range query algorithms and showing how these algorithms can be generalised.

Chapter 11 reports on a number of experiments that were conducted to

test the effectiveness of various range query algorithms over spaces of the sort that may be used as part of the basis for a Knowledge Library.

Before beginning the main discussion, we note an important distinction between mathematics and computer science: in mathematics, convention dictates that any addition, subtraction or other modification to an object results in a new object. In computer science—especially if the object is large and the change is small—practical considerations of time and memory complexity have resulted in a convention where the modified object is treated as the same object as the original (with modifications).

Note that computer science, unlike mathematics, deals with both “semantics” and “implementation.” When we talk about an “object” in computer science, we are more often than not talking about an object as in the “Object Orientated Programming” paradigm, that is, something that can be modified.

Apart from the names of operators, this research follows mathematical conventions when giving definitions for information space operators. Practical computer implementations however will simply modify existing objects¹.

8.3 Networked Space

Many sets have simple, natural and intuitive distance functions. Any set of numbers N , for example, suggests the distance function $d(x, y) = |x - y|$, for any $x, y \in N$. Sets with more complex elements, such as sets of “research subject areas” or “web pages” frequently do not have such obvious distance functions. In many cases, these complex elements readily form hierarchies or networks.

This section shows how suitable distance functions can be defined over such sets.

¹Rather than setting aside a new block of memory, copying the existing object into the new block, making a small modification then discarding the original object.

Network definition. A **network** (M, R, w) is a binary relation R on a set M of **nodes** and a partial function $w : M \times M \rightarrow \mathbb{R}$ where, for all $x, y \in M$, $w(x, y)$ is defined iff xRy . The **weight** $w(x, y)$ corresponds to the **edge** xRy .

Networks are also called **weighted graphs**. Graphs are variously defined as a relation on a set (as this research does for networks), a set of nodes and a set of edges, or a set of pairs. These definitions are effectively equivalent.

Note that the elements of M can be both nodes in a network *and* points in a space.

Path definition. Given a relation R , a **path** is a sequence, with $n > 1$, x_1, x_2, \dots, x_n , where $i \neq j$ implies $x_i \neq x_j$, such that $x_1Rx_2, x_2Rx_3, \dots, x_{n-1}Rx_n$. Given such a path we say that x_1 is an **ancestor** of x_n in R and that x_n is a **descendant** of x_1 in R .

d -length and d -shortest path definition. Given some distance function d , the **d -length** of a path x_1, x_2, \dots, x_n is $(\sum_{i=1}^{n-1} d(x_i, x_{i+1})) + (\sum_{j=2}^{n-1} s_d(x_j))$. In any set of paths, a **d -shortest path** is a path with the least d -length.

Note that if $xR\dots Ry$ and $yR\dots Rx$, x is both an ancestor and a descendant of y . Also note, the d -length corresponding to a d -shortest path is normally used to select the “shortest path”, from a set of paths with the same beginning and end points.

Common descendant definition. Given a relation R , y is a **common descendant** of a set X if y is a descendant of each $x \in X$ in R .

Common ancestor definition. Given a relation R , x is a **common ancestor** of a set Y if x is an ancestor of each $y \in Y$ in R .

Networked Space definition. A **networked space** $\langle M, d \rangle_R$ is a space $\langle M, d \rangle$, where M is finite, with a relation R and a distance function d such that, if $x, y \in M$:

1. if x is an ancestor of y in R , $d(x, y) =$ the d -length of a d -shortest path x, \dots, y ;
2. otherwise $d(x, y) = \max_{dx}$, where \max_{dx} is some suitable constant larger than or equal to $\max\{d(x, z) | z \in M, z \text{ is a descendant of } x \text{ in } R\}$.

Note that, from the definition, a networked space $\langle M, d \rangle_R$ is just a space $\langle M, d \rangle$ and a relation R , where (1) and (2) (in the definition) are satisfied by d and R . The definition can be used as a “test” to determine if $\langle M, d \rangle_R$ is indeed a networked space.

It is easy to see, from this definition, that a networked space $\langle M, d \rangle_R$, provided d is a distance function, satisfies $G\Delta I$ and so is a set space. Note that for simplicity, if there is no path x, \dots, y , $d(x, y) = \max_d$, where $\max_d = \max\{d(x', y') | x', y' \in M\}$, is often used. Obviously $\max_d \geq \max\{d(x, z) | z \in M, z \text{ is a descendant of } x \text{ in } R\}$ for all $x \in M$. Note that this definition of \max_d ensures that $\langle M, d \rangle_R$ satisfies $G\Delta I$.

Later in this section the operator *make_distance_function* is defined. This operator, given a network (M, R, w) and a maximum distance, defines a distance function d so that $\langle M, d \rangle_R$ is a networked space.

Network spaces can be applied to a great variety of subject matter.

Example 8.1. M is a set of “research subject areas” and R is a binary relation on M such that (M, R) is a subject hierarchy. That is, for a single $y \in M$ (the root of the hierarchy), there is no $x \in M$ where xRy . For all other $y \in M$, there is exactly one $x \in M$ where xRy . For any $x, y \in M$, $xRy \leftrightarrow y$ is an immediate sub-subject of x in the subject hierarchy.

For all $x, y \in M$, we define: $s_d(x) = 0$; $d(x, y) = 1$ if xRy ; $d(x, y) =$ the d -length of the d -shortest path x, \dots, y if $xR\dots Ry$ and $d(x, y) = \max_d$ otherwise. Now $\langle M, d \rangle_R$ is a networked space.

Example 8.2. M is a set of “web pages” and R is binary relation on M such that, for any $x, y \in M$, $xRy \leftrightarrow x$ directly links to y . For all $x, y \in M$, we define: $s_d(x) = 0$; $d(x, y) = 1$ if xRy ; $d(x, y) =$ the d -length of the d -shortest path x, \dots, y if $xR\dots Ry$ and $d(x, y) = \max_d$ otherwise. Now $\langle M, d \rangle_R$ is a networked space.

The spaces in both of the above examples satisfy ΔI . Any spans $\in \mathbb{R}^{\geq 0}$ and distances $d(x, y) \in \mathbb{R}^{\geq 0}$ can be chosen, as long as, if $xR\dots Ry$ then $d(x, y) =$ the d -length of the d -shortest path x, \dots, y , and the resulting space would still satisfy $G\Delta I$.

The operators *add_node* and *make_distance_function* are now defined. These operators allow distance functions over graphs to be “built-up”.

$(M^+, R^+, w^+) = \text{add_node}(M, R, w, z, \mathcal{X}, \mathcal{Y})$ **definition.**

If (M, R, w) is a network, $z \notin M$ and $\mathcal{X} = (X, m_X)$, $\mathcal{Y} = (Y, m_Y)$ are $\mathbb{R}^{>0}$ -collections where, $X - \{z\} \subseteq M$ and $Y \subseteq M$ then:

1. $M^+ = M \cup \{z\}$.
2. For all $x, y \in M$: xR^+z iff $x \in \mathcal{X}$; zR^+y iff $y \in \mathcal{Y}$ and xR^+y iff xRy .
3. $w^+ : M^+ \times M^+ \rightarrow \mathbb{R}$ is a partial function where $w^+(x, y) = w(x, y)$ for all $x, y \in M$ where $w(x, y)$ is defined, $w^+(x, z) = m_X(x)$ for all $x \in X$, and $w^+(z, y) = m_Y(y)$ for all $y \in Y$.

Informally, the network (M^+, R^+, w^+) is just the network (M, R, w) with the addition of node z , with weighted edges connected to parents X and children Y . For each $x \in \mathcal{X}$, *add_node* $(M, R, w, z, \mathcal{X}, \mathcal{Y})$ adds the edge xRz with the weight $m_X(x)$. Similarly, for each $y \in \mathcal{Y}$, *add_node* $(M, R, w, z, \mathcal{X}, \mathcal{Y})$ adds the edge zRy with the weight $m_Y(y)$. As $X - \{z\} \subseteq M$, edges zRz can be added.

For simplicity, in the following examples assume $s_w(x) = 0$ for all $x \in M$.

Example 8.3. If xRx , $w(x, x) = 0$ and

$$(M^+, R^+, w^+) = \text{add_node}(\{x\}, R, w, y, \{x|1, y|0\}, \emptyset)$$

then $M^+ = \{x, y\}$, xR^+x , yR^+y , xR^+y and $w^+(x, x) = w^+(y, y) = 0$, $w^+(x, y) = 1$.

Example 8.4. If xRx , yRy , xRy , $w(x, x) = w(y, y) = 0$, $w(x, y) = 1$ and

$$(M^+, R^+, w^+) = \text{add_node}(\{x, y\}, R, w, z, \{y|1, z|0\}, \emptyset)$$

then $M^+ = \{x, y, z\}$, xR^+x , yR^+y , zR^+z , xR^+y , yR^+z and $w^+(x, x) = w^+(y, y) = w^+(z, z) = 0$, $w^+(x, y) = w^+(y, z) = 1$.

$d = \text{make_distance_function}(M, R, w, \max_d)$ definition.

If (M, R, w) is a network and $\max_d \geq \max\{w(x, y) | x, y \in M, xRy\}$ then $d(x, y) =$

1. the w -length of a w -shortest path x, \dots, y (if any such path exists) and
2. \max_d (if no path x, \dots, y exists).

Clearly, from the definitions of network, network space and $\text{make_distance_function}$:

Theorem 8.1 If $d = \text{make_distance_function}(M, R, w, \max_d)$, (M, R, w) is a network and $\max_d \geq \max\{w(x, y) | x, y \in M, xRy\}$ then $\langle M, d \rangle$ is a networked space.

Example 8.5. If xRx , yRy , zRz , xRy , yRz , $w(x, x) = w(y, y) = w(z, z) = 0$, $w(x, y) = w(y, z) = 1$, $s_w(x) = s_w(y) = s_w(z) = 0$ and

$$d = \text{make_distance_function}(\{x, y, z\}, R, w, 3)$$

then:

1. $d(x, x) = d(y, y) = d(z, z) = 0$,

$$2. \ d(x, y) = d(y, z) = 1,$$

$$3. \ d(x, z) = 2$$

$$\text{and } d(y, x) = d(z, y) = d(z, x) = 3.$$

Note that examples 8.3, 8.4 and 8.5 form a sequence with the “output” from examples 8.3 and 8.4 providing the “input” for examples 8.4 and 8.5 respectively.

8.4 Classification Space

Classification space is simply a space that provides “attach points” for information units. Each point in a classification space represents a “class”—a concept grouping based on common characteristics, attributes, properties, qualities etc.

The mathematics used to compare, relate and select groups of information units will be based on comparing, relating and selecting points in classification space. In section 8.7, information space is defined as—essentially—classification space, with an “index” that “attaches” points in the space to information units.

Classification spaces can have n different “facets” or dimensions, each of which is a space itself. As was shown in section 4.4, these n spaces $\langle M_1, d_1 \rangle, \dots, \langle M_n, d_n \rangle$ can be combined to form an n -dimensional space

$$\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle.$$

Each point in $M_1 \times \dots \times M_n$ can be a “hook” to which information units can be attached.

Note that an n -dimensional space $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ has $\prod_{i=1}^n |M_i|$ unique points. This allows very compact representations of large classification spaces.

Example 8.6. A 4-dimensional classification space ($n = 4$) with only 10 coordinates in each dimension ($|M_1| = |M_2| = |M_3| = |M_4| = 10$), only

40 coordinates in total, results in a classification space with $10^4 = 10000$ unique points. A single extra coordinate, in any dimension, gives a further $10^3 = 1000$ points. An extra 10 coordinate dimension gives a further $10^5 - 10^4 = 90000$ points!

Some classification spaces require dimensions $\langle M_i, d_i \rangle$ where M_i is a set, the elements of which are themselves sets and d_i is a set distance function. If Y_i is a set and $M_i = \mathcal{P}(Y_i)$ then $|M_i| = 2^{|Y_i|}$.

This allows very compact representations of extremely large classification spaces.

Example 8.7. $|Y_1| = |Y_2| = |Y_3| = |Y_4| = 10$. For $1 \leq i \leq n$, $M_i = \mathcal{P}(Y_i)$ so $|M_i| = 2^{10} = 1024$. Now $M_1 \times M_2 \times M_3 \times M_4$ has $1024^4 = 2^{40} = 1099511627776$ unique points.

If the elements of M_i are L -collections, as they are if Y_i is an L -collection and $M_i = \mathcal{P}(Y_i)$, the classification space will often be theoretically infinite!

8.4.1 Classification spaces for uncertain and partial classifications

The classification spaces discussed so far work best when classifications are 100% certain or reliable. But they are not always appropriate for use when there may be some uncertainty involved in the classification of information units.

As already noted (see sections 6.2.5, 6.2.6, 6.4.3 and 6.4.4), L -collections can be used as part of the definition of “information systems” that provide “uncertain” classifications for information. For a Rough-set approach to uncertainty, $\{0.5, 1\}$ -collections can be used, while $(0, 1]$ -collections can be used for a Fuzzy-set approach.

In general, the approach adopted in this research is as follows. Begin with a set of points M and a set of “certainty levels” L , then an L -collection $\mathcal{M} = (M, m)$ where, for each $x \in M$, $m(x) = \max_{z \in L}$. Now $\mathcal{P}(\mathcal{M})$ (see

L -collection power set definition, section 6.3.1) is a set of L -collections. Each $x \in M$ has a membership grade $\leq m(x)$ in each L -collection in $\mathcal{P}(\mathcal{M})$ that it is a member of. All possible combinations of subsets of M and element membership grades (certainty levels), selected from L , exist in $\mathcal{P}(\mathcal{M})$. If d is a distance function over M , any ${}_k^{\mathcal{M}}d$ or $d_{ij}^{\mathcal{M}}$ can be used as a distance function over $\mathcal{P}(\mathcal{M})$.

Example 8.8. In a—hypothetical—study of field mouse behaviour in response to certain stimuli, a number of controlling physiological, psychological, and environmental factors have been identified. This suggests that mouse behaviour is determined by a combination of these factors and that mouse behaviour can be classified in a 4-dimensional space consisting of stimuli (M_1), physiology (M_2), psychology (M_3) and environment (M_4) dimensions. However—perhaps due to uncontrollable experimental variation—some variation in mouse behaviour is occurring. As a result, the best we can do is associate each observed mouse behaviour with a fuzzy-set of points in $M = M_1 \times M_2 \times M_3 \times M_4$. To do this we require a classification space such as $\langle \mathcal{P}(\mathcal{M}), {}_{\text{av}}^{\mathcal{M}}d \rangle$ where d is a distance function over M and $\mathcal{M} = (M, m)$ is a $(0, 1]$ -collection where $m(x) = 1$ for all $x \in M$.

L -collections can also be used to help form classification spaces that allow the contribution proportion of authors to research papers to be recorded.

Example 8.9. $M = \{x_1, \dots, x_{10}\}$ is a set of “research paper authors”, d is a distance function over M , $\mathcal{M} = (M, m)$ is a $(0, 1]$ -collection where $m(z) = 1$ for all $z \in M$. Now each point in the space $\langle \mathcal{P}(\mathcal{M}), {}_k^{\mathcal{M}}d \rangle$ is a $(0, 1]$ -collection of research paper authors.

This space makes points available such as $\{x_1|_{\frac{1}{3}}, x_3|_{\frac{1}{6}}, x_7|_{\frac{1}{2}}\} \in \mathcal{P}(\mathcal{M})$ which we can later use to attach information units.

Note that, following the example, there is no reason why $\langle \mathcal{P}(\mathcal{M}), {}_k^{\mathcal{M}}d \rangle$ should not be a dimension in an n -dimensional “research paper” classification space. However a 2-dimensional approach—with “research paper author” and “contribution” dimensions—to creating a classification space is usually

not ideal, as the following example illustrates.

Example 8.10. We want a distance function between groups of research paper authors that is small iff similar groups have made similar contributions. Let $\langle M_1, d_1 \rangle$ be a space of “research paper authors” and $\langle (0, 1], d_2 \rangle$ be a space of “contributions”. If $M = M_1 \times (0, 1]$ and $d = G_p^{d_1, d_2}$ then each point in the space $\langle \mathcal{P}(M), d_{ij}^M \rangle$ is a set of “research paper author, contribution” pairs. Points such as $\{(y_1, \frac{1}{3}), (y_3, \frac{1}{6}), (y_7, \frac{1}{2})\} \in \mathcal{P}(M)$ are available. However a point $\{(y_i, 0.99)\}$ will be relatively close to a point $\{(y_j, 1)\}$ —even if $i \neq j$ —just because 0.99 is very close to 1. This is *not* what we want.

Note that—although L -collections have been used in the definitions of the spaces in examples 8.9 and 8.10, the resulting classification spaces are just spaces as this research defines space—a set with a distance function over the set. It is the *elements* of the set $\mathcal{P}(\mathcal{M})$ that are L -collections.

8.4.2 Many levelled classification spaces

As noted in section 4.5, a distance function d' that gives the distance between *sets* of information units of one type may be utilised to give the distance between individual *information units* of another type. By nesting a number of spaces in this way many levelled spaces can be built-up .

A distance function d' between sets of research papers could well be used as a distance function between researchers. Then a distance function d'' between sets of researchers could be used as a distance function between research groups.

If $\langle M, d \rangle$ is a “research paper” classification space, $\langle M', d' \rangle$, where $M' \subseteq \mathcal{P}(M)$, could be a “researcher” classification space and $\langle M'', d'' \rangle$, where $M'' \subseteq \mathcal{P}(M')$, a “research group” classification space .

Now we can have a point $x \in M$ in the research paper classification space, a set $x' \subseteq M$ which is a point in the researcher classification space and a set (of sets) $x'' \subseteq \mathcal{P}(M)$ which is a point in the research group classification space.

If $d' = d_{\text{av } 0}^M$ or $d' = d_{100 \ 0}^M$, and

1. $\langle M, d \rangle$ satisfies $G\Delta I$ then $\langle M', d' \rangle$ satisfies $G\Delta I$ (theorem 5.9);
2. $\langle M, d \rangle$ is \subseteq -recursive then $\langle M', d' \rangle$ is \subseteq -recursive (theorem 5.10);
3. $\langle M, d \rangle$ is \mathcal{Z}^d -strict positive then $\langle M', d' \rangle$ is $\mathcal{Z}^{d'}$ -strict positive (theorem 5.11).

Any number of levels (with appropriate properties) can be built-up “on top of one another” in this way.

More generally, if $d' = \overset{\mathcal{M}}{1}d$ or $d' = \overset{\mathcal{M}}{\text{av}}d$, M is a finite set, $\mathcal{M} = (M, m)$ is an L -collection, where $L \subseteq \mathbb{Q}^{>0}$, and

1. $\langle M, d \rangle$ satisfies $G\Delta I$ then $\langle \mathcal{P}(\mathcal{M}), d' \rangle$ satisfies $G\Delta I$ (theorems 7.5, 7.12);
2. $\langle M, d \rangle$ is \subseteq -recursive then $\langle \mathcal{P}(\mathcal{M}), d' \rangle$ is \subseteq -recursive (theorems 7.6, 7.13);
3. $\langle M, d \rangle$ is \mathcal{Z}^d -strict positive then $\langle \mathcal{P}(\mathcal{M}), d' \rangle$ is $\mathcal{Z}^{d'}$ -strict positive (theorems 7.7, 7.14).

Again, any number of levels (with appropriate properties) can be built-up in this way.

8.4.3 Projected classification spaces

When organising information, it pays to be as thorough as possible so that all relations between the information units that may be of interest are reflected in whatever classification scheme is used. Generally however, individual users of classification schemes are interested in only a small fraction of these relations.

Although an n -dimensional classification space could be used to realise such a classification scheme, users should not be expected to be interested in all n dimensions every time they use the space. If all n dimensions are not of interest, a dimensionally reduced classification space is required.

Formally, an ordered list of dimension indices $T = t_1 \dots t_k$ where $k \leq n$ is used to “select” dimensions of interest. So, for $1 \leq i \leq n$ and $1 \leq j \leq n$ we have $1 \leq t_i \leq n$ and $t_i < t_j$ if $i < j$.

Definition of d^- . Given an ordered list of dimension indices $T = t_1 \dots t_k$, if $d = G_p^{d_1 \dots d_n}$, $d^- = G_p^{d_{t_1} \dots d_{t_k}}$.

Now, an n -dimensional distance function $G_p^{d_1 \dots d_n}$ can be used to define a k -dimensional distance function d^- .

It is also possible to project n -tuples.

Definition of x^- . Given an ordered list of dimension indices $T = t_1 \dots t_k$, if $x = (x_1, \dots, x_n)$, $x^- = (x_{t_1}, \dots, x_{t_k})$.

We will call $d^-(x^-, y^-)$ the **projected distance** between points x and y .

Now a set of n -tuples M can be used to define a set of k -tuples M^- .

Definition of M^- . Given an ordered list of dimension indices T , if M is a set of n -tuples, $M^- = \{x^- | x \in M\}$.

Now if d is a distance function over M , d^- is a distance function over M^- . Also if $x = (x_1, \dots, x_n) \in M$, $y = (y_1, \dots, y_n) \in M$ and $T = t_1$ then $d_{t_1}(x_{t_1}, y_{t_1}) = d^-(x^-, y^-)$.

As L -collections can be used to indicate how many information units are attached to each point (see section 8.6), or to handle uncertain classifications (see section 8.4.1), L -collection projections are also useful.

Definition of \mathcal{X}^- . Given an ordered list of dimension indices T , if $\mathcal{X} = (X, m_X)$ is an L -collection of n -tuples, $\mathcal{X}^- = (X^-, m_{X^-})$ is an L -collection where, for each $x^- \in X^-$,

$$m_{X^-}(x^-) = \sum_{\{x | x^- \text{ is a projection of } x\}} m_X(x).$$

Note that there may be a number of points $x \in X$ for each projected point x^- . So $m_{X^-}(x^-)$ is the sum of all these $m_X(x)$ where x^- is a projection of x . See section 4.4 for the definition of “projection.”

8.5 Working with Classification Space

8.5.1 Creation

As discussed in section 4.2.1, a space is just a pair $\langle M, d \rangle$ where d is a distance function over M . So there is no need to define an operator to create simple classification spaces. Even moderately complex spaces such as $\langle \mathcal{P}(M), d_{ij}^M \rangle$, $\langle \mathcal{P}(M), {}^M_k d \rangle$ and $\langle M_1 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ are similarly straight-forward to define.

Spaces with points that are L -collections are slightly more troublesome. To assist, the following operator is defined.

$\langle \mathcal{P}(\mathcal{M}), {}^M_k d \rangle = \text{create_space}(L, M, d, k)$ **definition.**

If $\langle M, d \rangle$ is a space and $k \in (0, 1]$ or $k = av$ then $\mathcal{M} = (M, m)$ is an L -collection where $m(x) = \max_{y \in L} d(x, y)$ for all $x \in M$.

Note that the space $\langle \mathcal{P}(\mathcal{M}), {}^M_k d \rangle$ defined by this operator has points $\mathcal{X} = (X, m_X) \in \mathcal{P}(\mathcal{M})$ where $X \subseteq M$ and, for each $x \in X$, $m_X(x) \in L$.

8.5.2 Addition and subtraction

Dimensions can be added and subtracted to an n -dimensional space simply by creating a new space. So dimension addition and subtraction operators are not required.

Example 8.11. We have a space $A = \langle M_1 \times M_2 \times M_3, G_2^{d_1 d_2 d_3} \rangle$ to which we want to add a further dimension $\langle M_4, d_4 \rangle$. But we can do this simply by defining a space $B = \langle M_1 \times M_2 \times M_3 \times M_4, G_2^{d_1 d_2 d_3 d_4} \rangle$.

Note that A is a projection (see definition section 4, chapter 4) of B .

Points can be added to, and subtracted from, an n -dimensional space simply by adding and subtracting coordinates to and from its dimensions. Section 8.3 discusses how points can be added to networked spaces. It is trivial to add and subtract points to a (non networked) space by creating a new space.

Example 8.12. We want to add a point “11” to the space $\langle M_1, d_1 \rangle$ where $M_1 = \{1, 2, \dots, 10\}$ and $d_1(x, y) = |x - y|$ for all $x, y \in M_1$. To do this we create a new space $\langle M_1^+, d_1^+ \rangle$ where $M_1^+ = M_1 \cup \{11\}$ and $d_1^+(x, y) = |x - y|$ for all $x, y \in M_1^+$.

Note that, in the example, if $\langle M_1, d_1 \rangle$ were a dimension of an n -dimensional space $\langle M_1 \times M_2 \times \dots \times M_n, G_p^{d_1 \dots d_n} \rangle$ then $\langle M_1^+ \times M_2 \times \dots \times M_n, G_p^{d_1^+ \dots d_n} \rangle$ makes an additional set of points available.

As is suggested in the introduction to this chapter, practical computer implementations would modify the space $\langle M_1, d_1 \rangle$, rather than creating a new space $\langle M_1^+, d_1^+ \rangle$. Mathematically however, M_1 and M_1^+ , and d_1 and d_1^+ , are distinct objects.

8.5.3 Point selection

Most simply, information space users may wish to “select” all the points in a space $\langle M, d \rangle$ that are “close” to a point. Given a point $x \in M$ and a limit r ,

$$Y = \{y' \in M | d(x, y') \leq r\}$$

is the set of all points $y' \in M$ where the distance $d(x, y)$ between x and y is less than or equal to r . Note that, in Euclidean and metric geometry, Y is a **ball** with radius r and centre x . Similarly, given a point $y \in M$ and a limit r ,

$$X = \{x' \in M | d(x', y) \leq r\}$$

is the set of all points $x' \in M$ where the distance $d(x, y)$ between x and y is less than or equal to r . Note that, as this research is not restricted to Euclidean and metric geometry, it may be that, even if $x = y$, $X \neq Y$ as the distance function d may not be symmetrical.

Each point selection method discussed in this section has a similar “mirror” to this—a set of points X can be obtained using a distance function and a specified point y —just as a set of points Y can be obtained using a distance function and a specified point x . Given the definition of the set Y ,

it is trivial to define the set X , so this is omitted for the sake of brevity.

Given an n -dimensional classification space $\langle M, d \rangle$, where $M = M_1 \times \dots \times M_n$ and $d = G_p^{d_1 \dots d_n}$, users may wish to select a set of points in M , based on the distances in only those dimensions they select.

Users might also want to select points that are close to a set of points. Given a set of points X and a limit r

$$\{y \in M | d_{ij}^M(X, \{y\}) \leq r\}$$

is the set of all points $y \in M$ where the distance $d_{ij}^M(X, \{y\})$ between X and y is less than or equal to r .

If $\langle M, d \rangle$ is a networked space then $\{y \in M | d_{100j}^M(X, \{y\}) \leq r\}$ can be used to select a set of *common descendants* of the set X .

Similarly $\{x \in M | d_{i100}^M(\{x\}, Y) \leq r\}$ can be used to select a set of *common ancestors* of the set Y . Of course this point selection method can also be used if $\langle M, d \rangle$ is not a networked space.

Example 8.13. $\langle M, d \rangle_R$ is a networked space where $M = \{1, 2, 3, 4, 5\}$, $1R2, 2R3, 3R4, 4R5$, $d(1, 2) = d(2, 3) = d(3, 4) = d(4, 5) = 1$ and if x is not an ancestor of y , $d(x, y) = 5$. The set of descendants of 2 is $\{y \in M | d_{100j}^M(\{2\}, \{y\}) \leq 4\} = \{3, 4, 5\}$.

There are many other methods that may be used to select sets of points in classification space.

Example 8.14. $\langle M, d \rangle$ where $M = M_1 \times \dots \times M_n$ and $d = G_p^{d_1 \dots d_n}$ is an n -dimensional classification space where, for $1 \leq i \leq n$, each M_i is a set of sets.

The *angle* between two points $y = (y_1, \dots, y_n)$ and $c = (c_1, \dots, c_n)$ in M , with respect to an origin $x = (x_1, \dots, x_n)$, can be defined as

$$\text{angle}(y, c, x) = \arccos \left(\frac{y \cdot c}{d(x, y)d(x, c)} \right)$$

where $y \cdot c = \sum_{i=1}^n d_i(x_i, y_i)d_i(x_i, c_i)$.

This definition of angle can be used to select *cones* of points. Given a limit $r \in \mathbb{R}$, a center $c \in M$, an apex $x \in M$ and an angle θ the corresponding cone is

$$\{y \in M | G_p^{d_1 \dots d_n}(x, y) \leq r, |\text{angle}(y, c, x)| \leq \theta\}.$$

Sometimes, rather than just selecting a set of points, users may be interested in the distance between the specified point and each point they select.

Example 8.15. Given a point $x \in M$ and a limit r , rather than the set $\{y \in M | d(x, y) \leq r\}$, we may be interested in the set

$$\{(y, d(x, y)) | y \in M, d(x, y) \leq r\}.$$

Similar (point, distance) pair sets can be defined for each of the point selection methods discussed.

If $\langle M, d \rangle$ is an n -dimensional space projected points can be used to select sets of points. Given a projected point $x^- \in M^-$,

$$\{x \in M | x^- \text{ is a projection of } x\}$$

Similar “projected” equivalents of each selection method discussed are possible.

Example 8.16. Given a projected point $x^- \in M^-$ and a limit r

$$\{y \in M | d^-(x^-, y^-) \leq r\}$$

is the set of all points $y \in M$ where the projected distance $d^-(x^-, y^-)$ between x and y is less than or equal to r .

Note, in the above example, d^- , x^- and y^- are defined with respect to the implied ordered list of dimension indices $T = t_1 \dots t_k$.

8.5.4 Ordering

The points in classification spaces, and the coordinates in classification space dimensions, may not be ordered. This is necessary as classification space needs to be general enough to classify any kind of information. However it may be useful to define an ordering for the coordinates or points of a classification space.

If $\langle M, d \rangle$ is a space and $z \in M$ an ordering relation R (dependent on z) can be defined where, for all $x, y \in M$,

$$xRy \text{ iff } d(z, x) \leq d(z, y).$$

Similarly, an ordering relation R (dependent on z) can be defined where, for all $x, y \in M$,

$$xRy \text{ iff } d(x, z) \leq d(y, z).$$

Example 8.17. $\langle M, d \rangle$ where $M = M_1 \times \dots \times M_n$ and $d = G_p^{d_1 \dots d_n}$ is an n -dimensional classification space where, for $1 \leq i \leq n$, each M_i is a set of sets. As $z = (\emptyset, \dots, \emptyset) \in M$ is a natural *origin* we can define an ordering relation $x_i R_i y_i$ iff $d_i(\emptyset, x_i) \leq d_i(\emptyset, y_i)$ for $x_i, y_i \in M_i$, for each dimension M_i .

Ordering dimensions can be useful for displaying classification and information spaces. For example, if the coordinates in each of the n dimensions of a classification space are ordered, then pairs of dimensions can form the axes for scatter graphs that plot points with attached information units.

8.6 Attaching Information Units to Points in Classification Spaces

Given a classification space $\langle M, d \rangle$, we may have a set of information units N that we want to attach to points in the space. This can be done using a set I of pairs (u, x) where $u \in N$ and $x \in M$.

Now if $M^I = \{x \in M \mid (u, x) \in I \text{ for some } u \in N\}$ —so M^I is the subset of M with attached information units—then the \mathbb{N}_1 -collection (M^I, m_I) can give

the number of information units attached to each point in the classification space $\langle M, d \rangle$. For each point $x \in M^I$, $m_I(x)$ is the number of information units attached to x .

If M is a set of sets then the point $\emptyset \in M$ can indicate “unclassified” or “classification unknown”.

If $M = M_1 \times \dots \times M_n$ where for $1 \leq i \leq n$, M_i is a set of sets, then \emptyset in dimension i can indicate “unclassified in dimension i ”. The point $(\emptyset, \dots, \emptyset) \in M$ can be used to indicate “unclassified” or “classification unknown”. Note that, for each $x \in M^I$, $m_I(x) \geq 1$ as M^I is the set of points in $M_1 \times \dots \times M_n$ that have at least one information unit attached.

8.6.1 Distance

The distance function $d = G_p^{d_1 \dots d_n}$ can be used to give the distance between any pair of points $x, y \in M = M_1 \times \dots \times M_n$. It will be seen that both x and y can be points that have information units attached. So, by giving the distance between points, d can be used to give the distance between information units.

Similarly, x can be any point in M and y can be a point in $M^I \subseteq M$, so d can be used to “retrieve” all information units y where the distance between x and y is less than or equal to r .

Also $d_{ij}^M(X, Y)$ and ${}_k^M d(X, Y)$, where $X, Y \subseteq M^I$, can be used to give distances between sets of information units. If more than one information unit can be attached to the same point, and it is important that this does not distort the distance, it is necessary to use $|\mathcal{X}| - |\mathcal{X} \cap \mathcal{Y}|$, $d_{ij}^{\mathcal{M}}(\mathcal{X}, \mathcal{Y})$ or ${}_k^{\mathcal{M}} d(\mathcal{X}, \mathcal{Y})$ to give distances between \mathbb{N}_1 -collections of points (with attached information units).

Example 8.18. The \mathbb{N}_1 -collection $\mathcal{M} = (M^I, m) = \{x_1|1, x_2|1, x_3|98\}$, where $M^I \subseteq M$, gives the number $m(x)$ of information units attached to each point $x \in M^I$ (if $x \in M - M^I$ then x has no attached information units). There is a distance function d over M where $d(x_1, x_2) = 1$, $d(x_2, x_2) = 0$ and $d(x_3, x_2) = 100$. Now $d_{50j}^M(\{x_1, x_2, x_3\}, \{x_2\}) = d(x_1, x_2) = 1$ whereas $d_{50j}^{\mathcal{M}}(\{x_1|1, x_2|1, x_3|98\}, \{x_2|1\}) = d(x_3, x_2) = 100$.

Note that d_{ij}^M and ${}_k^M d$ are distance functions over $\mathcal{P}(M)$ while $d_{ij}^{\mathcal{M}}$ and ${}_k^{\mathcal{M}} d$ are distance functions over $\mathcal{P}(\mathcal{M})$.

8.6.2 Indexing classification spaces

There are a number of ways to define indexes that attach information units to points. In section 8.7, information space will be defined by combining the definitions of index and classification space.

Type one index definition. *Given a set M of classes and a set N of information units, a **type one index I of N over M** is a set of pairs where, for each $u \in N$, there is at least one pair $(u, x) \in I$ where $x \in M$. If $(u, x) \in I$, u is **attached** to x .*

A limitation of type one indexes is that—because I is a *set*—given a number of identical information units u , only allow a single such information unit can be attached to the same point x . This is not a problem for many types of information space.

Example 8.19. Because we only require a single copy of each research paper, we normally do not want to attach more than one identical paper to a single point in an information space of research papers.

For other types of information space this is a serious shortcoming.

Example 8.20. In an information space of (anonymous) questionnaires, a number of respondents may return identical questionnaires. To avoid seriously misrepresenting survey results, questionnaire information spaces must reflect this.

For a questionnaire with n questions, an n -dimensional classification space $\langle M, d \rangle$ where $M = M_1 \times \dots \times M_n$ can be prepared. For $1 \leq i \leq n$, the set of coordinates M_i in dimension i reflects the different answers to question

i. Now a completed questionnaire is just a point $x \in M_1 \times \dots \times M_n$. For classification spaces such as these, another type of index is required.

Type two index definition. *Given a set M of classes and a set $M^I \subseteq M$ of information units, a **type two index** \mathcal{I} over M is an \mathbb{N}_1 -collection (M^I, m_I) . If $x \in \mathcal{I}$, $m_I(x)$ is the number of identical information units x **attached** to M .*

A limitation of type two indexes is that they can only cope with information units that are identical to points in M .

In general, indexes are required that can cope with both

1. information units that are distinct from points, and
2. more than one identical information unit attached to the same point.

Index definition. *Given a set M of classes and a set N of information units, an **index** $\mathcal{I} = (J, m)$ of N over M is an \mathbb{N}_1 -collection of pairs where, for each $u \in N$, there is one pair $(u, x) \in J$ where $x \in M$. If $(u, x) \in J$, $m((u, x))$ is the number of copies of the information unit u **attached** to x .*

For each type one index I_1 there is a corresponding index \mathcal{I} where,

$$(u, x) \in I_1 \text{ iff } (u, x) \in I \text{ and } m((u, x)) = 1.$$

Similarly, for each type two index $\mathcal{I}_2 = (M^I, m_I)$ there is a corresponding index $\mathcal{I} = (J, m)$ where,

$$x \in \mathcal{I}_2 \text{ iff } (x, x) \in \mathcal{I} \text{ and } m((x, x)) = m_I(x).$$

So the definition of index generalises both type one and type two index definitions.

Importantly, an index is an \mathbb{N}_1 -collection, rather than a set, in order to cope with a number of identical information units attached to the same point. So, for an index (J, m) , $m((u, x))$ is the number of identical information units u attached to the point x . Indexes are *not* used to handle uncertain or partial classifications (see section 8.4.1).

In this research, indexes are used to attach information units to points in M , where $\langle M, d \rangle$ is a classification space.

8.7 Information Space

Information Space definition. *An information space is a triple (M, d, \mathcal{I}) where $\langle M, d \rangle$ is a classification space and \mathcal{I} is an index of N over M , for some set N . The elements of N are **information elements** of the information space.*

If (M, d, \mathcal{I}) is an information space, $\langle M, d \rangle$ is the **corresponding classification space**.

Note that, as $\mathcal{I} = (J, m)$ is an \mathbb{N}_1 -collection of pairs, where there is a pair (u, x) for each $u \in N$, the set N of information units is implicit. Also note that, after they have been attached to an information space, information units are also called information elements of the information space.

Given an information space (M, d, \mathcal{I}) :

1. $N = \{u | (u, x) \in \mathcal{I}\}$
2. $M^I = \{x | (u, x) \in \mathcal{I}\}$ and
3. $m_I : M^I \rightarrow \mathbb{N}_1$ where, for each $x \in M^I$, $m_I(x) = \sum_{(u, x) \in \mathcal{I}} m((u, x))$
(so $m_I(x)$ is the number of—not necessarily distinct—information units attached to x).

In this way, the \mathbb{N}_1 -collection (M^I, m_I) as discussed in section 8.6, is implied from the definition of (M, d, \mathcal{I}) .

If an information space provides the mathematical basis for a Knowledge Library, that Knowledge Library provides the *context of meaning* for the information space.

Example 8.21. The information space $(M_1 \times M_2 \times M_3, G_p^{d_1 d_2 d_3}, \mathcal{I})$ provides the mathematical basis for a research paper Knowledge Library. Stripped of its context of meaning M_1 is just a set of coordinates in the space $\langle M_1 \times$

$M_2 \times M_3, G_p^{d_1 d_2 d_3}$. In the context of meaning provided by the Knowledge Library, M_1 is a set of paper titles, $x \in M_1$ is the title “A Relational Model of Data for Large Shared Data Banks”, $x' \in M_1$ is the title “Comparing Images Using the Hausdorff distance”, etc.

Note that there could be a research paper Knowledge Library with an information space $(M_2 \times M_3 \times M_1, G_p^{d_2 d_3 d_1}, I)$ that is effectively the same as the Knowledge Library in the example.

8.8 Working with Information Space

Information space operators that correspond to core Knowledge Library functionality, as identified in section 3.5 and used in examples in chapter 9 are now defined.

8.8.1 Creation

Note that an index of an n -dimensional information space is just an L -collection of information unit, n -tuple pairs. The operators needed to build L -collections, sets, n -tuples and hence indexes already exist. The existence of information units is assumed. Similarly, as information space is just a classification space, paired with an index, the operators needed to build information spaces already exist.

8.8.2 Index Manipulation

An information unit u can be attached to an information space $\langle M, d, \mathcal{I} \rangle$ at a point $x \in M$, simply by creating a new index $\mathcal{I}^* = \mathcal{I} \cup \{(u, x)\}$ for the classification space. To detach a single copy of an information unit u from a point $x \in M$, simply replace the existing index \mathcal{I} with the index $\mathcal{I}^* = \mathcal{I} - ((u, x), 1)$.

No change needs to be made to the index when adding coordinates. When subtracting coordinates, care must be taken to detach information units.

If (M, d, \mathcal{I}) is a 1-dimensional information space, an operator that detaches information units from the subtracted coordinate/point x and adds them to a “alternate” coordinate/point y is required.

$\mathcal{I}^* = \text{move_unit}(\mathcal{I}, x, y)$ definition.

If $\mathcal{I} = (J, m)$, $\mathcal{I}^* = (J^*, m^*)$ where

$J^* = J \cup \{(u, y) | (u, x) \in J\} - \{(u, x) | (u, x) \in J\}$ and

1. $m^*((u, x')) = m((u, x'))$ for any $(u, x') \in J$ where $x' \neq x$ and $x' \neq y$,
2. $m^*((u, y)) = m((u, y)) + m((u, x))$ where $(u, y), (u, x) \in J$, and
3. $m^*((u, y)) = m((u, x))$ where $(u, x) \in J$ and $(u, y) \notin J$.

Example 8.22. We have a 1-dimensional information space (M, d, \mathcal{I}) from which we want to subtract the coordinate/point x . We want each information unit attached to x to be attached to y . If $\mathcal{I}^* = \text{subtract_coordinate}(\mathcal{I}, x, y)$ then $(M - x, d, \mathcal{I}^*)$ is the information space we want.

For n -dimensional information spaces (M, d, \mathcal{I}) , an operator that detaches information units from any point $(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \in M$ and attaches them to the corresponding point $(x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n) \in M$ is required. This is more complicated to write but follows the same pattern.

$\mathcal{I}^* = \text{move_unit}(\mathcal{I}, x_i, y_i)$ **definition.**

If $\mathcal{I} = (J, m)$, $\mathcal{I}^* = (J^*, m^*)$ where

$J^* = J \cup \{(u, (x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n)) \mid (u, (x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)) \in J\}$
 $- \{(u, (x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n)) \in J\}$ and

1. $m^*((u, (x_1, \dots, x_i, \dots, x_n))) = m((u, (x_1, \dots, x_i, \dots, x_n)))$
 for any $(u, (x_1, \dots, x_i, \dots, x_n)) \in J$ where $x_i \neq x_i$ and $x_i \neq y_i$,
2. $m^*((u, (x_1, \dots, y_i, \dots, x_n))) = m((u, (x_1, \dots, x_i, \dots, x_n))) + m((u, (x_1, \dots, y_i, \dots, x_n)))$
 for any $(u, (x_1, \dots, x_i, \dots, x_n)) \in J$ where $(u, (x_1, \dots, y_i, \dots, x_n)) \in J$, and
3. $m^*((u, (x_1, \dots, y_i, \dots, x_n))) = m((u, (x_1, \dots, x_i, \dots, x_n)))$
 for any $(u, (x_1, \dots, x_i, \dots, x_n)) \in J$ where $(u, (x_1, \dots, y_i, \dots, x_n)) \notin J$.

Note that both versions of the operator give the same result when (M, d, \mathcal{I}) is a 1-dimensional space.

Adding (and subtracting) dimensions to information spaces is complicated by the need to change points in \mathcal{I} from n -tuples to $n+1$ -tuples ($n-1$ -tuples). When adding dimensions, each information unit should be attached to a suitable information unit attribute that is also a coordinate in the new dimension².

$\mathcal{I}^+ = \text{index_add_dimension}(\mathcal{I}, M_{n+1})$ **definition.**

If $\mathcal{I} = (J, m)$, $\mathcal{I}^+ = (J^+, m^+)$ where

$J^+ = \{(u, (x_1, \dots, x_n, x_{n+1})) \mid (u, (x_1, \dots, x_n)) \in J \text{ and } x_{n+1} \in M_{n+1} \text{ is some suitable attribute of } u\}$ and

$m^+((u, (x_1, \dots, x_n, x_{n+1}))) = m((u, (x_1, \dots, x_n)))$.

Subtracting dimensions is not complicated.

²If no such attribute exists, the information unit is attached to a “default” coordinate, perhaps \emptyset .

$\mathcal{I}^- = \text{index_subtract_dimension}(\mathcal{I}, M_i)$ **definition.**

If $\mathcal{I} = (J, m)$, $\mathcal{I}^- = (J^-, m^-)$ where

$J^- = \{(u, (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)) \mid (u, (x_1, \dots, x_n)) \in J\}$ and

$m^-((u, (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n))) = m((u, (x_1, \dots, x_n)))$ for the $x_i \in M_i$ where $(u, (x_1, \dots, x_n)) \in J$.

8.8.3 Information unit selection

The simplest information unit selection method is to select all information units attached to a point in space. That is, given a point x and an index \mathcal{I} ,

$$\{u \mid (u, x) \in \mathcal{I}\}$$

is the set of all information units attached to x .

Users often want to select all information units attached to a set of points. Given a set of points X and an index \mathcal{I} ,

$$\{u \mid (\exists x \in X)(u, x) \in \mathcal{I}\}$$

is the set of all information units attached to some point in X . The number of distinct information units attached to X is the cardinality of this set, while the number of—not necessarily distinct—information units attached to X is

$$\sum_{\{(u,x) \in \mathcal{I} \mid x \in X\}} m((u, x)).$$

Users may want to select all information units attached to a point that is “close” to a specified point. Given a point x , a limit r , a space $\langle M, d \rangle$ and an index \mathcal{I} ,

$$\{u \mid (\exists y \in M)(u, y) \in \mathcal{I}, d(x, y) \leq r\}$$

is the set of all all information units attached to points within distance r of x .

Just as for point selection (see section 8.5.3) this can be mirrored—given a point y and r, M, d, \mathcal{I} as above, $\{u \mid (\exists x \in M)(u, x) \in \mathcal{I}, d(x, y) \leq r\}$. Again,

this is possible for many selection methods, but quite trivial, so it is omitted.

Rather than selecting *all* the information elements within a fixed radius of a point, users may want to select the first k information elements. That is, given a point $x \in M$ and an index $\mathcal{I} = (J, m)$ over M , users may want to select a minimal set $Y \subseteq M^I$ where, for all $y \in M^I$ and $y' \in M^I - Y$, $d(x, y) \leq d(x, y')$ and $\sum_{\{x \in Y\}} m(x) \geq k$.

Users may also want to select all information units attached to a point that is “close” to a set of points. Given a set of points X , a limit r , a space $\langle M, d \rangle$ and an index \mathcal{I} , this is

$$\{u | (\exists y \in M)(u, y) \in \mathcal{I}, {}^M_k d(X, \{y\}) \leq r\}.$$

Note that, not only can this be mirrored (use Y and ${}^M_k d(\{x\}, Y)$ to find u where $(u, x) \in \mathcal{I}$), it is possible to replace ${}^M_k d$ with d_{ij}^M in both the mirrored and non-mirrored versions.

Given an n -dimensional classification space users may want to select all information units that are attached to, or are close to a point in a projected space—so only dimensions that users are interested in count towards selecting information units.

Given a point x^- in a projected space and an index \mathcal{I} ,

$$\{u | (\exists x \in M) \ x^- \text{ is a projection of } x, (u, x) \in \mathcal{I}\}$$

is the set of all information units attached to x^- .

There are similar projected versions for all the information unit selection methods that have been discussed

Example 8.23. Given a point x^- in a projected space, a limit r , an n -dimensional distance function d and an index I ,

$$\{u | (\exists y \in M) y^- \text{ is a projection of } y, (u, y) \in \mathcal{I}, d^-(x^-, y^-) \leq r\}$$

is the set of all information units attached to a point y where $d^-(x^-, y^-)$ is no greater than r .

Note, in the above example, d^- , x^- and y^- are defined with respect to the implied ordered list of dimension indices $T = t_1 \dots t_k$.

8.8.4 Selecting points in information space

Points in information space are also points in classification space and can, of course, be selected in the same way (see section 8.5.3).

Users often want to select information units that are close to other information units. As distance functions give distances between points, the attached points must first be obtained.

Given an information unit u , and an index \mathcal{I} , u is attached to is the point x where $(u, x) \in \mathcal{I}$. Given a set of information units N^* and an index \mathcal{I} , the set of points that have information units in N^* attached is the set

$$\{x | (\exists u \in N^*)(u, x) \in \mathcal{I}\}.$$

Having obtained the attached points in this way, information units can be selected as is discussed in section 8.5.3.

Sometimes the number of information units—identical or not—attached to each point is important.

Example 8.24. Given \mathbb{N}_1 -collections $\mathcal{N}_1 = (N_1, m_1)$ and $\mathcal{N}_2 = (N_2, m_2)$ of information units where (for example) $m_1(u)$ is the number of identical information units u in \mathcal{N}_1 , we want the distance ${}^M_k d(\mathcal{X}, \mathcal{Y})$ between \mathbb{N}_1 -collections of points $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$ where $m_X(x)$ is the number of—not necessarily distinct—information units in \mathcal{N}_1 attached to the point $x \in X$ and $m_Y(y)$ is the number of information units in \mathcal{N}_2 attached to the point $y \in Y$.

Hence an operator is required that, given an \mathbb{N}_1 -collection \mathcal{N} of information units and an index \mathcal{I} , defines an \mathbb{N}_1 -collection of points with the number of information units attached to each point.

$\mathcal{X} = \text{get_collection}(\mathcal{N}, \mathcal{I})$ **definition.**

If $\mathcal{I} = (J, m)$ is an index and $\mathcal{N} = (N^*, m_N)$ is an \mathbb{N}_1 -collection of information units then $\mathcal{X} = (X, m_X)$ where

1. $X = \{x \mid (\exists u \in N^*)(u, x) \in \mathcal{I}\}$ and

2. $m_X(x) = \sum_{u \in N^*, (u, x) \in \mathcal{I}} m_N(u).$

So, for each point $x \in X$, $m_X(x)$ is the total number of, not necessarily distinct, information units in \mathcal{N} attached to x .

Users may want the distance between two \mathbb{N}_1 -collections of information units, where each \mathbb{N}_1 -collection is specified by a set of points. If each information unit in an \mathbb{N}_1 -collection is attached to a point in X , rather than using X to define an \mathbb{N}_1 -collection of information units which, in turn, is used to define an \mathbb{N}_1 -collection of points, the \mathbb{N}_1 -collection of points is defined directly from X .

If $\mathcal{I} = (J, m)$ is an index and X is a set of points and, for each $x \in X$,

$$m_X(x) = \sum_{(u, x) \in J} m((u, x))$$

then $m_X(x)$ is the total number of, not necessarily distinct, information units attached to the point $x \in X$. The \mathbb{N}_1 -collection $\mathcal{X} = (X, m_X)$ can be used in an \mathbb{N}_1 -collection distance function.

8.9 What this Chapter Achieved

This chapter showed how networks can be used to create spaces with the desired properties. Examples 8.1 and 8.2 demonstrated the utility of this. The chapter showed how multidimensional spaces can be classification spaces, which provide attach points for information units. Classification space is later used as part of the definition of information space.

The chapter also showed how classification spaces for uncertain or partial classifications can be formed. Examples 8.8 and 8.9 demonstrated the utility

of this. It was also shown how many levelled classification spaces can be formed. This is useful for spaces, such as that discussed in example 4.12.

The projections of distance functions, distances, n -tuples, sets of n -tuples, and L -collections of n -tuples were defined. These definitions will be used to enable information space users to query projections of an n -dimensional information space, when all n dimensions are not of interest.

The chapter discussed how classification spaces can be created, added to and subtracted from. This will be used for the creation and maintenance of Knowledge Libraries. Various methods of point selection were also discussed. This will be used for the forming of queries. Index was defined and used, along with classification space, to define information space. Various operators were defined for manipulating indexes. The selection of information units and points with attached information units was also discussed. Chapter 9 will show how information space can provide core Knowledge Library functionality.

Chapter 9

Basing Knowledge Libraries on Information Space

9.1 Overview

This chapter presents two paper based examples that outline how information space can provide the mathematical basis for two important types of Knowledge Library. It is left to the reader to construct further examples to satisfy themselves of the utility and general applicability of information space.

Section 9.2 discusses how information space can be used as the mathematical basis for questionnaire Knowledge Libraries. This section shows how such information spaces can be queried to reveal points attached to questionnaires, how N_1 -collections of questionnaires can be compared, how to count the number of questionnaires attached to, or “close” to, a point, and how dimensions can be ordered to allow the space to be “displayed”. The section gives an example profession/salary/hours worked questionnaire and shows how a distance function can be used to answer the question “how much more, on average, do doctors earn than nurses?” It goes on to show how a further “pay rate” dimension can be used to compare the pay rate of doctors and nurses.

Section 9.3 discusses how information space can be used as the mathe-

mathematical basis for research paper Knowledge Libraries. This section shows how a suitable information space can be defined and illustrates a number of methods of selecting and comparing research papers. The section goes on to discuss how the dimensions can be “extended” to provide even greater functionality—how “subject” and “title” dimensions can be combined to disambiguate keywords in the title, how an “author” dimension can be extended to record the contribution percentage of each author of a paper, and how “subject” dimensions based on subject hierarchies can be upgraded into “concept” dimensions based on more finely grained concept hierarchies.

9.2 Information Space for Questionnaire Knowledge Libraries

Given a questionnaire or exam we would like to construct a Knowledge Library to store, analyse and present the results. The questionnaire consists of n questions. Questions can have a number of different standard answers (for example tick one of five boxes). Questions may also have short written responses (for example age, name, address, short comment).

Use n different spaces, one space for each question, as the basis for this type of Knowledge Library. Each space has a point for each different answer. Example spaces:

1. $\{1, \dots, 5\}$ with a distance function $d(x, y) = |x - y|$. This distance function can also be applied to age, income etc. spaces.
2. $\{‘M’, ‘F’\}$ with a distance function $d(‘M’, ‘M’)= d(‘F’, ‘F’)=0$ and $d(‘F’, ‘M’)= d(‘M’, ‘F’)=1$.
3. The set of submitted names with a distance function $d(x, x) = 0$ and, if $x \neq y$, $d(x, y) = 1$ (as above).
4. The set of submitted short comments with a distance function based on a topical analysis of the comments.

5. A GIS (geographic information system) that associates addresses with longitude and latitudes can also be used. A suitable distance function might give the Euclidean distance between addresses, or perhaps even the travelling time.

By combining n “question” spaces $\langle M_1, d_1 \rangle, \dots, \langle M_n, d_n \rangle$, a single n -dimensional classification space $\langle M, d \rangle$ can be formed where $M = M_1 \times \dots \times M_n$ and $d = G_p^{d_1 \dots d_n}$. Each point in this space represents a possible way to complete the questionnaire.

Perhaps machine readable paper questionnaires are scanned before being attached to the information space. If so, it may be useful to store scanned questionnaires so that their information space representation can be manually checked. If this is the case, as each scanned questionnaire image is unique—even for identical questionnaires—and information units are scanned images, a *type one index* (see section 8.6.2) is required.

Perhaps the questionnaire is computer based—so responses are submitted via check-boxes, radio-buttons or text entry fields—then a point $x \in M$ can record *all* the information in the questionnaire. So a questionnaire u can be *entirely* reconstructed as long as its attach point x is known. If this is the case a *type two index* (see section 8.6.2) is required.

Whichever type of index is used, given a questionnaire information space (M, d, \mathcal{I}) and an \mathbb{N}_1 -collection of questionnaires $\mathcal{N} = (N, m_N)$, the \mathbb{N}_1 -collection of points \mathcal{X} they are attached to can be found using the operator:

$$\mathcal{X} = \text{get_collection}(\mathcal{N}, \mathcal{I}).$$

Two \mathbb{N}_1 -collections of questionnaires, \mathcal{N}_1 and \mathcal{N}_2 , can be compared using

$$\mathcal{X} = \text{get_collection}(\mathcal{N}_1, \mathcal{I}),$$

$$\mathcal{Y} = \text{get_collection}(\mathcal{N}_2, \mathcal{I})$$

$$\text{and } {}^{\mathcal{M}}_k d(\mathcal{X}, \mathcal{Y})$$

If preferred, a list T of dimension indexes can be used to “select” dimensions of interest, taking the projected distance $({}^{\mathcal{M}}_k d)^-(\mathcal{X}^-, \mathcal{Y}^-)$.

Given a set of points $X \subseteq M$, the number c of attached questionnaires can be counted:

$$c = \sum_{\{(u,x) \in \mathcal{I} | x \in X\}} m((u,x)).$$

Given a list T of dimension indexes, use

$$Y = \{y \in M | d^-(x^-, y^-) \leq r\}$$

and then

$$\sum_{\{(u,y) \in \mathcal{I} | y \in Y\}} m((u,y))$$

to count the number of completed questionnaires with a projected distance no greater than r from a point $x \in M$.

Given an “origin” $(z_1, \dots, z_n) \in M$ where, for $1 \leq i \leq n$, z_i is a “base” answer to question i , it is possible to define n ordering relations R_i where, for each dimension M_i , for each pair $x_i, y_i \in M_i$, $x_i R_i y_i$ iff $d_i(z_i, x_i) \leq d_i(z_i, y_i)$. After the coordinates of each dimension have been ordered in this way, dimension pairs can be selected to provide the axes for scatter graphs that plot points with attached questionnaires.

9.2.1 Example Questionnaire Information Space

Consider a very brief questionnaire consisting of the questions:

1. Profession,
2. Salary, and
3. Hours worked per week (note that $7 \times 24 = 168$).

The dimensions of our information space will be:

1. $\langle M_1, d_1 \rangle$ where M_1 is a set of possible professions and, for all $x, y \in M_1$, $d_1(x, x) = 0$ and, if $x \neq y$, $d_1(x, y) = 1$.
2. $\langle \mathbb{R}^{>0}, d_2 \rangle$ where, for all $x, y \in \mathbb{R}^{>0}$, $d_2(x, y) = x - y$.

3. $\langle (0, 168], d_3 \rangle$ where, for all $x, y \in (0, 168]$, $d_3(x, y) = x - y$.

These spaces can be combined into an n -dimensional classification space $\langle M, d \rangle$ where $M = M_1 \times \mathbb{R}^{>0} \times (0, 168]$ and $d = G_p^{d_1 d_2 d_3}$.

Now, after defining a suitable index $\mathcal{I} = (J, m)$ which associates information units with points in the classification space (see sections 8.6.2 and 8.7), we have an information space (M, d, \mathcal{I}) . We also have (M^I, m_I) where $M^I = \{x | (u, x) \in \mathcal{I}\}$ and, for each $x \in M^I$, $m_I(x) = \sum_{(u, x) \in \mathcal{I}} m((u, x))$.

How much more, on average, do Doctors earn than Nurses?

To answer this question, we require the set of points X with “Doctor questionnaires” attached and the set of points Y of points with “Nurse questionnaires” attached. That is,

$$\begin{aligned} X &= \{(\text{Doctor}, x_2, x_3) | (u, (\text{Doctor}, x_2, x_3)) \in \mathcal{I} \text{ for some } u, x_2, x_3\} \text{ and} \\ Y &= \{(\text{Nurse}, x_2, x_3) | (u, (\text{Nurse}, x_2, x_3)) \in \mathcal{I} \text{ for some } u, x_2, x_3\}. \end{aligned}$$

Now if, for each $x \in X$, $m_X(x) = \sum_{(u, x) \in J} m((u, x))$ and, for each $y \in Y$, $m_Y(y) = \sum_{(u, y) \in J} m((u, y))$ then $m_X(x)$ gives the number of “Doctor questionnaires” attached to each point $x \in X$ while $m_Y(y)$ gives the number of “Nurse questionnaires” attached to each point $y \in Y$.

If $\mathcal{X} = (X, m_X)$, $\mathcal{Y} = (Y, m_Y)$ and $T = 2$ then the “average salary” of Doctors answering the questionnaire is

$$\frac{1}{|\mathcal{X}^-|} \sum_{x^- \in X^-} m_{X^-}(x^-) x^-$$

(see “projected classification spaces” section 8.4.3) and the answer to the question “How much more, on average, do Doctors earn than Nurses?” is

$$\left(\frac{1}{|\mathcal{X}^-|} \sum_{x^- \in X^-} m_{X^-}(x^-) x^- \right) - \left(\frac{1}{|\mathcal{Y}^-|} \sum_{y^- \in Y^-} m_{Y^-}(y^-) y^- \right).$$

As—from the definition of L -collection cardinality (see section 6.3)—

$\frac{1}{|\mathcal{Y}^-|} \sum_{y^- \in Y^-} m_{\mathcal{Y}^-}(y^-) = 1$ and $\frac{1}{|\mathcal{X}^-|} \sum_{x^- \in X^-} m_{\mathcal{X}^-}(x^-) = 1$ this is

$$\begin{aligned}
& \frac{1}{|\mathcal{X}^-||\mathcal{Y}^-|} \left(\left(\sum_{x^- \in X^-} m_{\mathcal{X}^-}(x^-) x^- \sum_{y^- \in Y^-} m_{\mathcal{Y}^-}(y^-) \right) \right. \\
& \quad \left. - \left(\sum_{x^- \in X^-} m_{\mathcal{X}^-}(x^-) \sum_{y^- \in Y^-} m_{\mathcal{Y}^-}(y^-) y^- \right) \right) \\
&= \frac{1}{|\mathcal{X}^-||\mathcal{Y}^-|} \sum_{x^- \in X^-} m_{\mathcal{X}^-}(x^-) \left(x^- \left(\sum_{y^- \in Y^-} m_{\mathcal{Y}^-}(y^-) \right) - \left(\sum_{y^- \in Y^-} m_{\mathcal{Y}^-}(y^-) y^- \right) \right) \\
&= \frac{1}{|\mathcal{X}^-||\mathcal{Y}^-|} \sum_{x^- \in X^-} m_{\mathcal{X}^-}(x^-) \left(\sum_{y^- \in Y^-} m_{\mathcal{Y}^-}(y^-) (x^- - y^-) \right) \\
&= (d_2)_{\text{av av}}(\mathcal{X}^-, \mathcal{Y}^-).
\end{aligned}$$

Adding a pay rate dimension

Perhaps we decide that we are really interested in comparing the *pay rate* of Doctors and Nurses. Indeed, we are generally interested in pay rate, so we create a “pay rate” space $\langle \mathbb{R}^{\geq 0}, d_4 \rangle$ where $d_4(x, y) = x - y$. We use this as a dimension in the 4-dimensional space $\langle M^+, d^+ \rangle$ where $M^+ = M_1 \times \mathbb{R}^{>0} \times (0, 168] \times \mathbb{R}^{\geq 0}$ and $d^+ = G_p^{d_1 d_2 d_3 d_4}$.

We also require a new index:

$$\mathcal{I}^+ = \text{add_dimension}(\mathcal{I}, \mathbb{R}^{\geq 0})$$

where the ‘suitable attribute’ of each information unit u , where $(u, (z_1, z_2, z_3)) \in J$, is $\frac{z_2}{50z_3}$. Note that, as z_2 is “salary” and z_3 is “hours worked per week”, $\frac{z_2}{50z_3}$ gives an hourly pay rate—assuming both doctors and nurses work 50 weeks a year.

We can now compare the average *pay rate* of Doctors and Nurses:

$$X^+ = \{(\text{Doctor}, x_2, x_3, x_4) | (u, (\text{Doctor}, x_2, x_3, x_4)) \in \mathcal{I}^+ \text{ for some } u, x_2, x_3, x_4\},$$

$$Y^+ = \{(\text{Nurse}, x_2, x_3, x_4) | (u, (\text{Nurse}, x_2, x_3, x_4)) \in \mathcal{I}^+ \text{ for some } u, x_2, x_3, x_4\}.$$

Now if, for each $x^+ \in X^+$, $m_{X^+}(x^+) = \sum_{(u, x^+) \in J^+} m^+((u, x^+))$ and, for each $y^+ \in Y^+$, $m_{Y^+}(y^+) = \sum_{(u, y^+) \in J^+} m^+((u, y^+))$ then $m_{X^+}(x^+)$ gives the number of “Doctor questionnaires” attached to each point $x^+ \in X^+$ while $m_{Y^+}(y^+)$ gives the number of “Nurse questionnaires” attached to each point $y^+ \in Y^+$.

If $\mathcal{X} = (X^+, m_{X^+})$, $\mathcal{Y} = (Y^+, m_{Y^+})$ and $T = 4$ then $((d^+)^-)_{\text{av av}}(\mathcal{X}^-, \mathcal{Y}^-)$ is the distance we want.

Of course we could improve the accuracy of this result by first adding a “weeks worked per year” question.

9.3 Information Space for Research Paper Knowledge Libraries

We can create a basic research paper classification space with Title, Author, Subject, Journal, Length and Cites dimensions:

1. $\langle M_1, d_1 \rangle$ is a space where M_1 is a set of paper titles and d_1 is one of the information retrieval distance functions we discuss in section 4.2.3.
2. $\langle M_2, d_2 \rangle$ is a space with, $M_2 = \mathcal{P}(Y_2)$, where Y_2 is a set of paper authors and d_2 is a distance function, again from section 4.2.3.
3. $\langle M_3, d_3 \rangle$ is a space where $M_3 = \mathcal{P}(Y_3)$, Y_3 is a set of subjects, d_3 is a set distance function based on d_3^* and $\langle Y_3, d_3^* \rangle$ is a networked space.
4. $\langle M_4, d_4 \rangle$ is a networked space (by subject) where M_4 is a set of Journals (so d_4 is defined by edge weights and node spans, see section 8.3).
5. $\langle M_5, d_5 \rangle$ is a space where $M_5 = \mathbb{N}_1$ is a space of paper lengths (by word count) and $d_5(x, y) = x - y$ for all $x, y \in M_5$.
6. $\langle M_6, d_6 \rangle$ is a networked space where M_6 is the set of all research papers in the information space and $d_6(x, y) = 1$ if x “cites” y .

We combine these spaces into a 6-dimensional classification space $\langle M, d \rangle$ where $M = M_1 \times M_2 \times M_3 \times M_4 \times M_5 \times M_6$ and $d = G_p^{d_1 d_2 d_3 d_4 d_5 d_6}$.

Given a suitable index \mathcal{I} , we now have an information space (M, d, \mathcal{I}) which allows us to perform a number of useful functions.

Note that each paper attached to the space can have only one title, but may have a number of authors and may be about a number of subjects.

9.3.1 Selecting and Comparing Research Papers

If $x \in M$ then we can select all papers within r of x with

$$\{u | (u, y) \in \mathcal{I}, d(x, y) \leq r\}.$$

We can use a list of dimension indexes T to define a projection, and so restrict the search to only those dimensions of interest.

$$\{u | (u, y) \in \mathcal{I}, y^- \text{ is a projection of } y, d^-(x^-, y^-) \leq r\}.$$

If u is a research paper attached to the space then we find the point x where $(u, x) \in \mathcal{I}$. Then $\{u | (u, y) \in \mathcal{I}, d(x, y) \leq r\}$ is the set of research papers within r of u .

Similarly, if $X \subseteq M$ then we can select all papers within r of X with

$$\{u | (u, y) \in \mathcal{I}, {}^M_k d(X, \{y\}) \leq r\}.$$

If N is a set of information units attached to the space, we first define $X = \{x | (u, x) \in \mathcal{I}, u \in N\}$. Then $\{u | (u, y) \in \mathcal{I}, {}^M_k d(X, \{y\}) \leq r\}$ is the set of all information units with a ${}^M_k d$ distance of no more than r from N .

If we want to compare a paper u_1 with a paper u_2 we find x where $(u_1, x) \in J$ and y where $(u_2, y) \in J$. Then

$$d(x, y)$$

is the distance we require. If we want to use only dimensions T , $d^-(x^-, y^-)$ is what we want.

Similarly, if we want to compare two sets of papers N_1 and N_2 we can use $X = \{x | (u, x) \in \mathcal{I}, u \in N_1\}$, $Y = \{x | (u, x) \in \mathcal{I}, u \in N_2\}$ and

$${}_k^M d(X, Y).$$

If we want to use only dimensions T , ${}_k^M d^-(X^-, Y^-)$ is what we want.

If the number of papers attached to each point may be significant, we define, for each $x \in X$, $m_a(x) = \sum_{(u,x) \in J} m((u, x))$ and, for each $y \in Y$, $m_Y(y) = \sum_{(u,y) \in J} m((u, y))$. If $\mathcal{X} = (X, m_X)$ and $\mathcal{Y} = (Y, m_Y)$,

$${}_k^M d(\mathcal{X}, \mathcal{Y})$$

is a distance that takes the number of papers attached to each point in account.

Another way to compare papers u_1 and u_2 is to find the closest common “ancestor” paper (by citation). To do this we first find y_1 where $(u_1, y_1) \in J$ and y_2 where $(u_2, y_2) \in J$. The citation dimension is dimension 6, so, with $T = 6$, the closest common ancestor paper (by citation) is the paper u where $(u, x) \in J$ and $(d^-)_{100}^M(x^-, \{y_1, y_2\}^-)$ is minimised.

If u_1 and u_2 are well established papers, the closest common descendant paper (by citation)—which is the paper u where $(u, x) \in J$ and $(d^-)_{100}^M(\{y_1, y_2\}^-, x^-)$ is minimised—could be interesting.

9.3.2 Extended Dimensions

There are a number of ways we can extend the basic dimensions suggested above.

Extending the “Title” Dimension

We could improve on the information retrieval keyword matching formula by making use of contextual information from the subject dimension to disambiguate keywords.

Example 9.1. The keyword “Model” means something different when used

in the context of “Model Theory” to “Model”, when used in other branches of mathematics. Combining “Title” and “Subject” dimensions to create a new dimension would make available distinct points (Model, General Mathematics) and (Model, Model Theory) for different uses of the same keyword.

Extending the “Author” Dimension

We may want to record the contribution percentage of the authors of each paper. If Y_2 is a set of authors, d_2^* a distance function over Y_2 , $\mathcal{Y}_2 = (Y_2, m_{Y_2})$ is a $(0, 1]$ -collection where $m_{Y_2}(y) = 1$ for all $y_2 \in Y_2$ we can have an “Author” dimension $\langle M_2, d_2 \rangle$ where $M_2 = \mathcal{P}(\mathcal{Y}_2)$ and $d_2 = \mathcal{Y}_2^2 d_2^*$. Now each point in $\langle M_2, d_2 \rangle$ is a $(0, 1]$ -collection of authors. We might choose to attach a paper u to a point $\{^c\text{Smith}|_{\frac{1}{2}}, \text{Brown}|_{\frac{1}{3}}, \text{Jones}|_{\frac{1}{6}}\} \in M_2$.

Example 9.2. If

1. Y_2 is a set of authors,
2. d_2^* a distance function over Y_2 where $d_2^*(x, x) = 0$ and $d_2^*(x, y) = 1$ for all $x, y \in Y_2$ where $x \neq y$,
3. $\mathcal{Y}_2 = (Y_2, m_{Y_2})$ is a $(0, 1]$ -collection where $m_{Y_2}(y) = 1$ for all $y_2 \in Y_2$,
4. $M_2 = \mathcal{P}(\mathcal{Y}_2)$ and $d_2 = \mathcal{Y}_2^2 d_2^*$

then, with $T = 2$ and

$$Y = \{y \in M | d_2(\{^c\text{Smith}|_{\frac{1}{2}}\}, y^-) = 0, y^- \text{ is a projection of } y\},$$

$$\{u | (u, y) \in \mathcal{I}, y \in Y\}$$

is the set of papers at least 50% by Smith.

If we have a space of “Researchers”, with dimensions like “Name”, “Age”, “Gender”, “Research Interests”, “School” and “Contact e-mail” another possibility would be to use the information units (Researchers) in this space as coordinates in our “Author” dimension.

Extending the “Subject” Dimension

Subject hierarchies for classification, such as the Dewey decimal system¹ have tended to have quite broad or *coarsely grained* “subject” nodes. One important reason for this has been the necessity of identifying only one subject for each information unit (book), so that the unit can be placed in a corresponding location in physical space (on the library shelves). Our subject dimension—where each point is a *set* of subjects—has no such constraint.

Indeed, rather than a “subject hierarchy”, we could make use of a much more finely grained “concept hierarchy” R where xRy implies that the concept x is required to understand the concept y . “Subjects” could be sets (possibly even rough or fuzzy-sets) of concepts.

If $\langle Y_3, d_3^* \rangle$ is a networked space where Y_3 is a set of “concepts” then points in the concept dimension $\langle M_3, d_3 \rangle$, where $M_3 = \mathcal{P}(Y_3)$ and $d_3 = (d_3^*)_{ij}^{Y_3}$ or ${}_k^{Y_3}d_3^*$, could also be used to represent *user knowledge*.

Example 9.3. Given a set X of all the concepts in Y_3 I understand, with $T = 3$ and $Y = \{y \in M | d_3(X, y^-) = 0, y^- \text{ is a projection of } y\}$,

$$\{u | (u, y) \in \mathcal{I}, y \in Y\}$$

is the set of research papers for which I have the required background knowledge.

9.3.3 Further Dimensions

Other dimensions such as “index terms”, “Abstract”, “Full Text” could also be included. These dimensions would allow information units to be searched for (or compared by) index terms, abstract or full text respectively.

¹Discussed briefly in section 2.3

9.4 What this Chapter Achieved

This chapter outlines, through two worked examples, how information space can be used to provide core Knowledge Library functionality.

Chapter 10

The Efficient Implementation of Knowledge Libraries

10.1 Overview

This chapter discusses how the less easily implementable core Knowledge Library functionality can be provided. The desired functionality is provided by implementations of **range query** algorithms/data structures, either directly, or after some modification.

Section 10.2 introduces the problem, outlining the basic types of search queries in the range query group. These include **distance**, **range**, **k nearest neighbour** and **ranked** queries. The sequential search range query algorithm is used to illustrate the time complexity challenges associated with these queries.

Section 10.3 reviews the relevant literature. Although range query algorithms have not been developed for searching set spaces, a body of literature exists that discusses the closely related problem of searching metric space. These range query algorithms are characterised as either **radius partitioning** or **hyperplane partitioning**. The section includes a brief discussion of how range query algorithms/data structures can be adapted to provide k nearest neighbour and ranked query functionality.

Section 10.4 discusses how radius partitioning and hyperplane partition-

ing algorithms can be adapted for set spaces. The main difficulty is due to the lack of the symmetry property.

Section 10.5 discusses the difficulties involved in searching **hard spaces**. Due to the relatively small variance of some distance functions over n -dimensional spaces where n is large, high dimensional spaces can be difficult to search.

Section 10.6 briefly discusses how this chapter contributes towards the development of Knowledge Libraries.

10.2 Introduction

Given sensible data structures corresponding to the classification space, index and information space mathematical objects described in chapter 8, much of the Knowledge Library core functionality, described in chapter 3, is straightforward to implement. This chapter discusses the core functionality that, due to time complexity considerations, is often less easily implementable. That is, the chapter discusses distance query, range query, k nearest neighbour query and ranked query algorithms.

10.2.1 Distance query

The distance query is a very basic type of query that underpins other common queries. Given a set space $\langle M, d \rangle$, a distance query determines $d(x, y)$ for an $x, y \in M$. For many spaces, $d(x, y)$ is computationally expensive to determine. For example, M^I could be a set of images¹ and $d(x, y)$ could be based on a pixel by pixel comparison. Because of this, algorithms—such as range query and k nearest neighbour query—that rely on distances are usually most efficient when the number of required distance computations (or “look ups”) can be limited.

¹Recall from chapter 8 that $M^I \subseteq M$ is the set of points in M that have attached information elements.

10.2.2 Range query

Range query algorithms are often used as the basis for k nearest neighbour query algorithms. Range queries are used to determine the set of points $Y = \{y \in M^I \mid d(x, y) \leq t\}$ for some arbitrary “query” point $x \in M$ and range limit t . Efficiency considerations generally dictate that distances be precomputed and indeed the number of (precomputed) distances “looked-up” be minimised when executing a range query.

10.2.3 k Nearest neighbour query

When searching spaces, users are often interested in the information elements that are closest to (most similar to) a particular point in the space. Often only the first 5, or 10, or 20 information elements are of interest. range queries can be used to accomplish this task, but it is rarely apparent what value of t should be specified in order to select an appropriate number of information elements. If too small a value of t is chosen, too few, often no, information elements are selected. If too large a value is chosen, too many information elements are selected.

It is often more appropriate to specify the number k of information elements that are desired. The literature (summarised in [22, 43]) does not consider the possibility of more than one information element being attached to each point, so k nearest neighbour, abbreviated k -NN, queries are used to determine a set $Y \subseteq M^I$ where $|Y| = k$ and, for all $y \in Y$ and $y' \in M^I - Y$, $d(x, y) \leq d(x, y')$. Fortunately only minor amendments are required to generalise existing k -NN algorithms (which are themselves closely based on range query algorithms) to determine a similar (minimal) set Y where (rather than $|Y| = k$), given (J, m) is an index over M , $\sum_{\{y \in Y\}} m(y) \geq k$.

10.2.4 Ranked query

A ranked query is essentially a k -NN query where the resulting set Y is ranked according to $d(x, y)$ for each $y \in Y$. Ranked query algorithms based on best-first tree traversal exist[43] that, after the first k information elements

have been retrieved, retrieve the *next* k information elements if desired (and so on)².

10.2.5 Sequential search range query algorithm

The “brute force” approach to range queries is to simply sequentially check each point in the space with at least one attached information unit. Implementations of this algorithm use an “information unit surrogate table” that “attaches” information units to points in the space. The table has rows (indexed by *point*):

point; iu_name; iu_address

where *iu_name* is the name and *iu_address* is the address (perhaps a URI) of an information unit. To select all information elements attached to a point y where $d(x, y) \leq t$, the algorithm simply iterates through the table, checking the distance from x to each point y in the table. If the number n of information elements is small or distances can be computed quickly, this method is effective. However, if n is very large and distances are computationally expensive to determine (which is the standard assumption), this can take too long— $O(nm)$ where m is the time complexity of distance computation.

10.3 Metric Space Algorithms

A number of researchers have considered the closely related problem of searching metric spaces. This research is adequately summarised in [22], a paper that provides the basis for the more focused and refined summary [43].

Generally, these algorithms rely on a “search tree” data structure (sometimes implemented with an array). The root node of the tree is the set of all the points (with attached information elements) in the space. Other nodes are a subset of their parent. The set of all children of a node is a partition

²Users of internet search engines, such as Google, will be familiar with this type of ranked query.

of that node. Nodes are partitioned according to a “split criterion”, which involves a “center”³ (or pivot) point, either for each node, or for each level of the search tree.

A number of different algorithms have been used to select centers for (see [85, 103]). One standard technique involves, given a space $\langle M, d \rangle$, selecting, from a number of random samples of M , the sample with the maximum, minimum distance between points.

Because they are self balancing and can be constructed so that nodes fit precisely into a computer’s page size—optimising read operations—B-trees[8], are a suitable data structure for these search trees. The use of B-trees to index metric spaces is discussed in [23].

The traversal of the search tree, and so the search for points y where $d(x, y) \leq t$ (for some “query point” x), is guided by a “search criterion”. This criterion is developed from the split criterion, ΔI , the symmetry property, and the inequality $d(x, y) \leq t$. The motivation behind the search criterion is to reduce the number of distance calculations that are required to determine the set $Y = \{y \in M | d(x, y) \leq t\}$, so distances such as $d(x, y)$, for $x, y \in M$, are approximated. Importantly, this approximation is done in such a way as to ensure that no node containing a point in Y is pruned from the search.

The subsections below provides a “distillation” of the literature. Quite a number of notions that, upon critical examination, did not appear particularly useful or effective, have been left out. The remaining concepts have been expressed succinctly.

10.3.1 Relative ordering

One attempted improvement of the sequential search range query algorithm involves ordering the points so that an initial point, close to x , can be selected before continuing to search “outwards”, checking progressively more distant points until the search is terminated with the discovery of a too distant point.

³For consistency with the literature, the spelling “center”, rather than the more standard “centre.”, is used in this thesis.

If the distance limit t is relatively small, this sort of search is very efficient⁴.

Relative ordering algorithms rely on the following two inequalities. For all points y where $d(x, y) \leq t$, from $\triangle I$ and symmetry,

$$d(c, y) \geq d(x, c) - t \quad (10.1)$$

and

$$d(c, y) \leq d(c, x) + t. \quad (10.2)$$

In essence, points y in the space are ordered by $d(c, y)$. The search for all points y where $d(x, y) \leq t$ proceeds as follows. To begin with, the first point y where $d(c, y) \geq d(x, c) - t$ is located in the ordered list. The inequality $d(x, y) \leq t$ is checked for this, and each successive, y in the list until a y is reached where $d(c, y) > d(c, x) + t$.

The relative ordering approach is extended by radius partitioning.

10.3.2 Radius partitioning

Radius partitioning, based on the “covering radius criterion” [22] uses “covering radii” to define tree nodes. Most simply, a “ball” with radius r_c partitions each node Y_c , with “center point” c , into two parts (inside and outside the ball). More generally, m “shells” are created. That is, for $1 \leq i \leq m$,

$$Y_{c_i} = \{y \in Y_c | r_{c_{i-1}} \leq d(y, c) < r_{c_i}\}$$

where $r_{c_0} = 0 < r_{c_1} < \dots < r_{c_m} = \max_{y' \in Y_c} d(y', c) + 1$.

From $\triangle I$, $d(x, c) \leq d(x, y) + d(y, c)$. For each point $y \in Y_{c_i}$, $d(y, c) < r_{c_i}$. From this, when searching for points $y \in Y_c$ where $d(x, y) \leq t$, it is only necessary to search nodes Y_{c_i} where

$$d(x, c) < t + r_{c_i}. \quad (10.3)$$

Similarly, from $\triangle I$, $d(y, c) \leq d(y, x) + d(x, c)$. For each point $y \in Y_{c_i}$,

⁴For relatively large t , there will be little improvement over the sequential search algorithm. For “large” spaces, there are practical limits to the size of t .

$r_{c_{i-1}} \leq d(y, c)$. From symmetry, $d(y, x) = d(x, y)$. From this, if points $y \in Y_c$ where $d(x, y) \leq t$ are required, it is only necessary to search nodes Y_{c_i} where

$$r_{c_{i-1}} \leq t + d(x, c). \quad (10.4)$$

Combining inequalities (10.3) and (10.4) gives the “covering radius node search criterion”

$$r_{c_{i-1}} - t \leq d(x, c) < t + r_{c_i}. \quad (10.5)$$

This approach results in “covering radius” range query algorithms such as the one below.

```

rangeQuery( $Y, x, t$ )
   $result = \emptyset$ 
  if  $Y$  is a leaf node
    for each  $y \in Y$ 
      if  $d(x, y) \leq t$ ,  $result = result \cup \{y\}$ 
    return  $result$ 
  for each child node  $Y_{c_i}$  of  $Y$ 
    if  $r_{c_{i-1}} - t \leq d(x, c) < t + r_{c_i}$  then
       $result = result \cup \text{rangeQuery}(Y_{c_i}, x, t)$ 
  return  $result$ 

```

Another way to generalise covering radius partitioning is to use a number of centers (rather than a single center and a number of shells) to partition each node. However it is often not possible to effectively partition the space in this way without substantial overlap between balls, reducing search efficiency. These difficulties tend to increase proportionally to the number of dimensions of the space, as is discussed in section 10.5.2.

For radius partitioning algorithms, it is best to use only one “center point” per level of the search tree, so all nodes on the same level share the same “center”. Following this policy, usually the center c of a node Y_c is $c \notin Y_c$. This limits the number of distances such as $d(x, c)$ that have to be computed when searching the tree.

The main advantage of the center selection algorithm that maximises the minimum distance between centers with (shell type) radius partitioning, in Euclidian space at least, is that the space is quite effectively “sectioned”, so the “diameter” $\max_{x,y \in Y_c} d(x,y)$ of nodes Y_c decreases towards the leaf nodes. This sectioning effect also occurs in many non-Euclidian spaces.

10.3.3 Hyperplane partitioning

Hyperplane partitioning[22] involves, given a tree node Y_c and m distinct centers $c_1 \in Y_c, \dots, c_m \in Y_c$, dividing the points $y \in Y_c$ so that

$$Y_{c_i} = \{y \in Y_c \mid d(y, c_i) < d(c_j, y) \text{ for each } 1 \leq j \leq m\}.$$

Note that, if for all $y \in Y_c$ there are no ties where $d(y, c_i) = d(c_j, y)$, $\{Y_{c_1}, \dots, Y_{c_m}\}$ partitions Y_c as $d(y, c_i) = d(c_i, y)$ due to symmetry.

For any y where $d(x, y) \leq t$, from ΔI ,

$$d(y, c_i) \geq d(x, c_i) - d(x, y) \geq d(x, c_i) - t$$

and

$$d(c_j, y) \leq d(c_j, x) + d(x, y) \leq d(c_j, x) + t.$$

The above inequalities can be used to assist in searching the tree. With $d(x, c_i) - t$ for $d(y, c_i)$ and $d(c_j, x) + t$ for $d(c_j, y)$ in the partitioning inequality $d(y, c_i) < d(c_j, y)$, it can be concluded that, if

$$d(x, c_i) - t < d(c_j, x) + t \tag{10.6}$$

for all $1 \leq j \leq m, j \neq i$, there may be points $y \in Y_{c_i}$ where $d(x, y) \leq t$. This approach produces “hyperplane partitioning” range query algorithms such as the one below.

rangeQuery(Y, x, t)

$result = \emptyset$

if Y is a leaf node

```

for each  $y \in Y$ 
    if  $d(x, y) \leq t$ ,  $result = result \cup \{y\}$ 
return  $result$ 
for each child node  $Y_{c_i}$  of  $Y$ 
    if  $d(x, c_i) - t < d(c_j, x) + t$  for all  $1 \leq j \leq m, j \neq i$ 
         $result = result \cup rangeQuery(Y_{c_i}, x, t)$ 
return  $result$ 

```

Note that, while a generic algorithm has been presented, a time complexity analysis of this algorithm suggests that the optimum number of children for each node is 2. Also, if $d(y, c_i) = d(c_j, y)$ for some c_i, c_j and $y \in Y_c$, a more complex definition of Y_{c_i} is required. That is

$$Y_{c_i} = \{y \in Y_c \mid d(y, c_i) < d(c_j, y) \text{ for each } 1 \leq j < i$$

$$\text{and } d(y, c_i) \leq d(c_j, y) \text{ for each } i < j \leq m\}.$$

Corresponding modifications to the above algorithm are also necessary.

10.3.4 Ranked query and k -NN query algorithms

Ranked query and k -NN query algorithms can be obtained by modifying range query algorithms/data structures. Perhaps the most effective way of doing this is to do a best-first-search, rather than a depth-first-search of the search tree. Nodes Y_c , where $\min_{y \in Y_c} \{d(x, c) - d(y, c)\}$ (which estimates $\min_{y \in Y_c} \{d(x, y)\}$) is small, are searched first. When leaf nodes are reached, only those points y where

$$d(x, c) - d(y, c) \leq \max_{y' \in Y_{c'}} \{d(x, c') + d(c', x)\}$$

and

$$d(x, y) \leq \max_{y' \in Y_{c'}} \{d(x, c') + d(c', x)\}$$

for a number of “candidate nodes” $Y_{\mathcal{C}}$ are included in the ranked “candidate points” set. This is done to eliminate points y where there is no chance that y might be included in the final ranked set of k points[43].

10.3.5 A critique of the literature

A number of things are not well expressed in the literature.

While there are a number of papers that discuss the topic of searching metric spaces, it is normal for each paper to include a description of the particular problem that the paper attempts to solve. Many papers that at first appear to be addressing the same problem do, in fact, address subtly different problems. In order for comparisons between different techniques to be meaningful, the different techniques need to be attempting to achieve the same result. It is also important that testing is performed in a standardised manner for comparisons between test results to be meaningful. The *de facto* standard appears to be to use uniformly distributed points in Euclidean space. While this standard may be appropriate for “Euclidean space algorithms”, to warrant the name, “metric space algorithms” should also be tested on non Euclidean metric spaces. To advance this topic, it is necessary

1. for the metric space searching problems to be formally defined, and for metric space papers to refer to this formal definition;
2. that, for comparison purposes, proponents of particular metric space searching algorithms make standard reference implementations available;
3. to develop standardised data sets (metric spaces) on which metric space searching algorithms may be tested.

In [23] it is observed that, due to disk latency (that is, the time it takes to position the read head over the selected disk sector), it may be preferable to store more than one point in each leaf node. Indeed it is advocated that the number of points stored in leaf nodes be limited only by the computer’s page size, so each leaf node can be “read” in a single operation. However, while

this optimises disk usage in terms of point access, it has the disadvantage that the distances for all the points in each selected leaf node must be computed (or looked up) each query. The literature does not provide a formula for the optimum number of points in each leaf node.

10.4 Adapting Metric Space Algorithms for Set Space

There are two key difficulties when adapting the metric space algorithms discussed for set space spaces. First, the adapted algorithms rely on $G\Delta I$, rather than ΔI , so they must cope with the insertion of spans into affected inequalities. Second, as set spaces often do not have the symmetry property, the adapted algorithms must not depend on this property.

The search criterion of each of the three metric space range query algorithms discussed relies on two fundamental inequalities. The relative ordering search criterion relies on

$$d(c, y) \geq d(x, c) - t \text{ and } d(c, y) \leq d(c, x) + t,$$

the radius partitioning search criterion relies on

$$d(x, c) < t + r_{c_i} \text{ and } r_{c_{i-1}} \leq t + d(x, c)$$

and the hyperplane partitioning search criterion relies on

$$d(y, c_i) \geq d(x, c_i) - t \text{ and } d(c_j, y) \leq d(c_j, x) + t.$$

For each adapted algorithm, if the existing metric space split criterion is used, because $G\Delta I$ must be satisfied rather than ΔI , $s_d(y)$ must be inserted into one of the two fundamental inequalities. As has already been discussed (see section 10.3), it is important that each search criterion does not involve y , so this immediately renders the inequality unusable.

Because symmetry does not hold for set spaces in general, and one of the fundamental inequalities always relies on symmetry, one fundamental inequality does not hold, even after the insertion of appropriate spans. The

fundamental inequality that does not hold can be swapped by reversing the order of the arguments in each split criterion. The usefulness of this is not immediately apparent however as both orders cannot be used simultaneously.

Fortunately, these problems are not insurmountable. The unwanted span $s_d(y)$ can be moved to the split criterion where it does not cause any problems, while another inequality can be defined to replace the remaining problematic fundamental inequality.

Note that, because node partitions are based on one argument order, the replacement inequality, which works best when node partitions are based on the other argument order (for spaces that lack the symmetry property), may be ineffective at pruning nodes from the search. This potential defect could be addressed by reversing the order of the arguments in the split criterion on alternate levels of the search tree. This alternation does not increase the memory or time complexity of the algorithm and the “friendly” partition from the previous level increases the effectiveness of the replacement inequality.

10.4.1 Relative ordering for set spaces

For set spaces spaces (that is, spaces that satisfy $G\Delta I$) the inequality corresponding to (10.1) is

$$s_d(y) + d(y, c) \geq d(x, c) - t.$$

It is important that the span $s_d(y)$ is kept on the LHS of this inequality, so the points y are ordered by $s_d(y) + d(y, c)$. To begin the search for all points y where $d(x, y) \leq t$, locate the first y in the list where $s_d(y) + d(y, c) \geq d(x, c) - t$. Unfortunately, there is no appropriate inequality corresponding to (10.2) that begins $s_d(y) + d(y, c) \leq \dots$ so this technique can not be generalised to set spaces (even symmetric ones).

10.4.2 Radius partitioning for set space

Using $G\Delta I$ rather than ΔI , it is straight forward to determine the set space equivalent for inequality (10.3). This is

$$d(x, c) < t + s_d(y) + r_{c_i}.$$

However it is important to produce a search criterion that does not depend on y . To achieve this, the span of y is moved to the partitioning criterion. That is, a node Y_c with center c is partitioned into m parts using, for $1 \leq i \leq m$,

$$Y_{c_i} = \{y \in Y_c \mid r_{c_{i-1}} \leq d(y, c) + s_d(y) < r_{c_i}\}$$

where $r_{c_0} = 0 < r_{c_1} < \dots < r_{c_k} = \max_{y' \in Y_c} d(y', c) + 1$ as before. Now, because $d(y, c) + s_d(y) < r_{c_i}$, inequality (10.3) can be used without modification. Unfortunately, for set spaces that lack the symmetry property, there is no set space equivalent (with $r_{c_{i-1}}$) to inequality (10.4). To solve this problem, $r'_{c_i} = \min_{y \in Y_{c_i}} d(c, y)$ is determined for each node Y_{c_i} (at indexing time). Note that, unlike r_{c_i} , it is *not* expected that $r'_{c_1} < \dots < r'_{c_k}$. Using r'_{c_i} , a set space equivalent of inequality (10.4) can be determined. This is

$$r'_{c_i} \leq d(c, x) + s_d(x) + t.$$

With this the “set space covering radius node search criterion” (the set space equivalent to equation (10.5)) can now be expressed. That is, a node Y_{c_i} is searched iff

$$r'_{c_i} \leq d(c, x) + s_d(x) + t \text{ and } d(x, c) < t + r_{c_i}.$$

From this, the following “set space, covering radius, range query algorithm” is given.

rangeQuery(Y, x, t)

$result = \emptyset$

 if Y is a leaf node

 for each $y \in Y$


```

    if  $d(x, y) \leq t$ ,  $result = result \cup \{y\}$ 
  return  $result$ 
for each child node  $Y_{c_i}$  of  $Y$ 
  if  $r'_{c_i} \leq d(c, x) + s_d(x) + t$  and  $d(x, c) < t + r_{c_i}$  then
     $result = result \cup rangeQuery(Y_{c_i}, x, t)$ 
return  $result$ 

```

Note that each node Y_c is partitioned according to $d(y, c) + s_d(y)$, so the algorithm prunes nodes Y_{c_i} where $d(x, c)$ is comparatively large, effectively “zooming in” on nodes where $d(x, c)$ is small. The partition is not based on $d(c, y)$, so the lower limit $d(c, y)$ for all $y \in Y_{c_i}$ is somewhat random (although tending to be smaller when $|Y_{c_i}|$ is large). Because of this, the algorithm is not as effective at pruning nodes Y_{c_i} where $d(c, x)$ is small.

With this in mind, the algorithm might be improved by alternating the partitioning criteria between levels. On odd numbered levels, the partitioning criterion discussed above is used. On even numbered levels, $d(y, c) + s_d(y)$ is replaced with $d(c, y)$, giving

$$Y_{c_i} = \{y \in Y_c | r'_{c_{i-1}} \leq d(c, y) < r'_{c_i}\}.$$

where $r'_{c_0} = 0 < r'_{c_1} < \dots < r'_{c_k} = \max_{y' \in Y_c} d(c, y') + 1$. Note that each even numbered level can reuse the same center as the previous, odd numbered level, minimising memory usage.

When this, even numbered level, partitioning criterion is used, a set space equivalent of inequality (10.4) (with $r'_{c_{i-1}}$) is easily obtained, while, in order to obtain a set space equivalent of inequality (10.3), it is necessary to pre-determine $r''_{c_i} = \max_{y \in Y_{c_i}} \{d(y, c) + s_d(y)\}$ (the working is along the same lines as the above, so it is omitted).

Set space covering radius range query algorithms based on alternating level partitioning criteria should be effective at pruning nodes where $d(x, c)$ is large (on odd numbered levels) and nodes where $d(c, x)$ is small (on even numbered levels).

10.4.3 Hyperplane partitioning for set spaces

The general approach discussed in section 10.3.3 cannot be readily adapted for spaces that lack the symmetry property. Nodes $Y_{c_i} \subseteq Y$ are required to be defined by inequalities such as $d(y, c_i) \leq d(c_j, y)$ iff $i < j$ and $d(y, c_i) < d(c_j, y)$ iff $i > j$. For spaces that lack the symmetry property, the set of such nodes defined by these inequalities does not partition Y .

Restricting to just two centers c_1 and c_2 however, Y can be partitioned into two sets

$$Y_{c_1} = \{y \in Y | d(y, c_1) \leq d(c_2, y)\} \text{ and } Y_{c_2} = \{y \in Y | d(y, c_1) > d(c_2, y)\}.$$

Following a similar line of reasoning to that which lead to inequality (10.6), with $G\Delta I$ for ΔI , for hyperplane partitioned set spaces, the node Y_{c_1} is searched iff

$$d(x, c_1) - t - \max\{s_d(y) | y \in Y_{c_1}\} \leq d(c_2, x) + t + s_d(x).$$

However, without symmetry, $G\Delta I$ cannot be used to establish an appropriate upper bound for $d(y, c_1)$, so the node Y_{c_2} must always be searched.

One attempt at solving this problem involves storing covering radii for Y_{c_2} when constructing the search tree and using covering radius node search criteria to determine if Y_{c_2} needs to be searched. The partitioning criteria suggests four radii

$$\begin{aligned} r_1 &= \min_{y \in Y_{c_2}} \{d(c_2, y)\}, & r_2 &= \min_{y \in Y_{c_2}} \{d(c_1, y)\} \\ r_3 &= \max_{y \in Y_{c_2}} \{d(y, c_2) + s_d(y)\} & \text{and } r_4 &= \max_{y \in Y_{c_2}} \{d(y, c_1) + s_d(y)\}. \end{aligned}$$

The internal radii r_1 and r_2 can be used to define a version of inequality (10.4), which is used to prune relatively large (hollow) balls where $d(x, c)$ is small. From the partitioning criteria, normally $r_1 < r_2$, so r_2 should be the best internal radius to use.

The external radii r_3 and r_4 can be used to define a version of inequality (10.3), which is used to prune relatively small balls where $d(c, x)$ is large. From the partitioning criteria, normally $r_3 < r_4$, so r_3 should be the best external radius to use.

This approach leads to set space hyperplane partitioning algorithms such as the following.

```

rangeQuery( $Y, x, t$ )
   $result = \emptyset$ 
  if  $Y$  is a leaf node
    for each  $y \in Y$ 
      if  $d(x, y) \leq t$ ,  $result = result \cup \{y\}$ 
    return  $result$ 
  for child nodes  $Y_{c_1}$  and  $Y_{c_2}$  of  $Y$  (with  $r_2$  and  $r_3$  values for  $Y_{c_2}$ )
    if  $d(x, c_1) - t - \max\{s_d(y) | y \in Y_{c_1}\} \leq d(c_2, x) + t + s_d(x)$ 
       $result = result \cup \text{rangeQuery}(Y_{c_1}, x, t)$ 
    if  $r_2 \leq d(c_1, x) + s_d(x) + t$  and  $d(x, c_2) < t + r_3$ 
       $result = result \cup \text{rangeQuery}(Y_{c_2}, x, t)$ 
  return  $result$ 

```

Note that the internal and external radii r_2 and r_3 are predetermined for each Y_{c_2} . Also, just as for the other algorithms discussed, precomputed distances between centers can be used to approximate $d(c_1, x)$ and $d(c_2, x)$ and so potentially reduce the number of distance computations that must be made.

Also note that, for large $|Y_{c_2}|$, due to the randomness of $d(c_1, y)$ and $d(y, c_2) + s_d(y)$ for $y \in Y_{c_2}$, normally r_2 is relatively be small and r_3 relatively large, making pruning fairly ineffective. Just as for set space radius partitioning, this could perhaps be addressed by reversing the order of arguments in the partitioning criteria on alternate levels.

10.5 Searching hard spaces

It is noted in [22] that (in essence) the effectiveness of the metric space algorithms (that are discussed in section 10.3) is proportional to the variance of d . The distance function $d(x, x) = 0$, $d(x, y) = 1$ for all $x \neq y$, $x, y \in M$ is used to illustrate this effect. Quite clearly, no relative orderings or partitions

can be effective at decreasing the time complexity of ordering or partitioning search algorithms below $O(n)$ with this d .

Example 10.1. For all $x \neq y$, $x, y \in M$, $d(x, x) = 0$, $d(x, y) = 1$. Now an arbitrary c induces the ordering $d(c, c) = 0$, $d(c, y_1) = 1$, ..., $d(c, y_{n-1}) = 1$. With $t = 0.5$ and $x \neq c$, so $d(x, c) = d(c, x) = 1$. The search for any $y \in M$ where $d(x, y) \leq t$ begins at the first row where $d(c, y) \geq d(x, c) - t$ (the second row of the table) and continues until a row where $d(x, y) > d(c, x) + t$ is reached (this never happens, so the search continues until the end of the table).

In general, for spaces $\langle M, d \rangle$, as the variance of d decreases, relative orderings are less useful and partitions are less effective at dividing the space. Spaces $\langle M, d \rangle$ that are hard to search due to the relatively small variance of d are called **hard spaces** in [67].

10.5.1 Specialised algorithms for searching hard spaces

Because more general algorithms are not effective at searching hard spaces, it is sensible to consider specialised algorithms for particular hard spaces. Consider the space $\langle M, d \rangle$ where, for all $x \neq y$, $x, y \in M$, $d(x, x) = 0$, $d(x, y) = 1$. The general algorithms described above are not effective at searching this space however the following, rather obvious, specialised range search algorithm is very effective.

```
rangeQuery( $M, x, t$ )
  if  $t \geq 1$  return  $M$ 
  return  $\{x\}$ 
```

It is apparent that specialised algorithms such as this one, that have a separate conditional for each possible value of d , are often easier to develop when the variance of d is smallest. This is convenient as it is precisely these d that cause the most problems for the more general algorithms.

10.5.2 Specialised algorithms for searching n -dimensional spaces

It is also noted in [22] that (essentially) if points are “uniformly distributed” in n dimensions (for example $M = \{(x_1, \dots, x_n) | x_i \in \mathbb{N}_1 \leq 1000 \text{ for } 1 \leq i \leq n\}$) and $d((x_1, \dots, x_n), (y_1, \dots, y_n)) = (\sum_{r=1}^n (|x_r - y_r|)^p)^{\frac{1}{p}}$ then the variance of d decreases as n increases, so spaces with high “intrinsic dimensionality” are hard spaces.

Consider the class of n -dimensional spaces $\langle M, d \rangle$ where

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{r=1}^n d_r(x_r, y_r)^p \right)^{\frac{1}{p}}$$

for all $\mathbf{x} = (x_1, \dots, x_n), \mathbf{y} = (y_1, \dots, y_n) \in M$. An obvious specialised algorithm for finding all $\mathbf{y} \in M$ where $d(\mathbf{x}, \mathbf{y}) \leq t$ involves projecting the space onto a single dimension (the i^{th} dimension $\langle M_i, d_i \rangle$), selecting all $y_i \in M_i$ where $d_i(x_i, y_i) \leq t$ and then selecting each $\mathbf{y} \in M$, corresponding to a selected y_i , where $d(\mathbf{x}, \mathbf{y}) \leq t$. This algorithm relies on the inequality

$$d_i(x_i, y_i) \leq \left(\sum_{r=1}^n d_r(x_r, y_r)^p \right)^{\frac{1}{p}}$$

which holds if each d_r is non negative. This inequality ensures that the set of $\mathbf{y} \in M$, corresponding to a selected y_i , will be a superset of the required set. If desired, the intersection of the initial set of selected \mathbf{y} s with selections in all other dimensions can be taken, giving the set $Y = \{\mathbf{y} \in M | d_1(x_1, y_1) \leq t, \dots, d_n(x_n, y_n) \leq t\}$, from the final selection of \mathbf{y} s where $d(\mathbf{x}, \mathbf{y}) \leq t$ would be made. This Y is a minimal n dimensional hypercube containing the n -ball centered on x with radius t .

The problem with this algorithm is that, while the volume of the hypercube approaches infinity as $n \rightarrow \infty$, the volume of the n -ball approaches zero⁵. The ratio of ball volume to hypercube volume is small, even for relatively small n . As a result of this, the set Y is very large (for $n > 10$ say)

⁵[15] mistakenly asserts that the volume of an n -ball increases exponentially with n .

compared to the size of the set $\{y \in M | d(\mathbf{x}, \mathbf{y}) \leq t\}$, making this algorithm ineffective (see table 10.1 below).

n	ball volume over enclosing hypercube volume
1	1
2	0.7854
3	0.6981
4	0.3843
6	0.08075
8	0.01585
10	0.002490
20	0.0000002461
40	$10^{-20} * 0.3278$

Table 10.1: Ball to enclosing hypercube volume (4 s.f.) for different n in Euclidian space

The problem is worst for small p . As $p \rightarrow \infty$, the ratio gets larger, and decreases more slowly as n increases. However, $p = 1$ and $p = 2$ are common choices for p , so this will often not be helpful.

For a solution, the practical applications of range query is considered. Fortunately, Knowledge Library users should normally not require a large n when formulating a query. If this is the case, the problem can be simplified by projecting the space.

Example 10.2. $\langle M, d \rangle$ is an n -dimensional space where n is large. However typical users are only interested in a small number of dimensions, computing sets such as $\{\mathbf{y}^- \in M^- | d^-(\mathbf{x}, \mathbf{y}^-) \leq t\}$, where $\langle M^-, d^- \rangle$ is an m -dimensional space for some small m .

There are a number of things worth noting about this.

First, n -dimensional information spaces (for large n) are still useful, even if they are always projected onto m -dimensional spaces whenever they are searched, as users are able to choose the m dimensions in each projected space from any of the original n dimensions.

Second, restrictions on the number of dimensions that can be used to search a space should not be particularly onerous to users. As has already

been noted (see section 8.4), even a *very* simple 4 dimensional space with 10 points in each dimension has $10^4 = 10000$ distinct points (this number grows very fast as further points and dimensions are added), so projected spaces should be sufficiently descriptive.

Third, even when searching for information elements that are *similar* to an existing information element in a high dimensional space, it is normally reasonable to identify the dimensions that are important, and project the space onto these dimensions, before making comparisons.

Fourth, for many practical information spaces, specialised algorithms can be used to effectively increase the practical limit on m . Many of the dimensions of an m dimensional space may have “rapid” distance functions that can be computed in very little time (the standard assumption is that distance functions take some time to compute). If $A \subseteq \{1, \dots, n\}$, is the set of (indexes of) dimensions with rapid distance functions the inequality

$$\left(\sum_{r \in A} d_r(x_r, y_r)^p \right)^{\frac{1}{p}} \leq \left(\sum_{r \in \{1, \dots, n\}} d_r(x_r, y_r)^p \right)^{\frac{1}{p}},$$

that holds for non negative distance functions, can be used to combine all of the rapid dimensions into a single dimension (with the distance function $(\sum_{r \in A} d_r(x_r, y_r)^p)^{\frac{1}{p}}$), effectively reducing the number of dimensions in the search space. Dimensions with extremely fast (specialised) searching algorithms, such as the one in section 10.5.1, can also be utilised. It is also often possible to make use of information such as the weight associated with a dimension, and the average distance calculation time for dimensions, to prioritise dimensions.

10.6 What this Chapter Achieved

This chapter relates the problem of providing core Knowledge Library functionality to the literature on range, k -nearest neighbour, and ranked queries. Range query algorithms provide the basis for k -nearest neighbour, and ranked queries algorithms. Generalisations of the existing metric space partitioning

range query algorithms are proposed that function over set spaces. As information space, the mathematical basis for Knowledge Libraries is itself based on set space, these generalisations extend the existing algorithms towards the provision of core Knowledge Library functionality.

Chapter 11

Experimental Results and Discussion

11.1 Overview

This chapter presents the results of experiments conducted to determine the effectiveness of range query algorithms. A clearer understanding of the limitations of existing range query algorithms is developed, and a number of improvements are suggested.

Section 11.3 describes an experiment that compares and contrasts the time required to execute range queries for spatial partitioning algorithms (over symmetric and non symmetric spaces) and sequential search algorithms. The scope of sequential search, and the best statistics to use to compare range query algorithms is discussed.

Section 11.4 examines how the variance of distances in the space effects the performance of spatial partitioning algorithms. The **pruning hypothesis** is advanced to explain the poor performance of these algorithms when the variance of the space is low.

Section 11.5 discusses techniques to improve the performance of spatial partitioning algorithms. Two methods are discussed: center selection; and multiple search trees. Consistent with the pruning hypothesis, center selection does not significantly improve performance, while multiple search trees

do notably improve performance for “medium difficulty” spaces.

Section 11.6 examines the effectiveness of spatial partitioning over multi-dimensional spaces. Both the variance, and performance decrease as the number of dimensions increase. The multiple tree technique extends the effectiveness of these algorithms from 2 to 3-dimensional spaces.

Section 11.7 evaluates the effectiveness of (modified metric space) set space spatial partitioning. Non symmetric space partitioning was effective for the “Euclidean like” space $\langle \{1, \dots, 1000\}, x - y \rangle$. However it was not effective for randomly generated spaces.

Section 11.8 discusses the sequential search algorithm in greater detail. The performance of sequential search over multi-dimensional spaces and set spaces is evaluated. Sequential search is effective, and should be applicable to the majority of Knowledge Libraries.

Section 11.9 introduces the sequential-hybrid algorithm. This algorithm extends the effective scope of sequential search over n -dimensional spaces that include a dimension for which it is infeasible to precompute *distance-Matrix*. Essentially the $n - 1$ dimensions, for which *distanceMatrix* can be precomputed, are used to index the remaining dimension. This algorithm can be used to index high dimensional spaces (where spatial partitioning is ineffective).

Section 11.10 summarises and discusses the main findings of this chapter, while section 11.11 reports what this chapter has achieved, with respect to Knowledge Libraries.

11.2 Introduction

This chapter investigates range query algorithms. Following the notation developed in earlier chapters, a range query determines the set of points

$$Y = \{y \in M^I \mid d(x, y) \leq t\},$$

for a “query point” $x \in M$, where (M, d, \mathcal{I}) is an information space and $M^I \subseteq M$ is the set of points that have information units attached by the

index \mathcal{I} . In the literature, $\langle M, d \rangle$ is a metric space and each point in M^I has exactly one information unit attached. We are interested in generalising, so $\langle M, d \rangle$ can be a set space, and each point in M^I can have a number of information units attached.

Although the number of information units attached to each point is relevant when determining k -nearest neighbour queries, which are closely related to range queries, it is of no concern when determining range queries. Also, the required generalisation is easily implemented.

Metric spaces satisfy ΔI , are reflexive, strict positive and symmetric. However set spaces, which satisfy $G\Delta I$, are \subseteq -reflexive, $\not\subseteq^d$ -strict positive and lack the symmetric property, are the main focus of this research. The lack of the symmetric property presents the only substantive difficulty as far as adapting the existing metric space algorithms, so we focus solely on this.

For simplicity, the experiments in this chapter are limited to spaces where $M = M^I$. That is, to where each point in M has at least one information unit attached. More general cases can be obtained from the experimental ones either by adding further, unattached, points to M ; or by “detaching” points in M^I .

In practice, as the points in M^I are attached to information units, the cardinality of M^I would only exceed one hundred thousand in large information spaces. In contrast, as the points in M correspond to possible queries, the cardinality of M can be much larger, even theoretically infinite (as is discussed in section 8.4). As will be seen, this is important because, while it is often practical to precompute distances between points in M^I , it is often impractical to precompute distances between points in M .

11.3 Set Space Radius Partitioning Implementation: Non symmetric Experiments

The initial objective was to test the effectiveness of the chapter 10 set space radius partitioning range query algorithms on set spaces that lack the symmetric property.

11.3.1 Non Symmetric Experiment setup

For this experiment, a network (with n nodes and e edges) was generated. All edges in the network were assigned a weight of 1, and nodes a span of 0. The edges connected randomly selected pairs of nodes. A minimum number (1 in these experiments) of outward edges per node was specified (to contribute towards network connectivity). All nodes were also connected by “virtual” (uncounted) edges each with a weight of *maxDist*.

The network was used to generate a 2-dimensional array *distanceMatrix*, the cells of which contained distances, generated from the network following the method discussed in section 8.3. For efficient disk access, two copies of *distanceMatrix* were written to disk; one allowing entire rows to be loaded from disk in a single read operation; the other allowing entire columns to be loaded from disk in a single read operation. This enabled distances from a particular point, to all other points, and distances to a particular point from all other points, to be accessed as a block.

The constant *maxDist* was used to limit the range of distances so they could be stored in single byte `unsigned chars` (in C++ implementations). This relatively efficient use of memory allowed spaces with up to one hundred thousand points to be stored on the 40GB disk used for the experiment.

The *distanceMatrix* distances were essentially normally distributed (see the central limit theorem), with the exception of a single outlier at *maxDist*. Decreasing the number of (non virtual) edges (approaching n) increased the number of *maxDist* points (which decreases the variance), but otherwise increased the variance of the space. All this is well illustrated by figure 11.1.

Larger spaces required (proportionally) more edges to maintain a similar distribution. For example, 1000-point symmetric spaces with 1100 undirected edges had a similar distribution of distances to 10000-point spaces with 13000 edges and 100000-point spaces with 150000 edges.

Note that, 1000-point reflexive, strict positive spaces, such as those illustrated in figure 11.1, have exactly 1000 distances=0, which corresponds to the point (0,1000) in each graphic. Due to the scale of the graphics, this point may appear to be positioned at (0,0).

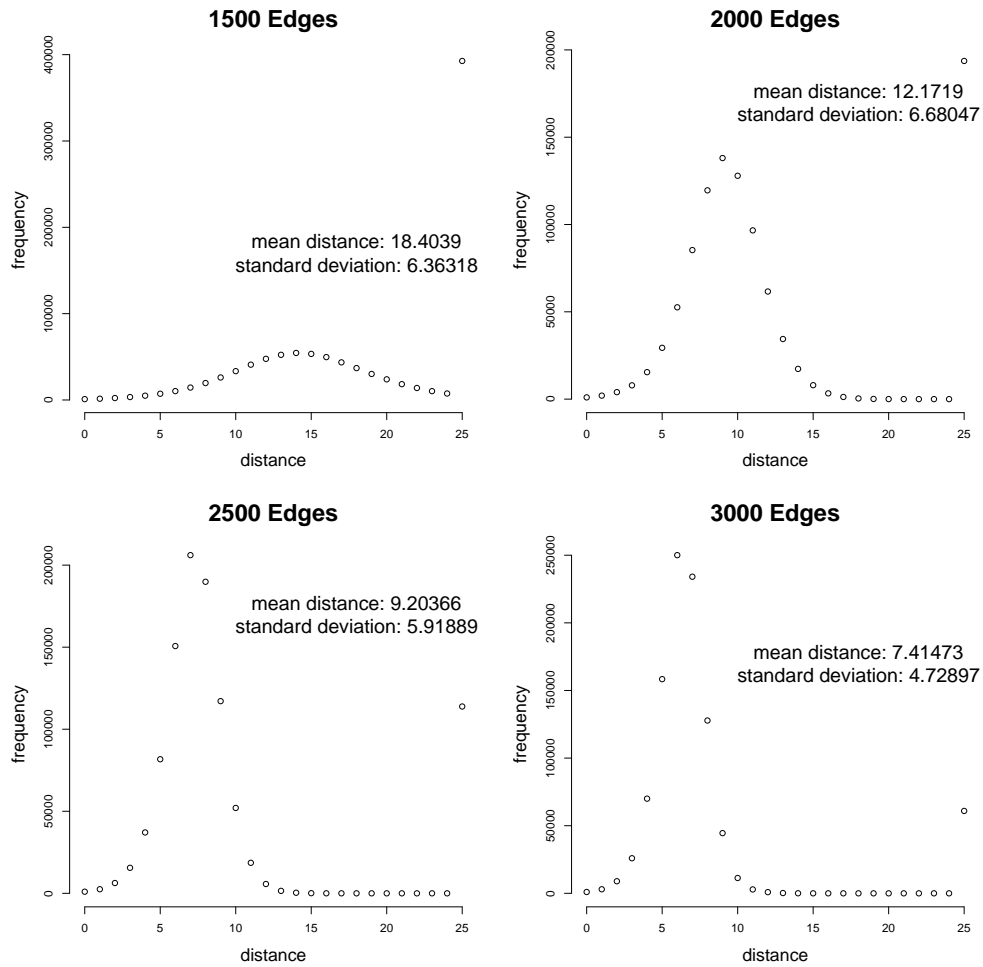


Figure 11.1: Frequency of distances for typical 1000-point network spaces with $maxDist = 25$ and 1500, 2000, 2500 and 3000 directed, $weight = 1$ edges.

The experiments required the implementation of the set space, radius partitioning search tree algorithm (without the partitioning criteria alternating modification) as discussed in section 10.4.2. The range query (tree search) algorithm discussed in this section was also implemented.

Importantly, both the tree generation and tree search implementations perform breadth first, rather than (the discussed) depth first tree traversals. This was done so that only a single row read operation, and a single column read operation, need be performed for each level of the tree (both when constructing the tree, and when searching it).

No special center selection algorithm was used, with centers being sequentially numbered points (from the first point). This is effectively random center selection as each network was random.

The sequential search range query algorithm discussed in section 10.2.5, and the (symmetric) metric space radius partitioning search tree generation and range query algorithms discussed in section 10.3.2 were also implemented.

The sequential search implementation utilised the existing *distanceMatrix* data structure, so all the distances (that is $d(x, y)$ for all $y \in M$), required for a single sequential search range query could be loaded from disk in a single read operation.

The metric space implementation was backward engineered from the set space implementation, the differences being that the edges in the initial network were undirected, and only one copy of *distanceMatrix* needed to be written to disk, as *distanceMatrix* columns and rows were identical due to the symmetry of the space.

The (symmetric) metric spaces were generated from networks with undirected edges, while the (non symmetric) set spaces were generated from networks with directed edges. For approximate parity, networks with directed edges were assigned twice the number of edges as corresponding networks with non directed edges.

Note that networks with $2m$ directed edges are not generally isomorphic to networks with m undirected edges (see figure 11.2), so the results are indicative rather than directly comparable. Also note that the sequential

search implementation is identical for symmetric and non symmetric spaces and is unaffected by the distribution of distances.

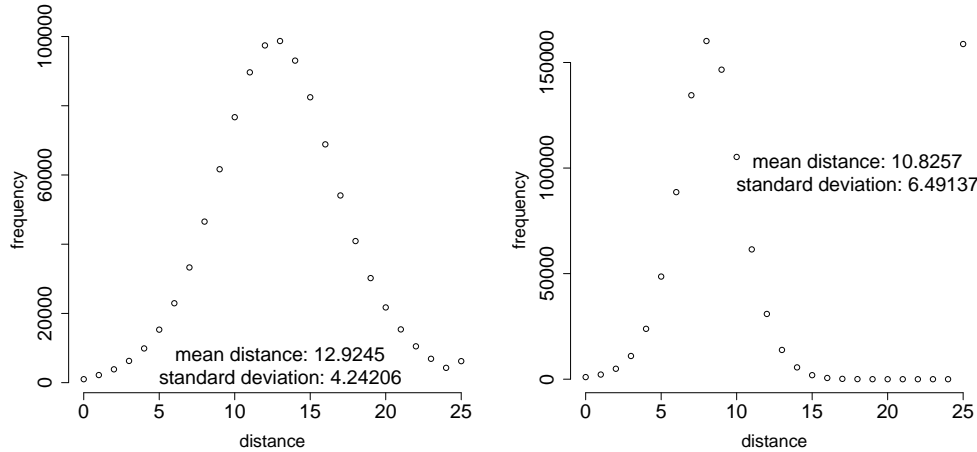


Figure 11.2: Frequency of distances for typical 1000-point network spaces with $maxDist = 25$. **LHS**: 1100 undirected, $weight = 1$ edges. **RHS**: 2200 directed, $weight = 1$ edges.

For each experiment, range queries were performed for one thousand different points and the average time to retrieve points, over these one thousand queries, was recorded. Due to time and disk size limits, none of the initial experiments involved spaces with more than one hundred thousand points.

11.3.2 Non Symmetric Experimental results

The results (see table 11.1) at first seemed encouraging. Changing the number of shells partitioning each node (and so the “branch factor” or “arity” of the search tree) did not have an appreciable impact on performance. If the leaf capacity was too small, sometimes the search trees failed to generate, as some small sets of points can be effectively indistinguishable. However increasing the leaf capacity of the trees did not increase the run time of the tree search implementation.

With $branchFactor = 2$ and $leafCapacity = 10$ (for this $leafCapacity$, tree generation was successful for at least 99% of the spaces tested), the non

space size (points)	tree search		sequential search
	non symmetric	symmetric	
1000	0.21	0.07	0.007
10000	2.02	0.92	0.04
100000	36.73	16.06	0.57

Table 11.1: Average, over 1000 distinct queries ($t = 2$, $0 \leq x < 1000$), radius partitioning tree search ($branchFactor = 2$, $leafCapacity = 10$) and sequential search range query times (milliseconds) for typical symmetric network spaces generated from (random) $n = 1000$, $e = 1100$; $n = 10000$, $e = 13000$; and $n = 100000$, $e = 150000$ (undirected) networks and non symmetric network spaces generated from (random) $n = 1000$, $e = 2200$; $n = 10000$, $e = 26000$; and $n = 100000$, $e = 300000$ (directed) networks.

symmetrical test space with one hundred thousand points required around 0.036 seconds to search (on the 1.8GHz machine used for the experiment) on average, for small radius ($t = 2$) range queries. The, similar, symmetrical test space required only around 0.016 seconds to search on average.

Note that, due to the, greater than 2GB file size, 64bit file offsets were required to store and access *distanceMatrix* for the 100000-point test spaces, whereas standard file offsets sufficed for the 10000 and 1000-point test spaces.

11.3.3 Discussion of non symmetric results

The test results suggest that the time complexity of both the symmetric and the non symmetric tree search algorithms is, at best, $O(n)$. Also, the sequential search range query implementation was extremely fast, searching spaces with ten thousand points in only 0.04 milliseconds and taking only 0.57 milliseconds to search spaces with one hundred thousand points.

It can be concluded from this that, for practically sized network spaces similar to those tested, despite having an $O(n)$ time complexity, sequential search appears to be far superior to radius partitioning.

The set space range query implementation, which was expected to be slightly less effective, gives comparable performance to the metric space implementation. However, while both implementations give acceptable per-

formance, neither looks efficient when compared with the sequential search range query implementation. Why would anyone want to use range query (or hyperplane partitioning) algorithms? Perhaps the answer to this question lies in the time and memory complexity, both $O(n^2)$, of the construction of *distanceMatrix*.

Example 11.1. A space with 10^5 points requires a *distanceMatrix* with 10^{10} cells. If each distance takes one hundredth of a second to compute, then it would take at least 10^8 seconds, which is 3.17 years, of computing time to construct this *distanceMatrix*.

For the randomly generated spaces used, range query and hyperplane partitioning, search trees can be readily constructed that never require more than 25 levels (a trees *branchFactor* and/or *leafCapacity*, rather than the number of its levels, can be increased to accommodate larger spaces). These search trees can work quite efficiently when only *distanceMatrix* rows (and columns for non symmetric spaces) corresponding to the centers used in tree construction have been precomputed. As a single center is used for each level, only 25 rows (and 25 columns) of *distanceMatrix* need be precomputed.

Example 11.2. Following example 11.1, a single row of *distanceMatrix* would take 16.67 minutes to compute. That works out as 6.94 hours to compute 25 rows. Considering this is a one off setup cost for a very large information space, this is quite reasonable.

While these examples may, at first, appear conclusive, two further points should be considered.

First, the 3.17 years mentioned in example 11.1 is a one off setup cost for a large information space. The algorithm is amenable to parallelism. Utilising 100 sequential processors, for example, would result in a hundred fold reduction in the execution time of this algorithm.

Second, the 3.17 years mentioned in example 11.1 only applies to quite large spaces that are constructed all at once, and does not apply to spaces that are built up incrementally. The time of 16.67 minutes required to add

a single row to *distanceMatrix*, and so a single point to a one hundred thousand point (symmetric) space might be more acceptable if distributed over a number of years of incremental build up. It should even be possible, in many cases, to distribute this computation so, for example, contributors of research papers compute these distances on their own computers as part of a paper submission process.

This discussion clarifies the reasons for preferring radius and hyperplane partitioning over sequential search range query implementations. If a *distanceMatrix* is precomputed, sequential search gives the best performance for range queries at the expense of quadratic time and memory complexity for the construction and storage of *distanceMatrix*. Sequential search range queries are very fast, but initial setup and point insertion is slow. In contrast, radius and hyperplane partitioning range query implementations, which don't require a *distanceMatrix*, can give acceptable query performance while also having fast initial setup and point insertion.

It is also important to note, as pointed out in the introduction to this chapter, that when the set M of query points is very large (much larger than the set $M^I \subseteq M$ of points with attached information elements) it is often not possible to precompute and store distances corresponding to pairs $M \times M$. If it is not possible to precompute *distanceMatrix*, sequential search can be very slow. In these cases, the spatial partitioning/tree search algorithms may offer faster query times.

For a large class of applications, it is practical to precompute *distanceMatrix* for distances corresponding to pairs $M^I \times M^I$, but not pairs $M \times M$. For presently conceived applications, it is practical to precompute distances corresponding to pairs (c, y) for particular center points $c \in M^I$ and all points $y \in M^I$. This is the minimum requirement for the construction of spatial partitioning search trees.

While it has been a useful experiment to compare query times, timed experiments can easily be misinterpreted. Also, query time experiments, run on different machines with different memory sizes, instruction sets, compilers, operating systems, processor speeds... are not comparable.

For given spaces and queries, the best statistic to use to compare tree

and sequential search algorithms is candidate set to space size (see the LHS of figure 11.3). This proportion must be reliably small for tree search to be considered to have an advantage over sequential search.

Similarly, the best statistic to use to compare different tree search algorithms is retrieved to candidate set size (see the RHS of figure 11.3). Ideally, this proportion is 1/1. The proportion is smaller for less effective algorithms.

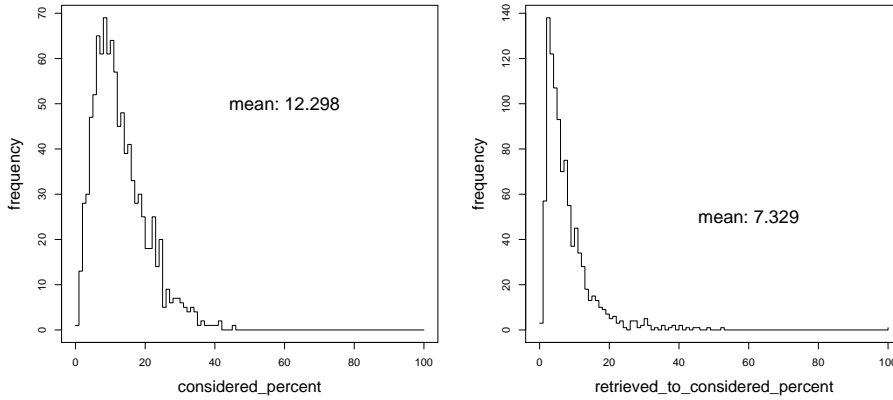


Figure 11.3: Data from a range query ($x = 999$, $t = 2$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) over 1000 different, 1000-point, metric (network) spaces generated from networks with 1100 $weight = 1$ undirected edges and $maxDist = 25$. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage).

11.4 Variance Experiments

The objective of the experiments in this section was to find out how the performance of the set and metric space tree search implementations could be improved and, at the same time, identify the reason for their relatively—when compared with sequential search—poor performance.

Pruning hypothesis. *The poor performance of the search trees tested is due to an insufficient number of nodes being pruned when searching, due the,*

too small, statistical dispersion of distances in the test spaces.

Consider the metric space, radius partitioning, node search inequality

$$r_{c_{i-1}} - t \leq d(x, c) < t + r_{c_i}$$

discussed in section 10.3.2. For spaces with standard deviations that are small relative to t , the mean distance is frequently relatively close to $r_{c_{i-1}}$, $d(x, c)$ and r_{c_i} , so the inequality would be frequently satisfied and few tree nodes would be pruned. Similar reasoning can be applied to the hyperplane partitioning node search inequality $d(x, c_i) - t < d(c_j, x) + t$ (see section 10.3.3).

Collision definition. *In the context of discussion of tree search algorithms, a **collision** occurs when the relevant search inequality is satisfied (and so a node is not pruned from the search tree). Less formally, a collision occurs when a query hyperball and a partitioning boundary intersect.*

Note that, while standard deviation should be a good approximate measure of the type of statistical dispersion that effects the likelihood of collisions, it is not exact. Collision prone spaces with large standard deviations are quite possible.

Example 11.3. A 1000-point space has 2 groups of 500 points. The standard deviation of distances between points in the same group is small, as are the distances. The standard deviation of distances between points in the different groups is also small, but the distances are large. Because the space has the same number of small and large distances, the standard deviation of all distances in the space is large. Even so, collisions will be likely after the first level of the space (which will partition points into the 2 groups).

Other measures of statistical dispersion, such as entropy, which all may give different scores for different partitions of the space, are similarly approximate.

11.4.1 Variance experimental results

Looking more closely at the behaviour of the tree search algorithm from the initial experiment (see tables 11.2 and 11.3), a reasonable number of search tree nodes were being pruned, but many of these nodes contained few, or even no points¹.

In order to test the pruning hypothesis, the variance of the test data was increased by generating network spaces from networks with randomly weighted edges. As single byte `unsigned chars` were used to store distances, the largest distance that could be represented was 255, so *maxDist* was set to 255 and uniform random edge weights between 0 and *maxDist/factor* were generated, where factor was experimentally determined and related to the size of the space. For symmetric spaces, the number of edges was set to $1.1n$. This produced distance distributions with reasonable variance, as illustrated in the left hand side of figure 11.4.

In order to retrieve a similar proportion of the space as earlier experiments, the query radius used in the, randomly weighted edges, experiment was increased from $t = 2$ to $t = 50$.

A 1-dimensional Euclidean space (that is, the distance function $d(x, y) = |x - y|$) was also implemented. Figure 11.4 also illustrates the distribution of distances from a uniform 1000-point 1-dimensional Euclidean space. Because distances can be quite readily computed directly from points, Euclidean spaces do not require the generation of a *distanceMatrix*.

Note the outlier at (0,1000) in both (figure 11.4) plots, corresponding to $d(x, x) = 0$ distances.

11.4.2 Random edge weight experimental results

Comparing the experimental results (see figure 11.5) for range queries on radius partitioning search trees over network spaces with randomly weighted edges, to the earlier results for *weight* = 1 edges (see figure 11.3), two things

¹For trees with *branchFactor* = 2, for example, nodes Y are divided by finding the median $d(c, y)$ for all points $y \in Y$. The LHS child of Y will contain points y where $d(c, y) \leq$ this median distance, this could be *all* points $y \in Y$, in which case the RHS child of Y will be empty.

level	center	internal nodes	leaf nodes
0	0	2	0
1	1	4	0
2	2	8	0
3	3	16	0
4	4	32	0
5	5	47	17
6	6	27	67
7	7	5	49
8	8	1	9
9	9	0	2

Table 11.2: Radius partitioning search tree ($branchFactor = 2$ and $leafCapacity = 10$) for a typical, random, 1000-point network space (with 1100 undirected $weight = 1$ edges).

level	candidate nodes	collisions	pruned nodes	remaining nodes	pruned points	considered points	retrieved points
0	2	1	0	2	0	0	0
1	4	1	1	3	281	0	0
2	6	1	2	4	208	0	0
3	8	3	1	7	40	0	0
4	14	3	4	10	166	0	0
5	20	7	3	17	52	0	0
6	28	11	5	23	41	23	0
7	6	15	1	5	9	144	7
8	2	0	1	1	5	22	0
9	0	0	0	0	0	9	0

Table 11.3: Candidate nodes, collisions, pruned nodes and points, considered and retrieved points by level from a range query ($x = 999, t = 2$) on the radius partitioning search tree in table 11.2. The retrieved point set contained 7 points, while the candidate point set contained 198 points, giving a retrieved to considered ratio of approximately 3%.

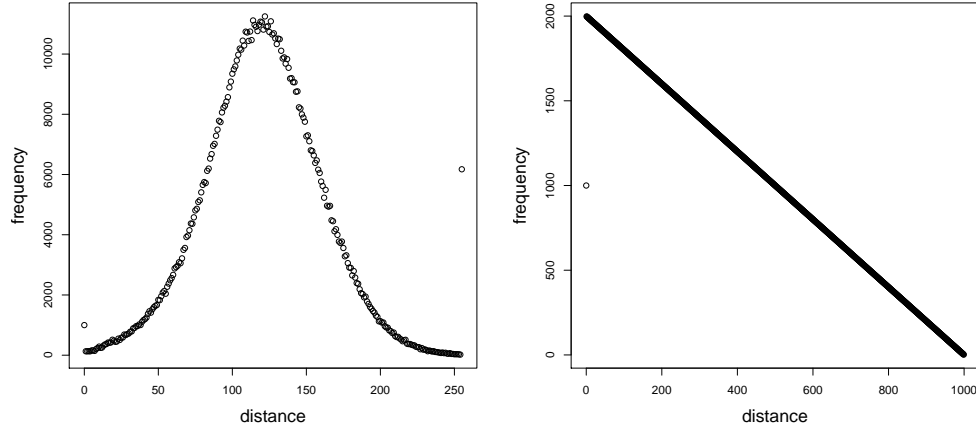


Figure 11.4: Frequency of distances. **LHS**: a typical 1000-point network space with $maxDist = 255$ and $1.1n$ undirected edges with uniform random weights (integers 1 to 19). Mean distance: 120.913, standard deviation: 39.2604. **RHS**: a 1-dimensional Euclidean space over integers 0–999. Mean distance: 333.333, standard deviation: 235.702.

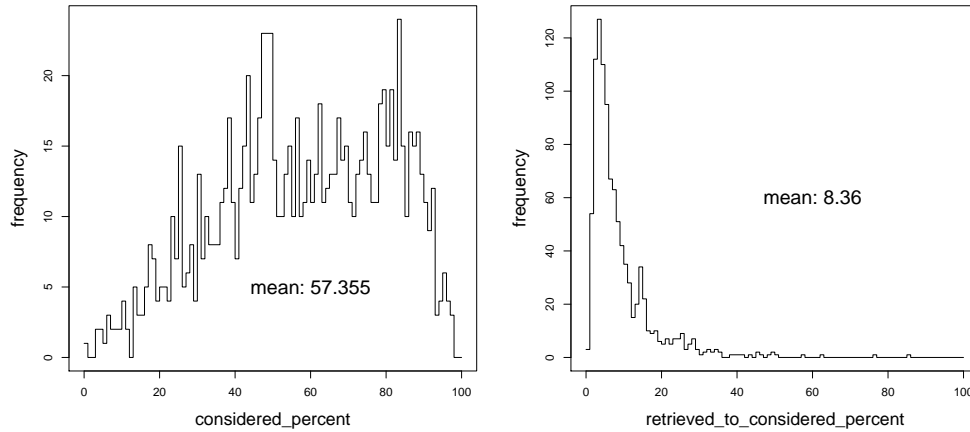


Figure 11.5: Data from a range query ($x = 999$, $t = 50$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) over 1000 different, 1000-point, metric (network) spaces (with 1100 randomly weighted undirected edges). **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage). Compare with figure 11.3.

are immediately apparent. First, the size of candidate point sets, though also quite variable, tends to be larger (and so worse) for the later experiment. Second, the retrieved to considered percent in the later experiment is similar to the earlier. This, seeming contradiction, indicates that the queries (with $t = 50$) in the later experiment tended to retrieve a greater proportion of the space than the queries (with $t = 2$) in the earlier experiment.

The lack of significant improvement is explained by the fact that, although the variance of the space has been increased, the radius of the search queries have been increased to match (so the queries in both experiments retrieve a similar proportion of their respective spaces).

11.4.3 Euclidean space experimental results

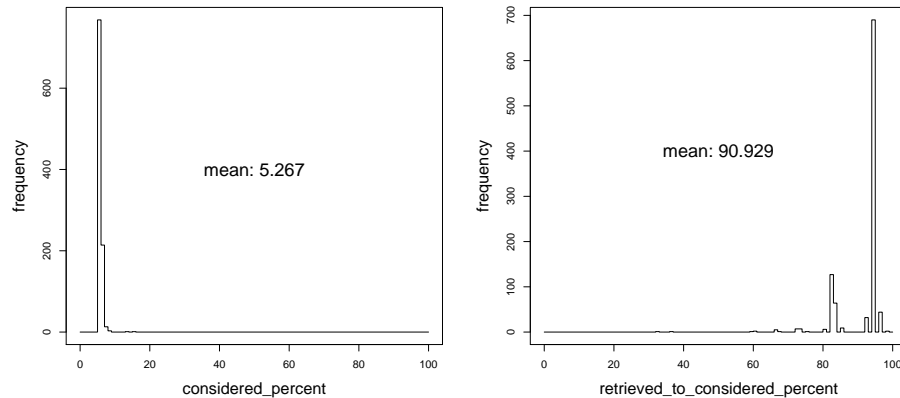


Figure 11.6: Data from the range query $x = 999$, $t = 50$ on 1000 radius partitioning search trees (with randomly selected centers, *branchFactor* = 2 and *leafCapacity* = 10) over a 1000-point uniform Euclidean space. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage). Compare with figure 11.3.

The Euclidean space experimental results (see figure 11.6) are distinct from the network space results in two main ways. First, there is little variation in size of candidate point sets, with all test results between 5% and

15%, and 99.8% of results between 5% and 8%. Second, the retrieved to considered test results are consistently high, with distinct peaks at 94% and 82%, and with 97% of results above 80%. Note that the peak at 94% can be moved to 100% by decreasing *leafCapacity* to 1.

It may be that the characteristic “peak-trough” pattern of retrieved-to-considered scores (see RHS figure 11.6) can be attributed to collisions occurring in the search tree. Noting that collisions are more damaging the closer to the root of the search tree they occur, it may be that, for example, the peak in frequencies at 94% corresponds to queries that are conducted without serious collisions and the peak at 82% corresponds to queries that are conducted with a single serious collision.

Similar reasoning can be applied to interpret earlier test results. Looking at the retrieved-to-considered results in figure 11.5 for example, the peak at 3% corresponds to queries that have been conducted with numerous serious collisions. The, much smaller, peak at 14% corresponds to queries that have been conducted with fewer collisions.

This reasoning is borne-out by more detailed, instance-by-instance examination, as illustrated in tables 11.4 and 11.5 (and the earlier tables 11.2 and 11.3)

11.5 Center Selection and Multiple Tree Experiments

Although the metric space radius partitioning algorithm created effective indexes for the 1-dimensional Euclidean spaces tested, radius partitioning appears to be far from effective when applied to randomly generated (network) metric spaces. It now seems likely that the “search tree” approach generally (including radius and hyperplane partitioning) is *not* an effective method of indexing spaces where typical query radii are large relative to the standard deviation of distances. However, possible improvements to the algorithm must be examined before firm conclusions can be drawn.

Center selection is the main technique for improving the performance of

level	center	internal nodes	leaf nodes
0	282	2	0
1	206	4	0
2	672	8	0
3	838	16	0
4	971	32	0
5	226	64	0
6	909	128	0
7	593	256	0
8	691	487	25
9	542	1	973
10	717	0	2

Table 11.4: Typical radius partitioning search tree (with random centers, $branchFactor = 2$ and $leafCapacity = 1$) for a 1-dimensional Euclidean space over integers 0–999. Compare with table 11.2.

level	candidate nodes	collisions	pruned nodes	remaining nodes	pruned points	considered points	retrieved points
0	2	0	1	1	501	0	0
1	2	0	1	1	250	0	0
2	2	0	1	1	125	0	0
3	2	0	1	1	62	0	0
4	2	1	0	2	0	0	0
5	4	2	0	4	0	0	0
6	8	3	1	7	8	0	0
7	14	7	0	14	0	0	0
8	28	13	1	27	2	0	0
9	50	24	1	49	1	2	2
10	0	0	0	0	0	49	49

Table 11.5: Collisions, pruned nodes and points, considered and retrieved points by level from the range query $x = 999$, $t = 50$ on the radius partitioning search tree in table 11.4. Both retrieved and candidate point sets contained 51 points, giving a retrieved to considered ratio of 100%. Compare with table 11.3.

search tree algorithms. The motivation for this idea is that “better” centers result in better partitions and so more effective search trees. Various algorithms have been used to select centers including selecting centers that are far apart[4].

Clearly (for symmetric spaces) centers that are close together will generate similar partitions, so centers that are not close together should be preferred.

11.5.1 Greatest minimum center selection experiment

The following algorithm was used to select centers that are far apart.

1. randomly select *numOfCenters* trial centers;
2. repeat this *numOfTrials* times, producing a set with *numOfTrials* “trial centers” sets, each containing *numOfCenters* centers;
3. select the trial centers set which has the greatest minimum distance between its centers.

This algorithm was used to help construct search trees “with specially selected centers”. The results of this experiment, which was conducted on 1000 different, 1000-point, metric (network) spaces (with 1100 *weight* = 1 undirected edges), is presented in figure 11.7.

Comparing figure 11.7 with figure 11.3, it is apparent that this center selection technique has not significantly improved performance.

11.5.2 Standard deviation center selection experiment

Another experiment was conducted with selecting sets of centers c where the variance of $d(c, y)$ for all y , is high. Part of the motivation behind this center selection technique is illustrated in the example below.

Example 11.4. For the randomly generated network spaces introduced in section 2, “virtual” edges with *weights*=*maxDist* connect all nodes/points,

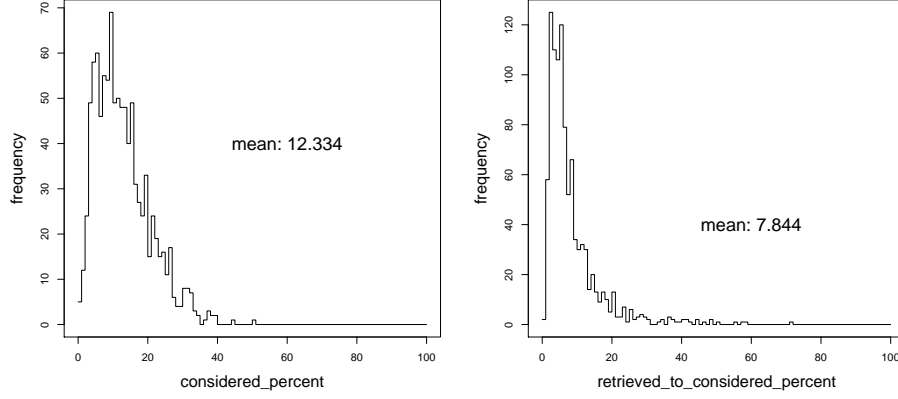


Figure 11.7: Data from a range query ($x = 999$, $t = 2$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) with specially selected centers over 1000 different, 1000-point, metric (network) spaces (with 1100 $weight = 1$ undirected edges). **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage). Compare with figure 11.3.

so if a point c is not otherwise connected to y , $d(c, y) = maxDist$. An (un-connected) point c where $d(c, y) = maxDist$ for all y , so the variance of $d(c, y)$ is 0, would *not* be a good choice for a center, as a single partition would contain all the points.

The algorithm used to select centers by standard deviation also incorporated greatest minimum selection.

1. randomly select $numOfCenters + numOfTrials$ trial centers;
2. select the $numOfCenters$ centers c with where $d(c, y)$ for all y has the largest standard deviation;
3. repeat this $numOfTrials$ times, producing a set with $numOfTrials$ “trial centers” sets, each containing $numOfCenters$ centers;
4. select the trial centers set which has the greatest minimum distance between its centers.

However the “standard deviation center selection experiment” produced results very similar to the greatest minimum experiment (see figure 11.7).

11.5.3 Discussion of center selection experiments

The results from these center selection experiments are broadly consistent with the literature [4, 6, 22, 44, 85, 103], which reports only marginal improvement in range query algorithm performance when center selection is used. The results are also consistent with the pruning hypothesis—center selection should not significantly improve performance as it is not possible to know (without knowing what queries will be performed) what centers will cause collisions.

The center selection algorithms we used involved numerous distance computations. In practical situations where search tree algorithms may have advantages over sequential search algorithms, it is not realistic to compute the standard deviation for each candidate center (as in section 11.5.2). Simpler center selection algorithms, such as maximising the minimum distance between centers (as in section 11.5.1) are more realistic.

11.5.4 Experiment with multiple search trees

In comparison to center selection, collisions may be more effectively reduced, and so performance improved, by *constructing multiple search trees and, for each query, completing the query on trees where serious collisions do not occur*. A similar idea involves, for each query, completing the query on all trees and taking the intersection of the resulting set of candidate point sets.

This, candidate set intersection, idea was implemented. Three radius partitioning search trees (with random centers, *branchFactor* = 2 and *leafCapacity* = 10) were constructed for each of 1000 different, 1000-point, metric (network) spaces (with 1100 *weight* = 1 undirected edges). A single range query ($x = 999$, $t = 2$) was executed on each of the search trees. The intersection of the three resulting candidate point sets was taken as the candidate point set for this search method. Detailed results for the first 10

spaces tested are presented in table 11.6, while the results for all 1000 spaces are summarised in figure 11.8.

space	candidate set 1 size	candidate set 2 size	candidate set 3 size	intersection set size	retrieved set size
0	179	248	52	18	3
1	134	189	242	45	7
2	41	74	158	10	3
3	83	178	302	37	12
4	100	90	184	4	2
5	158	81	111	9	5
6	388	213	366	92	3
7	60	66	30	13	6
8	117	30	166	11	8
9	222	184	73	22	10

Table 11.6: Candidate set sizes for the range query $x = 999$, $t = 2$ over 3 radius partitioning search trees (with random centers, *branchFactor* = 2 and *leafCapacity* = 10) for each of 10 different, 1000-point, metric (network) spaces (with 1100 *weight* = 1 undirected edges). The size of the intersection of these 3 sets, and the size of the retrieved set is also displayed.

Comparing the sizes of the candidate and intersection sets for each space (in table 11.6), the decision to take the intersection, rather than just the smallest candidate set, appears to be vindicated. Even when the (about 20) extra distance computations for the additional two search trees are taken into account, significantly fewer distance computations need to be made (for the spaces and range queries tested). Even just choosing the smallest, rather than the intersection, of the three sets would result in a significant reduction in distance computations.

Efficient implementations of this idea would maintain *ordered* lists of points in each leaf node. The candidate sets resulting from range queries could then be ordered by a single *merge* (as in the mergesort algorithm). The intersection of the resulting ordered candidate sets could then be efficiently computed.

Comparing figure 11.8 with figures 11.3 and 11.7, the improvement is dramatic. This strongly supports the pruning hypothesis (see section 11.4).

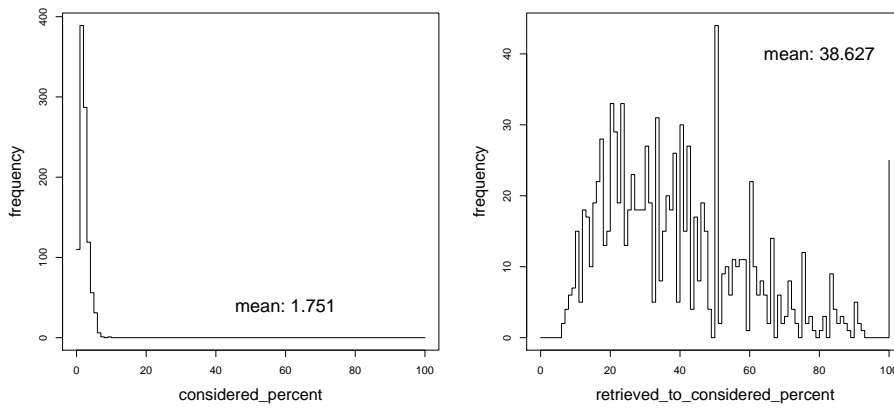


Figure 11.8: Data for the range query $x = 999$, $t = 2$ over 3 radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for each of 1000 different, 1000-point, metric (network) spaces (with 1100 $weight = 1$ undirected edges). The intersection of the 3 resulting candidate point sets was taken as the candidate point set for this search method. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage).

11.6 Experiments with Multi-Dimensional Spaces

The main objective of these experiments was to determine if the pruning hypothesis explains the performance of tree search algorithms over multi-dimensional spaces, or if a less general hypothesis is required, just for multi-dimensional spaces.

While examining multi-dimensional spaces, it was also useful to determine if multiple search trees are effective at improving the performance of tree search algorithms over multi-dimensional spaces.

11.6.1 Multi-Dimensional experiment setup

Numbered points in uniform Euclidean spaces were generated.

Example 11.5. To generate a 1000-point (numbered 0-999) 3-dimensional Euclidean space. First determine the number of coordinates in each dimension. This is $\dimNum = \lceil \sqrt[3]{1000} \rceil$. Now generate points of the form (x, y, z) from numbers $0 \leq i < 999$.

$$x = (i / \dimNum^2) \bmod \dimNum,$$

$$y = (i / \dimNum) \bmod \dimNum,$$

$$z = i \bmod \dimNum.$$

In this way the available points are uniformly “packed” into the space. The distance between points (x_1, y_1, z_1) , (x_2, y_2, z_2) is given by

$$d((x_1, y_1, z_1), (x_2, y_2, z_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}.$$

11.6.2 Multi-Dimensional experimental results and discussion

The distribution of distances for 1-dimensional Euclidean space has already been illustrated in the RHS of figure 11.4. The distribution of distances for 2,3,4 and 5-dimensional Euclidean spaces, generated as outlined in the above example, is illustrated in figure 11.9. As can be seen from the figure, the standard deviation of distances decreases as the number of dimensions increases.

1000 radius partitioning search trees (with random centers) were constructed for each of these spaces, and the range query $x = 0, t = 2$ was executed over each tree. The resulting distribution of retrieved to candidate set size is illustrated in figure 11.10. As can be seen, the mean retrieved to candidate set size decreases as the number of dimensions increases. In other words, the effectiveness of the algorithm decreases as the standard deviation of the space decreases—a result entirely consistent with the pruning hypothesis.

11.6.3 Multiple tree experiments

A further sequence of experiments was conducted to check the effectiveness of the multiple search tree approach when applied to multi-dimensional spaces. For each result, three search trees (with random centers) were generated and the query $x = 0, t = 2$ was executed over each tree. The resulting candidate set was taken to be the intersection of the candidate sets corresponding to each of the three trees. This was repeated 1000 times. The resulting distribution of retrieved to candidate set size is illustrated in figure 11.11.

As can be seen by comparing figure 11.10 to figure 11.11, the multi-tree technique was most successful when applied to the 3-dimensional space. Curiously, the resulting mean retrieved to considered percent was even slightly higher in the 3-dimensional space than in the 2-dimensional space!

The improvement for the 4 and 5-dimensional spaces is less dramatic. It would appear that the multi-tree technique is not overly effective in spaces

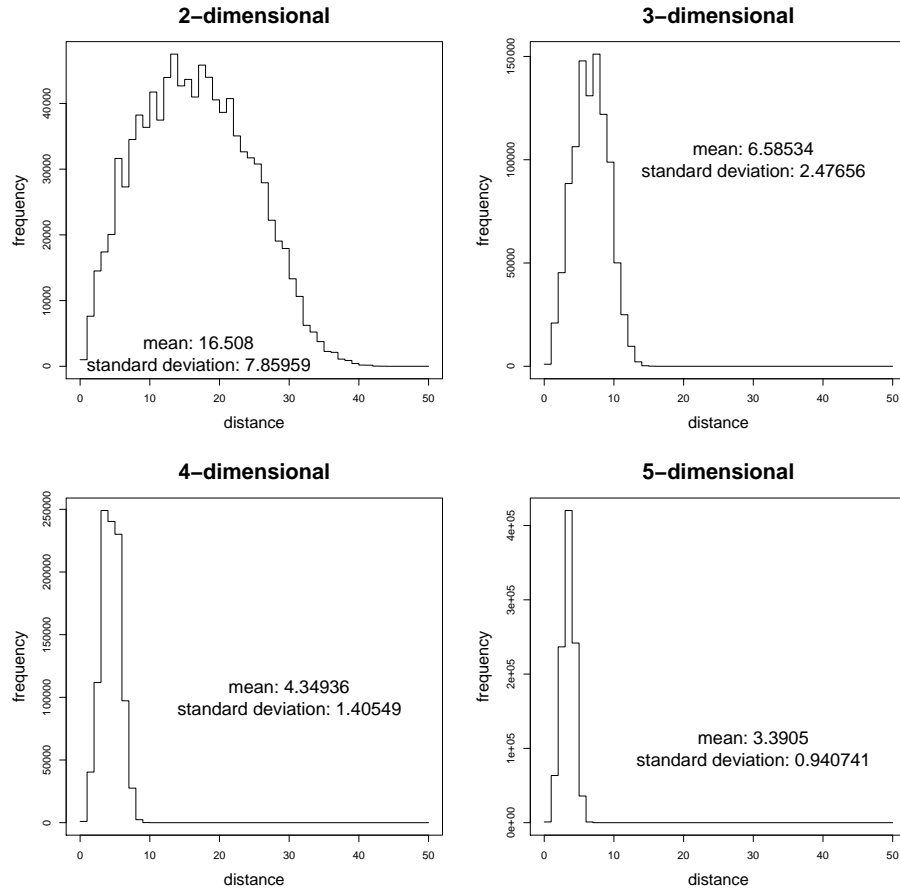


Figure 11.9: Distance frequencies for the “first” 1000 points in 2,3,4 and 5-dimensional uniform Euclidean spaces. Distances are truncated in the plot, but not when computing the mean and standard deviation. The 2-dimensional space has 32 coordinates each dimension. The others have 10, 6 and 4 (respectively).

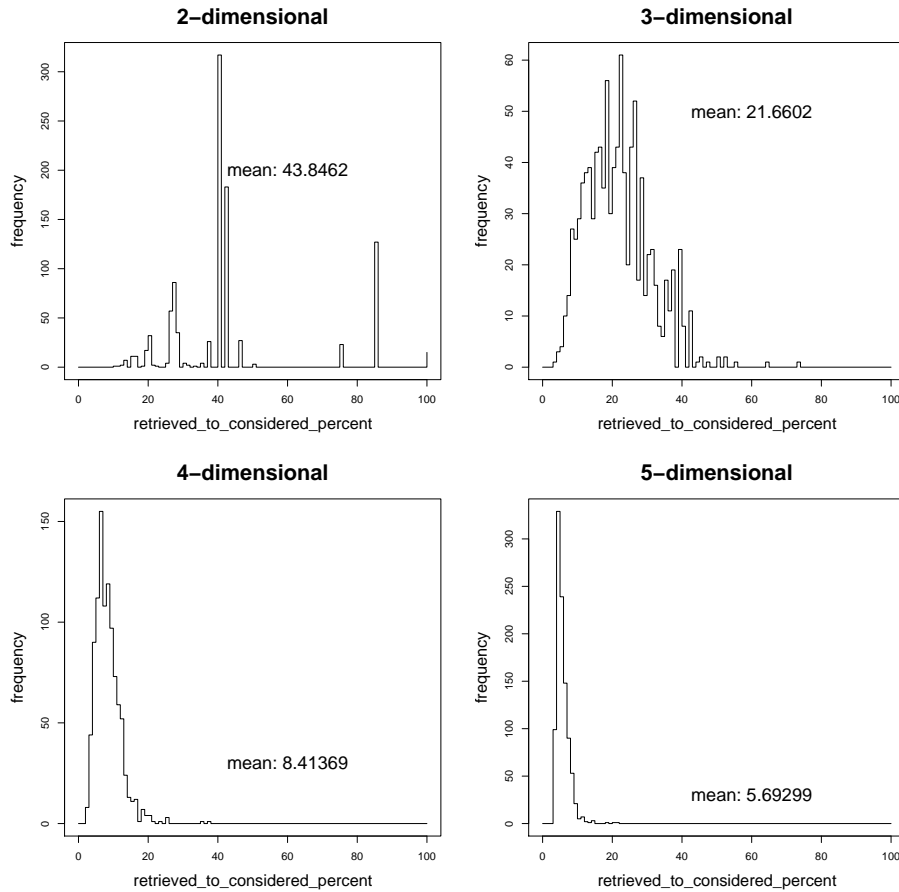


Figure 11.10: Distribution of retrieved to candidate point set sizes (by truncated percentage) for the range query $x = 0$, $t = 2$ over 1000 radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for uniform 1000-point, 2,3,4 and 5-dimensional Euclidean space.

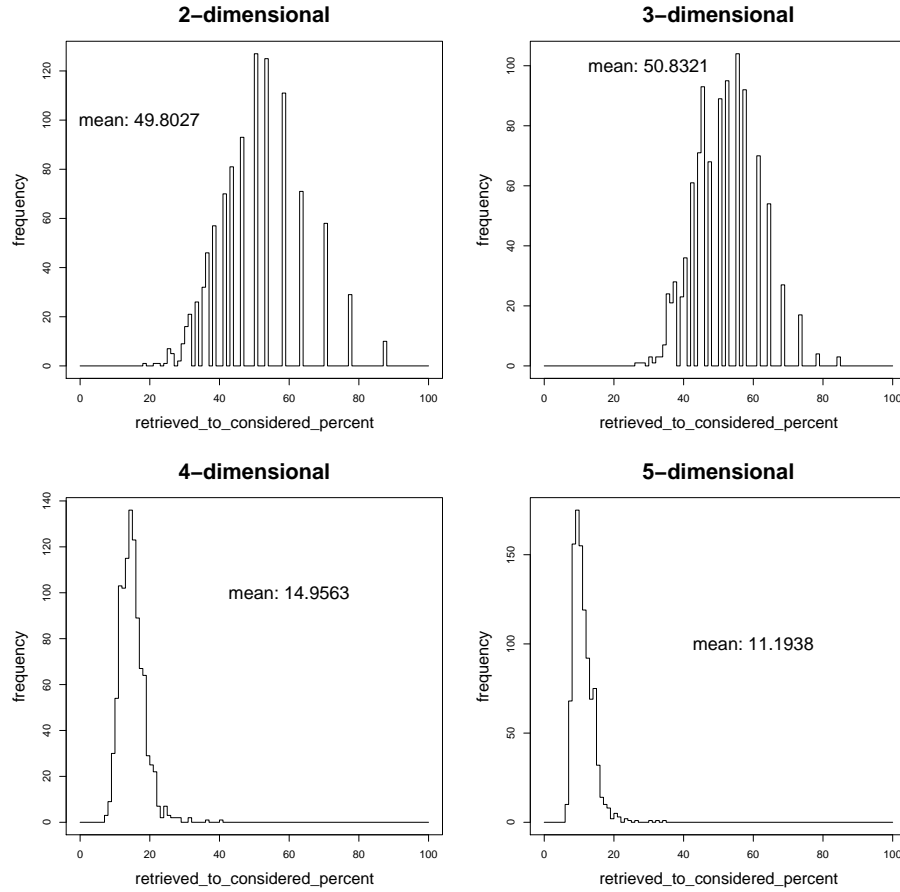


Figure 11.11: Distribution of retrieved to candidate set sizes (by truncated percentage) for the range query $x = 0$, $t = 2$ over 1000 different groups of three radius partitioning search trees (with random centers, $branchFactor = 2$ and $leafCapacity = 10$) for a uniform 1000-point, 2,3,4 and 5-dimensional Euclidean space. The group candidate set is the intersection of the candidate sets corresponding to each of the three trees.

where collisions are highly likely due to the relatively small standard deviation of the space, as all trees are similarly collision prone.

11.7 Set Space Experiments

The set space radius partitioning algorithm discussed in section 10.4.2 was implemented² and tested on the non symmetric space $\langle\{0, \dots, 999\}, x - y\rangle$. The results of this experiment are summarised in figure 11.12.

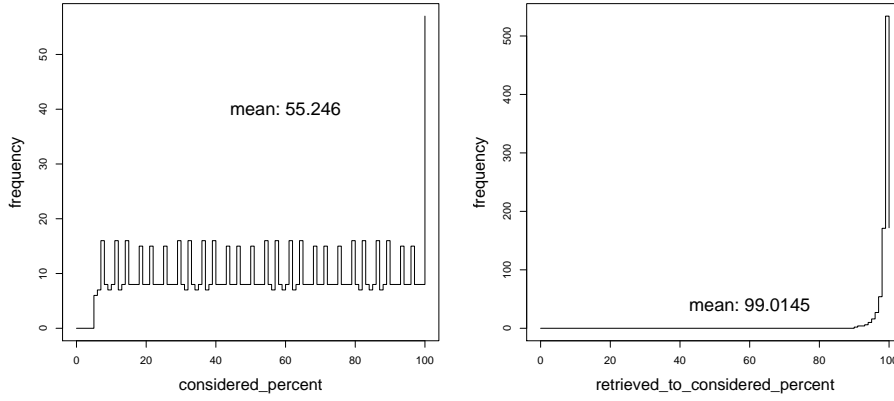


Figure 11.12: Data from 1000 range queries ($0 \leq x \leq 999$, $t = 50$) on 1000 different radius partitioning search trees (with random centers, $branchFactor = 2$, $leafCapacity = 10$) over the space $\langle\{0, \dots, 999\}, x - y\rangle$. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage).

The algorithm (both with and without the criteria alternating modification) were also tested on randomly generated non symmetric spaces. The results (without criteria alternation) summarised in figure 11.13, clearly show the algorithm is not effective for these spaces. The criteria alternating modification did not significantly improve the results.

²Without the criteria alternating modification, and with breadth first, rather than depth first, traversal.

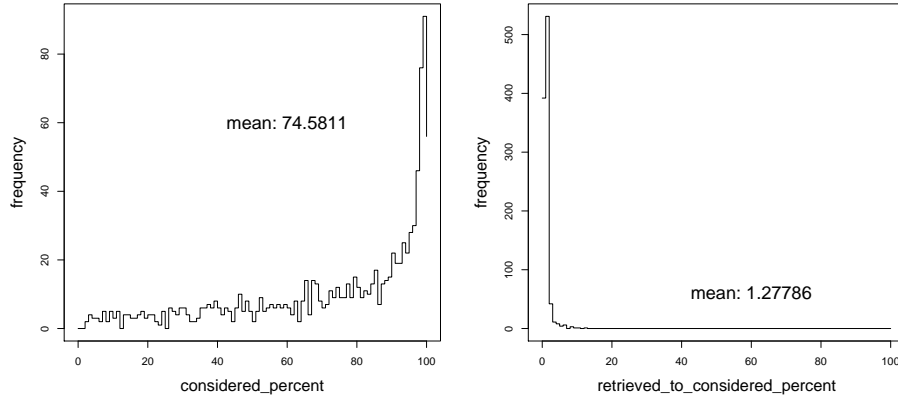


Figure 11.13: Data from a range query ($x = 999$, $t = 2$) on radius partitioning search trees ($branchFactor = 2$, $leafCapacity = 10$) over 1000 different, 1000-point, (non symmetric) network spaces generated from networks with 2200 $weight = 1$ directed edges and $maxDist = 25$. **LHS**: distribution of candidate point set size as a (truncated) percentage of space size. **RHS**: distribution of retrieved to candidate point set sizes (by truncated percentage).

11.7.1 Discussion of set space results

Just as for the metric space range query algorithm tested, the effectiveness of the set space range query algorithm varies widely. The algorithm appears to be highly effective over the “Euclidean like” space $\langle \{0, \dots, 999\}, x - y \rangle$ (with relatively large variance), but ineffective over the randomly generated spaces (with relatively small variance).

While the alternating criteria modification did not significantly improve the effectiveness of the algorithm over the spaces tested, it may make a more noticeable difference over other spaces.

11.8 Sequential search algorithms

While the effectiveness of spatial partitioning varies markedly for different spaces, the effectiveness of the sequential search algorithm is limited by the time and memory required to compute and store *distanceMatrix*.

As has already been discussed (see section 8.4), the number of coordinates in each dimension of even very large multi-dimensional spaces need not be great. As a separate *distanceMatrix* can be constructed for each dimension, the time and memory required to set-up sequential search over such spaces is not great.

Example 11.6. A 5-dimensional space with 1000 coordinates in each dimension has $1000^5 = 10^{15}$ points. If a separate *distanceMatrix* is constructed for each dimension, only $5 * 1000^2$ (non symmetric) distances need be computed. If each distance requires 0.01s to compute, less than 14 hours of computing time is required in total. If each distance is stored on one byte, only 5MB memory is required.

Similarly, if the points in a space are sets of points (in an underlying space) and the distance function between these points is a set distance function based on distances in the underlying space (such as the maximum, minimum distance between points) the *distanceMatrix* for the underlying space can be used to determine distances in (very large) set spaces. However, as the time complexity of set distance computation between (set) points (each consisting of n points from the underlying space) is $O(n^2)$, practical limits must be placed on the size of (set) points. This limits the size of the resulting set spaces.

Example 11.7. The points in a space are sets of points from an underlying, 1000-point space. Including the emptyset, there are $1 + \sum_{i=1}^{1000} i = 500501$ sets in total available. If the (set) points are limited to sets with cardinality ≤ 10 , the (set) space will have $1 + \sum_{i=991}^{1000} i = 9956$.

Two experiments were conducted to determine the effectiveness of sequential search. In the first experiment, the effectiveness of sequential search over n -dimensional spaces was determined. In the second experiment, set spaces with set distance functions were tested.

11.8.1 Sequential search for n -dimensional spaces

A number of n -dimensional information spaces (each with a different value of n) with 1000 points in each dimension, and 10^6 information elements were tested. In the tested spaces, the equation used to combine the n “dimensional distances” d_1, \dots, d_n was $\sqrt{(\sum_{i=1}^n d_i^2)}$. The time taken to conduct range queries (that is, the determination of $n * 10^6$ powers of 2 and 10^6 square roots)³ was recorded in each case. The results of this experiment is presented in table 11.7. Note that “considered to retrieved” ratios are *not* used to measure the effectiveness of sequential search as *all* information elements are checked. Also (unlike spatial partitioning algorithms) the query time for sequential search does *not* vary for different queries.

n	space size (points)	query time (seconds)
2	10^5	0.22
4	10^7	0.4
6	10^9	0.58
8	10^{11}	0.76
10	10^{12}	0.94

Table 11.7: Space size and query time for the sequential search algorithm (for a C++ implementation, using the `cmath sqrt()` and `pow()` functions, on a 1.8GHz machine with 265k memory running Linux) for various n -dimensional spaces with 1000 points in each dimension and 10^6 information elements. “Dimensional distances” d_1, \dots, d_n are combined using $\sqrt{(\sum_{i=1}^n d_i^2)}$.

11.8.2 Sequential search over set spaces with set distance functions

An experiment was conducted to determine the practical limit of set size for (set) points in realistic set spaces. The experiment tested a 3-dimensional space. Each dimension is a set space, based on an underlying space with 1000 points. The algorithm determines distances for 10^6 information units,

³For a C++ implementation (using the `cmath pow()` function) on a 1.8GHz machine with 256k memory running Linux.

attached to random points in the space. The results of this experiment is presented in table 11.8.

point size limit	space size (points)	query time (seconds)
2	1999 ³	3.37
4	3994 ³	3.92
6	5985 ³	4.94
8	7972 ³	6.32
10	9955 ³	8.1

Table 11.8: Space size and query time for the sequential search algorithm (for a C++ implementation, using the `cmath` `sqrt()` and `pow()` functions, on a 1.8GHz machine with 265k memory running Linux) for a 3-dimensional space. Each dimension is a set space, based on an underlying space with 1000 points. The algorithm determines distances for 10^6 information units, attached to random points in the space. “Dimensional distances” d_1, d_2, d_3 are combined using $\sqrt{d_1^2 + d_2^2 + d_3^2}$.

11.8.3 Discussion of sequential search results

The results presented in table 11.7 underscore the earlier sequential search results from table 11.1. Sequential search, where *distanceMatrix* is pre-computed, is a highly effective range query algorithm. Because individual dimensions of large spaces need not contain many points, *distanceMatrix* can often be readily precomputed for each of the n dimensions of large n -dimensional spaces. The resulting sequential search algorithms perform well.

Sequential search over set spaces, using set distance formulae to compute distances based on underlying *distanceMatrix* distances, is less effective. The size of (set) points needs to be limited to maintain efficiency. Nevertheless, an 8 second wait (on an old computer) to search through 1 million information elements (each attached to up to 10 points) in a 10-dimensional space is not unreasonable. The vast majority of queries should be far simpler (and require less time to compute) than this.

11.9 Introducing the Sequential-Hybrid Algorithm

The sequential search algorithm appears to be quite effective where it is possible to precompute *distanceMatrix* for each dimension of an n -dimensional space. However, if a single one of these n dimensions has too many points (perhaps an infinite number) and is *not* based on another space, *distanceMatrix* for this dimension cannot be precomputed. Because, for nonnegative d_i ,

$$\left(\sum_{i=1}^{n-1} d_i^p \right)^{\frac{1}{p}} \leq \left(\sum_{i=1}^n d_i^p \right)^{\frac{1}{p}},$$

the distances in the $n - 1$ spaces with a *distanceMatrix* can be used to give a lower bound for distances in the n -dimensional space. In this way $\left(\sum_{i=1}^{n-1} d_i^p \right)^{\frac{1}{p}}$ can be used to determine a set of candidate points. As with the spatial partitioning algorithms, the precise distance (in the n -dimensional space), need only be determined for points in the candidate set.

An experiment was conducted to test the effectiveness of this “sequential-hybrid” algorithm over (simulated) 3,5,7 and 9-dimensional spaces. Each dimension of the space consisted of 1000 points and distances were uniform random integers between 1 and 1000. In each n -dimensional space, the first $n - 1$ dimensions were used to determine the candidate set. The algorithm computed $\sum_{i=1}^{n-1} d_i^2$ for each of 10^6 (simulated) information elements, with the remaining d_n^2 being computed only for information elements where $\sum_{i=1}^{n-1} d_i^2 \leq t^2$. The resulting retrieved to candidate set sizes are presented in table 11.9. Note that, in order to maintain approximate parity of retrieved set size, t is increased with the dimension of the space.

11.9.1 Discussion of sequential-hybrid results

The sequential-hybrid algorithm can be used to index and search multi-dimensional spaces where *distanceMatrix* is precomputed for most dimensions. Conveniently, while the effectiveness of spatial partitioning algorithms

n	t^2	retrieved set size	candidate set size	retrieved to candidate set size (percent)
3	100	681	9068	7.50993%
5	1000	607	3456	17.5637%
7	3000	657	2418	27.1712%
9	6000	730	2364	30.8799%

Table 11.9: Retrieved to candidate set sizes (by truncated percentage) for a sequential search range queries over 3,5,7 and 9-dimensional spaces (with 10^6 randomly attached information elements). Each dimension consists of 1000 points. Distances are uniform random integers between 1 and 1000. In each n -dimensional space, the first $n - 1$ dimensions were used to determine the candidate set.

normally decreases with the number of dimensions, the effectiveness of the sequential-hybrid algorithm increases!

11.10 Summary, discussion and recommendations

Based on the experiments in this chapter, the metric space, spatial partitioning range query algorithm, and (necessarily) the set space, spatial partitioning range query algorithm (being somewhat weaker) are *not* generic algorithms, but rather suitable only for certain spaces. The standard deviation σ of the space, relative to t (the query radius), gives a good indication of how effective the algorithms will be. In general, the algorithms are only effective over spaces with large σ . From this, a reasonable formula to use to score the suitability of these algorithms is σ/t , where t is the maximum radius where range queries need to be effective. In order to retrieve the same proportion of the space, t must increase for spaces with increased mean distance μ , so the formula σ/μ could also be used. This agrees with [22], which introduces μ/σ as a measure of the “hardness” of the space.

The problem with these algorithms is that σ/μ often decreases as the number of dimensions increases. Also, σ/μ must be quite large for the al-

gorithms to work effectively⁴. While the multiple tree method (see section 11.6.3) improves the algorithm, this only extended the “effectiveness range” of the algorithm from 2 to 3-dimensional Euclidean space. From this, it can be concluded that the algorithm is normally ineffective at indexing spaces with more than three dimensions, making it an unsuitable basis for implementing Knowledge Library functionality.

Fortunately, if *distanceMatrix* can be precomputed, the sequential search algorithm is quite effective. As *distanceMatrix* need only be precomputed for individual dimensions of multi-dimensional spaces and underlying spaces (upon which set spaces are based), sequential search should be a suitable algorithm to implement range queries for many Knowledge Libraries. Furthermore, if sequential search is ruled-out due to the presence of a (near) infinite dimension, the sequential-hybrid algorithm can be used to expedite range queries.

Curiously, spatial partitioning algorithms, which are the main focus of the literature, appear to be effective at indexing only a small class of space, not particularly applicable to Knowledge Libraries. The sequential search algorithm, which appears far more universal and effective, is under represented in the literature, perhaps as it is so straight-forward.

11.11 What this Chapter Achieved

This chapter identified the limitations of both sequential search and spatial partitioning range query algorithms. Sequential search requires *distanceMatrix* to be precomputed (in some form), whereas spatial partitioning algorithms are only effective at indexing spaces where σ/μ is large. The effectiveness of spatial partitioning algorithms can often be improved using a number of search trees, and taking the intersection of the candidate set produced by each tree. The sequential-hybrid algorithm extends the scope of sequential search, as it can be applied to multi-dimensional space that have a large (or

⁴From the experiments in this chapter, a very approximate rule of thumb would be that σ/μ should be greater than 0.4 for metric spaces, and 0.5 for non symmetric spaces, in order for spatial partitioning algorithms to work effectively.

even infinite) dimension.

Candidate set size was identified as the best statistic to use to compare spatial partitioning and sequential search algorithms, while retrieved to candidate set size should be used to compare spatial partitioning algorithms.

Importantly, sequential search and sequential-hybrid algorithms should be effective at providing range query functionality for Knowledge Libraries.

Chapter 12

Summary, Discussion and Future Work

12.1 Chapter Overview

This chapter summarises and discusses the research in this thesis. The potential avenues for future development are also discussed.

Section 12.2 provides a summary of the thesis. Frequent references are made to thesis chapters, making this section an effective index of thesis content.

Section 12.3 discusses what has been achieved in this thesis. The research hypothesis is restated, and the degree to which this research confirms the hypothesis is discussed.

Section 12.4 discusses potential future directions of this research. The main areas with potential for development include: the implementation of Knowledge Libraries; the development of a graphical interface for Knowledge Libraries; improving existing (information retrieval and data warehousing) systems; and improving the way knowledge is disseminated.

12.2 Thesis Summary

12.2.1 The importance of information systems

In order to make decisions it is necessary to first review relevant information. In order to confidently make *good* decisions, *all* relevant information must be taken into account.

Effective information systems support decision making by providing ready access to relevant information—including an indication of what relevant information may be missing. Significantly, it may be that the information elements (that is, the information units indexed by the system) of an information system are not the only relevant information available to the system. In order to make reliably good decisions, system users will normally need to take into account further information such as: the scope and range of information elements in the system; how the system classifies and retrieves information; the reliability and possible bias of information in the system; who contributes information to the system and why; what contributions have been rejected by the system and why; ...

By reducing the cost (in terms of time, money, effort, ...) required to make good decisions, information systems have the potential to radically improve ... practically everything.

Would providing voters in a democracy ready access to high quality relevant information improve the functioning of the democracy?

Are voters really qualified to make an informed decision? Do we know how our representatives voted on issues we have an interest in? Do we know what contributions they made to relevant debates? Are we informed of the vested interests of major campaign contributors? Is the trust we place in journalists (and media empires) to discover, report and publish this information warranted?

Based on the, often vague, speeches of contemporary politicians, many voters make voting decisions based on general noise output, rather than any in depth analysis. Perhaps the two party system itself is necessary to present

under-informed voters with a simple choice.

Would providing key decision makers in large organisations ready access to high quality relevant information improve the functioning of the organisation?

Decision makers in competitive environments are already aware of the critical importance of quality information. Improved access to, and quality of, information gives a competitive advantage.

Would providing scientific researchers ready access to high quality relevant information improve the quality of scientific research?

Currently researchers have access a subset of the relevant (and a substantial amount of non relevant) research through the keyword interfaces of the research databases their sponsoring institution subscribes to. There is normally a delay of some years between the time a research advance is made, and when the corresponding (peer reviewed) paper becomes available through these databases. Due to limitations of the peer reviewed research publication process, some genuine research advances are never published, while other papers that do not represent genuine advances *are* published. Also, sometimes the connection between the “academic” and “real” worlds can be tenuous, with the real world impact of academic research often being difficult to assess.

Quite clearly, many aspects of the existing (global, research information) system could be improved. Any such improvements should generate a corresponding improvement in the quality and/or pace of scientific research.

12.2.2 The significance of Knowledge Libraries

The very perspective and terminology that characterises a discipline focuses, and so limits, the (research) advances within that discipline. For example, the discipline of information retrieval (see sections 2.6 and 4.2.3) appears to be limited by its perspective of the problem of classification and its close ties to keyword classification.

Information retrievalists avoid the use of the term “classification”, prefer-

ring “indexing”. In the context of this research it has been useful to differentiate between: *indexing* (herein also called “weak classification”), which is the classification of information to expedite retrieval; and *strong classification*, which is the positioning of information (relative to other information) within a “universe of knowledge” (see section 1.3). It would appear that, to make fundamental advances, information retrievalists will have to begin to consider the problem of classification in the stronger sense, yet to do so immediately casts doubt on their dependence upon non contextural keywords to classify information.

This research attempts to sidestep controversy within the discipline of information retrieval by framing entirely new research questions. While information retrieval systems simply map user queries onto sets of documents in a collection, *Knowledge Libraries* (see chapter 3) are idealised systems that organise information to promote understanding. While mapping user queries onto sets of information elements is part of the core functionality of Knowledge Libraries, this is only one aspect of what Knowledge Libraries do. Knowledge Libraries also convey to users an understanding of how the information elements of the library interrelate. Users of Knowledge Libraries may be interested in how the information elements are organised, rather than the information elements themselves. Knowledge Library users may want to identify, for example, “knowledge frontiers” or “gaps” in the classified information.

The, more holistic, consideration of the needs of users, sets Knowledge Libraries apart from information retrieval systems. Clearly, strong classification, rather than weak classification, must be utilised in order to provide core Knowledge Library functionality.

12.2.3 The mathematical basis for Knowledge Libraries

Without a firm mathematical basis, it is doubtful that anything approaching a Knowledge Library could be implemented. This research discusses how *information spaces*—triples (M, d, \mathcal{I}) where $\langle M, d \rangle$ is a classification space and \mathcal{I} is an index of N over M , for some set N of *information elements*—

provides the required mathematical basis for Knowledge Libraries (see, in particular, chapters 8 and 9).

This mathematical basis is very carefully developed over 5 chapters. There are a number of notable stages in this development.

1. The desired properties of distance functions d are identified (see sections 4.8 and 4.9).
2. The n -dimensional distance function $G_p^{d_1 \dots d_n}$ is defined (see section 4.4) and shown to have all the desired properties (see section 5.2).
3. Two families of set distance function d_{ij}^M (see section 5.4) and $\frac{M}{k}d$ (see section 7.5) are defined and it is shown that $d_{av\ 0}^M(x, y)$ and $d_{100\ 0}^M(x, y)$ (see section 5.4), $\frac{M}{1}d$ and $\frac{M}{av}d$ (see sections 7.5 and 7.6) have all the desired properties.
4. L -collections are introduced (see chapter 6). L -collections generalise sets (and the associated set operators), allowing each element of a set to be associated with a positive real.
5. It is shown that $\frac{\mathcal{M}}{1}d$ (see section 7.5) and $\frac{\mathcal{M}}{av}d$ (see section 7.6) have all the desired properties (where \mathcal{M} is an L -collection).
6. Given a network (see section 8.3), it is shown how a corresponding distance function d , based on the network, with all the desired properties, can be defined (also in section 8.3).
7. It is shown that the set distance function $d(X, Y) = |X| - |Y|$ and the L -collection distance function $d(\mathcal{X}, \mathcal{Y}) = |\mathcal{X}| - |\mathcal{Y}|$ have all the desired properties (see sections 5.3 and 7.3).
8. To complete the mathematical basis for Knowledge Libraries, this research discusses how useful classification spaces $\langle M, d \rangle$ can be defined (see section 8.4), and how indexes \mathcal{I} can attach information units to points $x \in M$ (see section 8.6), forming information spaces (M, d, \mathcal{I}) (see section 8.7).

9. Two in depth worked examples are given that demonstrate how information space provides core functionality (see chapter 9).

12.2.4 Implementing Knowledge Libraries

Due largely to their strong mathematical basis, the problem of implementing Knowledge Libraries can be related to the literature on non traditional databases based on metric space. Range queries (and the related k nearest neighbour and ranked queries) present the main challenges when implementing Knowledge Libraries. The existing (metric space) partitioning range query algorithms are presented (see section 10.3), improved (see section 11.5), and generalised to work over information space (see section 10.4). However, as experiment results show (see section 11.7), the existing algorithms are ineffective, even for many metric spaces. A, more suitable, alternate technique is discussed (see section 11.8) and extended (see section 11.9).

In summary, sequential search algorithms, where *distanceMatrix* is pre-computed for underlying spaces, are effective over n -dimensional spaces and set spaces where the size of point sets are limited. Sequential search algorithms (see section 11.8) should be sufficient to provide range query functionality for most Knowledge Libraries. If it is impractical to precompute or store *distanceMatrix* for a dimension of an n -dimensional space, the sequential-hybrid algorithm (see section 11.9) can be used. Sequential search and sequential-hybrid algorithms, together, should be a sufficient basis to implement Knowledge Library range query functionality. These algorithms can be adapted to implement other related queries, such as k nearest neighbour and ranked queries.

12.3 Discussion

The research hypothesis of this thesis was that information space provides a mathematical basis for the development of Knowledge Libraries, that in turn provides important functionality, not currently provided by existing applications such as relational databases and information retrieval systems.

This thesis has discussed the limitations of relational databases and information retrieval systems. Knowledge Libraries have been informally defined and information space has been formally defined. It has been shown how information space provides a basis for the implementation of Knowledge Library functionality, existing algorithms have been extended and new algorithms developed to effectively implement this functionality. In so far as is possible—without the more formal definition, implementation and testing of Knowledge Libraries—the hypothesis has been confirmed.

12.3.1 The Contribution of this Thesis

The main thrust of this thesis was to provide the basis for Knowledge Libraries; a new class of information organisation system with considerably more power than existing systems. On the way to achieving this goal, a number of other significant contributions were made.

This research:

1. identified the need for the development of a coherent “information organisation” discipline with computer filesystem, information retrieval, database and other sub disciplines. This new discipline would define common terminology and exploit synergies between these research areas;
2. identified significant limitations of existing information organisation techniques;
3. identified the need for the development of better information organisation techniques for scholarly research generally. Such techniques could dramatically increase the pace of scientific development;
4. described “Knowledge Libraries”—an idealised system for the storage, organisation, and display of information;
5. defined “ L -collections” a generalisation of sets (also multisets, rough sets and fuzzy sets) that allows elements to have membership “grades” chosen from the set L ;

6. defined “information space” and showed how information space can provide the mathematical basis for Knowledge Libraries;
7. provided effective range query algorithms for information space;
8. showed how Knowledge Libraries and information space generalise hierarchical filesystems, vector space information retrieval, relational databases, and OLAP datacubes.

12.3.2 The more formal development of Knowledge Libraries

This research, somewhat informally, describes Knowledge Libraries. Knowledge Libraries are more of an ideal—an approach to information organisation—rather than a concrete specification that can be implemented.

An alternate (or supplementary) approach would be to formally enumerate the functions and behaviour of Knowledge Libraries. This amounts to an application programming interface (abbreviated API) for Knowledge Libraries. There are two main limitations of this approach. First, APIs do not motivate—they do not connect Knowledge Libraries to realistic use scenarios. Second, it is highly likely that some important functionality would be missed in the initial formulation of an API. API versions are best developed as part of a development spiral that includes API (re)specification, implementation, testing, and analysis phases. This research lays the groundwork for this type of development.

12.3.3 The flexibility of information space

Information space is an extremely flexible classification system. Information space generalises hierarchical classification (such as the Dewey Decimal system, see section 2.3), faceted classification (see section 2.3.1), the relational model (see section 4.2.6), the vector model for information retrieval (see section 4.2.3), and metric space based databases (see section 4.2.1).

Section 8.3 shows how distance functions can be based on networks (including hierarchies), example 8.1 shows how metric space can provide the

mathematical basis for hierarchical classification. The dimensions of a multi-dimensional space correspond to the facets or fundamental classifications of faceted classification. Similarly, n -dimensional space corresponds to n -ary Relations of the relational model. The classification space $\langle \mathcal{P}(M), d \rangle$ where M is a set of keywords and $d(\mathbf{x}, \mathbf{y}) = \sqrt{1 - \text{sim}(\mathbf{x}, \mathbf{y})^2}$ (see section 4.2.3) can be used to provide an information space equivalent of “vector based” information retrieval.

Information space is closely related to data warehousing, and may one day contribute to the development of formal definitions of data warehousing systems.

12.4 Future Work

This research leaves no theorems unproved and no difficult and critical algorithms to write. Even so, substantial work remains.

12.4.1 The implementation of Knowledge Libraries

As is already discussed in section 12.3, the implementation and demonstration of Knowledge Libraries should provide final confirmation of the research hypothesis of this thesis. This implementation should be done as part of a development spiral, with API (re)specification, implementation, testing, and analysis phases. The first step would be to specify Knowledge Library API version 1.0, which should include a detailed list of functions required to provide core Knowledge Library functionality. This API should give precise definitions of each function.

12.4.2 Graphical Interface

Graphical interfaces may aid Knowledge Library users to better visualise, and so more readily understand, information spaces. As has been discussed (see section 3.7.5), scatter plots may be utilised to visualise 2-dimensional projections of n -dimensional information spaces.

The honours thesis [77], includes some preliminary work that could be applied towards the development of Knowledge Library graphical interfaces. In this work, dimensions can be “activated” by users. The coordinates of activated dimensions are displayed. Each activated dimension is displayed in a separate frame. Users specify query points by “selecting” coordinates in the activated dimensions. The information element icons of retrieved information elements are displayed in a further frame. The full classification of an information element is displayed in response to a single click of its icon. Double clicking the icon displays the information element in a separate window.

12.4.3 Improving existing systems

This research could be applied towards improving existing information retrieval and data warehousing systems.

There are a number of ways information retrieval systems could be improved.

One suggestion, is to provide users with a glimpse of the classification space—perhaps by providing an autocomplete menu when typing keywords. In this way, each user’s knowledge of the key terms used to classify documents would be improved with each search. This would help users learn how to formulate more effective queries.

Another suggestion would be to provide a mechanism to disambiguate key terms that are polysemes or homonyms (see section 2.6.2), both when indexing and retrieving documents. For many key terms, it should be possible to use the term’s context in a document to automatically disambiguate its meaning when indexing. Users could be presented with dictionary definitions to assist with disambiguation when forming queries.

Data warehousing systems could be improved by the development of a formal mathematical basis. It seems likely that this basis would be closely related to information space.

Wikipedia[48], the Encyclopedia of Life[101] and the Gene Ontology[28] are all major information organisation and presentation efforts. There are

many others. In all these collections, as far as the author is aware, information is exclusively indexed by keyword. How much more useful would the Encyclopedia of Life be, for example, if flora were classified by leaf shape (so users could locate entries by scanning exemplar leaves) and if fauna were classified by preferred food (so users could locate pest control species). All life in the encyclopedia could be also classified by habitat descriptors selected from a controlled vocabulary. The geographical location of wild populations could provide another useful dimension for classification. If an estimate of the size of these populations was also recorded (as the multiplicity of information elements) the encyclopedia could also be a useful conservation resource.

12.4.4 The dissemination of knowledge

One of the main areas of application envisioned for Knowledge Libraries is in assisting with the dissemination of research. As has been discussed (see sections 2.6.2, 3.4.1, 11.2.2), the existing “system” is far from ideal. Knowledge Libraries could help the way knowledge is disseminated in a number of ways. Knowledge Libraries could:

1. help identify and take advantage of synergies between related research areas;
2. help researchers identify gaps in research;
3. help researchers (more effectively) locate relevant research;
4. reduce the delay between submission and publication of research;
5. help relate research to the real world;
6. assist in more accurately valuing research;
7. allow different formats for research publication;
8. facilitate greater research collaboration.

Given the huge potential benefits, this area warrants special attention.

One, quite basic, suggestion would be to create an information space of “scientific” terminology. The information elements in the space would be the scientific terms and associated definitions. The classification space would include “alphabetic”, “first use” (by time), and “topic area” dimensions. Users of the space could browse the terms, contribute new terms and suggest improvements to existing entries.

This space could itself be the basis for a “contributor” information space. The contributor space would include “alphabetic”, “topic area”, “contributed term”, and possibly “sponsoring institution” dimensions. The coordinates in the contributed term dimension would be the information elements from the terminology space. The information elements of the “contributor” space would be “term contributors” and would allow users to view the contributions of contributors to the terminology space.

Ultimately, these spaces could be the basis for a “research” Knowledge Library that would also include “research paper” type and other research related information elements. Users of this library could inform themselves about research activity, and contribute and critique research related information elements.

Appendix A: A Guide to the Accompanying CD

This appendix describes the contents of the CD that accompanies this thesis. The CD includes the source code for the programs used in the experiments in chapter 11. The source code was compiled using g++, the GNU project C++ compiler (see `man g++` on UNIX or GNU/Linux machines). The compiler option `-D_FILE_OFFSET_BITS=64` was used when the compiled code as required to read from, or write to, large files. No other compiler options were used.

Note that source code written for one table or figure was often modified and reused to generate data for other tables and figures. As a result, the directories in the CD—which correspond to the figures and tables in chapter 11—often contain source files that are superficially similar to other source files, with the same name, in other directories. Even though these files have the same name and are often quite similar, they should not be substituted for one another.

The output from all these programs was written to `stdout`. The figures were generated, from this output, using R (see www.r-project.org).

Figure 1

The distributions illustrated in figure 11.1 were generated using *distanceMatrixGen.cpp* source code and *spaceGlobals.h* header files. A copy of each of these files can be found in the *figure1* directory on the CD.

The number of edges was manipulated by changing the value of e in *spaceGlobals.h*.

Figure 2

The distributions illustrated in figure 11.2 were generated using *distanceMatrixGen.cpp* and *symmetricalDMGen.cpp* source code, and *spaceGlobals.h* and *symSpaceGlobals.h* header files. A copy of each of these files can be found in the *figure2* directory on the CD.

Table 1

The query times presented in table 11.1 were generated using *distanceMatrixGen.cpp* and *symmetricalDMGen.cpp*, *treeGen.cpp* and *symmetricalTGen.cpp*, *bruteForceSearch.cpp* source code, and *spaceGlobals.h* and *symSpaceGlobals.h* header files. A copy of each of these files can be found in the *table1* directory on the CD.

Note that, to function, *symmetricalDMGen.cpp* and *symmetricalTGen.cpp* code requires a directory *data* to exist in the same directory as the source code. Also, for *symmetricalDMGen.cpp*, the number of nodes n and the number of edges e are read in as command line arguments. The number of nodes is the sole command line argument for the *symmetricalTGen.cpp* program.

The number of nodes and edges for *distanceMatrixGen.cpp* and *treeGen.cpp* was manipulated by changing the values of n and e in *spaceGlobals.h*.

Figure 3

The distributions illustrated in figure 11.3 were generated using *symmetricalDMGen.cpp* and *symmetricalTGen.cpp* source code, and *spaceGlobals.h* header, files. A copy of each of these files can be found in the *figure3* directory on the CD.

Note that *symmetricalDMGen.cpp* and *symmetricalTGen.cpp* require a directory **data** to exist in the same directory as the source code. Also, for *symmetricalDMGen.cpp*, the number of nodes n and the number of edges e are read in as command line arguments. The number of nodes is the sole command line argument for the *symmetricalTGen.cpp* program.

Tables 2 and 3

The search tree statistics presented in tables 11.2 and 11.3 were generated using *symmetricalDMGen.cpp* and *symmetricalTGen.cpp* source code, and *spaceGlobals.h* header, files. A copy of each of these files can be found in the *tables2-3* directory on the CD.

Figure 4

The distributions illustrated in figure 11.4 were generated using *symDMGen.cpp*, *dMGenLibrary.cpp* and *euclid.cpp* source code and *spaceGlobals.h* header files. A copy of each of these files can be found in the *figure4* directory on the CD.

The files *symDMGen.cpp*, *dMGenLibrary.cpp* and *spaceGlobals.h* were used to generate the LHS of figure 11.4. The number of nodes is read in as a command line argument. The file *euclid.cpp* was used for the RHS.

Figures 5 and 6

The *distanceMatrix* for each of the 1000 spaces in figure 11.5 was generated using *symDMGen.cpp*, *dMGenLibrary.cpp* source code and *spaceGlobals.h* header files. The compiled program, must be called *symdm*. The program generated from the source code *symTGen.cpp*, *sTGenLibrary.cpp* and *ranged-QueryLibrary.cpp*, and the header *spaceGlobals.h* uses a system call to invoke *symdm*.

The source code *euclidQuery.cpp*, along with the header *spaceGlobals.h*,

was used to generate figure 11.6. The source code *mean.cpp* was used to generate the mean for both figures.

A copy of each of these files can be found in the *figures5-6* directory on the CD.

Tables 4 and 5

The search tree statistics presented in tables 11.4 and 11.5 were generated using *euclidQuery.cpp* source code and *spaceGlobals.h* header files. A copy of each of these files can be found in the *tables4-5* directory on the CD.

The number of points is the sole command line argument for the program generated from *euclidQuery.cpp* source code.

Figure 7

The *distanceMatrix* for each of the 1000 spaces in figure 11.7 was generated using *symmetricalDMGen.cpp* source code and *spaceGlobals.h* header files. The compiled program, must be called *symdm*. The program generated from the source code *centerSelect.cpp* and *symmetricalTGen.cpp*, and the header *spaceGlobals.h* uses a system call to invoke *symdm*. The source code *mean.cpp* was used to generate the means in this figure.

A copy of each of these files can be found in the *figure7* directory on the CD.

Table 6 and Figure 8

The *distanceMatrix* for each of the 1000 spaces in figure 11.8 was generated using *symmetricalDMGen.cpp* source code and *spaceGlobals.h* header files. The compiled program must be called *symdm*. The program generated from the source code *symmetricalTGen.cpp* and the header *spaceGlobals.h* uses a system call to invoke *symdm*. The source code *mean.cpp* was used to generate the means in this figure.

A copy of each of these files can be found in the *figure8* directory on the CD.

Note that, to function, *symmetricalDMGen.cpp* and *symmetricalTGen.cpp* code requires a directory *data* to exist in the same directory as the source code. Also, for *symmetricalDMGen.cpp*, the number of nodes n and the number of edges e are read in as command line arguments. The number of nodes is the sole command line argument for the *symmetricalTGen.cpp* program.

An earlier version of the *symmetricalTGen.cpp* source code was used to generate table 11.6.

Figures 9 and 10

The distributions illustrated in figure 11.10 were generated using *euclid-Query.cpp* source code and *spaceGlobals.h* header files. A copy of each of these files can be found in the *figure10* directory on the CD.

The command line arguments for the compiled program are the desired number of points and the number of dimensions.

Figure 11.9 can be generated from *euclidQuery.cpp* source code by “un-commenting” the *euclidean_space_display()* function on line 351.

Figure 11

The distributions illustrated in figure 11.11 were generated using *MTeuclid-Query.cpp* source code and *spaceGlobals.h* header files. A copy of each of these files can be found in the *figure11* directory on the CD.

The command line arguments for the compiled program are the desired number of points and the number of dimensions. The number of search trees is controlled by the *numOfTrees* constant in *spaceGlobals.h*.

Figure 12

The distributions illustrated in figure 11.12 were generated using *treeGen.cpp* source code and *spaceGlobals.h* header files. A copy of each of these files can be found in the *figure12* directory on the CD.

The sole command line argument for the compiled program is the desired number of points.

Figure 13

The *distanceMatrix* for each of the 1000 spaces in figure 11.13 was generated using *distanceMatrixGen.cpp* source code and *spaceGlobals.h* header files. The compiled program must be called *dm*. The program generated from the source code *treeGen.cpp* and the header *spaceGlobals.h* uses a system call to invoke *dm*.

A copy of each of these files can be found in the *figure13* directory on the CD.

Note that, to function, *distanceMatrixGen.cpp* and *treeGen.cpp* code requires a directory *data* to exist in the same directory as the source code. For *distanceMatrixGen.cpp*, the number of nodes n and the number of edges e can be set by editing *spaceGlobals.h*. It is important that the value of n is not changed between compilations of corresponding *distanceMatrixGen.cpp* and *treeGen.cpp* programs.

Table 7

The query time statistics presented in table 11.7 were generated using a *ss.cpp* source code file. A copy of this file can be found in the *table7* directory on the CD.

Table 8

The query time statistics presented in table 11.8 were generated using a *sss.cpp* source code file. A copy of this file can be found in the *table8* directory on the CD.

Table 9

The retrieved and candidate set size statistics presented in table 11.9 were generated using a *sha.cpp* source code file. A copy of this file can be found in the *table9* directory on the CD.

Appendix B: Publications

Relating to this Thesis

This appendix very briefly describes the author's publications that relate to this thesis.

E. Rayner. Developing a method of algorithm classification for computer algorithm library applications. 2003. Honours Thesis.

This honours thesis motivates, develops and presents an application program for the organisation and presentation of algorithms. An algorithm classification space is formed from a number of hierarchical and ordinal dimensions. This early work is formalised and generalised in this (Ph.D.) thesis. The graphical user interface used in the application program is discussed in section 12.4.2 (of this Ph.D. thesis.)

E. Rayner, Searching for the Science in Information Science, *Rhizome*. 1, 2005, pp215-228.

This journal paper discusses the status and value of information science and how efforts towards establishing more formal information organisation systems (such as Knowledge Libraries and information space) can make an important, even critical, contribution.

E. Rayner, I. Piper, M. Bunder. Introducing *N*-Tree Space: A Classification System for Knowledge Library Applications. *Proceedings IEEE Tencon*, November 2005, pp399-403.

This conference paper discusses some of the limitations of information retrieval and how *N*-Tree Space, a precursor to information space, can provide a formal basis for some Knowledge Libraries.

E. Rayner, The Organisation of Information, *Proceedings of the 2nd International Conference on Information Management and Business*, Sydney, February, 2006.

This conference paper discusses the nature of information, describes Knowledge Libraries and motivates their development, and describes how information space could be used to provide a formal basis for Knowledge Libraries. This paper represents an early stage in the formal development of information space.

Appendix C: Glossary of Information Organisation Terms

This appendix provides a glossary of Information Organisation terms used in more than seven distinct computing disciplines (enumerated below) and, in particular, in this thesis. After each entry it is indicated, through the following notation, which discipline or disciplines a term is directly relevant to—within the context of this research.

1. Traditional classification (*c*).
2. Databases (*d*).
3. Computer file systems (*f*).
4. The internet, the world wide web and the semantic web (*i*).
5. Knowledge libraries and information space (*k*).
6. Information retrieval (*r*).
7. Data warehousing (*w*).
8. Metric space queries (non traditional databases) (*m*).

The symbol (*o*) is used to denote a term that has its origin in this thesis. Cross referenced terms are introduced with the notation *cf*.

Example 12.1. In the glossary entry for the term **access path consistency**, the notation (c, k, o) indicates that the term is directly relevant to the discussion on *traditional classification* and *Knowledge Libraries and information space*, and is originally defined in this thesis. At the end of the entry,

cf dynamic dimension, hospitality, indexing consistency.

lists other closely related terms included in the glossary.

This glossary has a dual purpose. First, it describes relevant objects or concepts and so provides a ready reference for this thesis. Second, it describes some important terms in the “language of information organisation”. As such, it is part of the research findings of this thesis.

Abstract. *See* classification.

Access path consistency. *A classification scheme is access path consistent if, when new classes are added, the path that users of the scheme use to navigate to existing classes remains unchanged. For example, hierarchical classification schemes are access path consistent if new classes are always leaf nodes, rather than internal nodes (and the hierarchy remains otherwise unchanged). See section 2.3.3.*
 (c, k, o)

cf dynamic dimension, hospitality, indexing consistency.

Analytico-synthetic classification [scheme]. *A type of classification scheme that is developed by first analysing the subject matter to identify separate concepts. Classes are then synthesised by joining together a number of separate concepts[16].* (c)

cf bottom-up classification, enumerative classification, top-down classification.

Apex topic. *An apex topic is an information element that is attached to a coordinate in a precedence dimension and is dependent on a number of other topics. See scenario 2.1, section 3.3. (k, o)*
cf information element, information space, precedence dimension.

Application program. *“a computer program that interacts with the database by issuing an appropriate request... to the data base management system[95].” (d)*
cf database, data base management system.

Attach. *See* index.

Automatic dimension. *A type of implemented information space dimension. Information units added to (n-dimensional) information spaces are automatically attached to a coordinate in automatic dimensions. See scenario 1, section 3.3. (k, o)*
cf dimension, dynamic dimension, information space, information unit, manual dimension, n-dimensional space.

Background. *See* classification.

Bottom-up classification [scheme]. *A type of classification scheme where classes are “built-up” from component terms[16]. (c)*
cf analytico-synthetic classification, enumerative classification, top-down classification.

Call number. *“Assigned by the cataloguer... composed of a classification number [often called “class number”] followed by additional notation to make the call number unique[78].” (c)*
cf cataloguer, class.

Candidate point. *An element of a candidate set. Candidate points are also known as considered points as they are considered (and possibly rejected) for retrieval. (m, o)*
cf Candidate set, *k* Nearest neighbour query, range query, ranked query, retrieved set.

Candidate set. *Spatial partitioning range query (also *k*-NN and ranked query) algorithms (discussed in chapter 10) and the sequential-hybrid range query algorithm (introduced in section 11.9) initially produce a **candidate set** of points from which the final retrieved set is determined. (m, o)*
cf Candidate point, *k* Nearest neighbour query, range query, ranked query, retrieved set.

Cataloguer. *“A librarian primarily responsible for preparing bibliographic records to represent the items acquired by a library...[78].”*
 (c)
cf call number, classificationist, classifier.

Class. *“A grouping of objects or concepts based on one or more characteristics, attributes, properties, qualities, etc., that they have in common...[78].”* (c)
cf call number.

Classificationist. *“...the person who devises a scheme of classification[76].” Bliss prefers the term “classificationner”[13].* (c)
cf cataloguer, classifier.

Classification.

i) “[a] division or category within a system...[89]”—essentially a synonym for “class”.

ii) “[the] process of putting things into groups according to similarities or relationships[89]”—essentially a synonym for “classify”.

iii) an abbreviation for “classification scheme”. (c)

cf analytico-synthetic classification, bottom-up classification, classification scheme, enumerative classification, strong classification, weak classification, top-down classification.

iv) The **classification** of an information element in an information space is the point to which it is attached. (k, o)

cf information element, information space.

Classification Scheme. “[A] system of organising things by dividing them into groups based on their similarities[89]. (c)

cf classification.

Classification Space. *The space that provides the points that are used to give distances between information elements of an information space. See the discussion section 8.4 and definitions sections 8.6.2 and 8.7. (k, o)*

cf information element, information space, space.

Classifier. “...the one who constructs class numbers for subjects in accordance with a preferred scheme of classification[76].” (c)

cf cataloguer, classificationist.

Collocation. “the act or result of placing or arranging together[1].” (c)

Coordinate. See dimension.

Data. *Raw facts without structure or context. (d, f, k)*

cf field, file, information.

Database. *“A shared collection of logically related data, and a description of this data, designed to meet the information needs of an organisation[95].” (d)*

cf data, database management system, database system, data directory, operational database, relational database.

Database management system. *“a software system that enables users to define, create, maintain, and control access to the database[95].” Abbreviated DBMS. (d)*

cf data, database, database system, data directory.

Database system. *“a collection of application programs that interact with the database along with the database management system and database itself[95].” (d)*

cf data, database, database management system, data directory.

Data dictionary. *Also known as a data directory, system catalogue, or metadata. A file that describes the data in a database[95, 78]. (d)*

cf data, database, database management system, database system.

Data mart. *A database that provides specialised data analysis[75]. (w)*

cf database, data warehouse, data warehousing, dimensional data-mart.

Data warehouse. *A generic solution to a business’s data analysis needs[75]. (w)*

cf data, data mart, data warehousing.

Data warehousing. *A topic about building databases that have been optimised for analysis[75]. (w)*

cf data, data mart, data warehouse.

Dimension. *Mathematically, a **dimension** is one of the spaces making up an n -dimensional space. The points in the dimension are called **coordinates** in the n -dimensional space. See definitions, section 4.4. (k, o)*
cf space, n -dimensional space.

*The term **dimension** is also used, with less formality, to describe Knowledge Libraries, information retrieval systems and data marts. (k, r, w)*
cf Automatic dimension, dimensional data mart, dynamic dimension, information retrieval, manual dimension, static dimension.

Dimensional data mart. *A data mart that is constructed using dimensional modelling techniques[75]. (w)*
cf data mart, star-join schema.

Directory. *A special type of file that can “contain” other files. Directories are sequential files where each record is a list of file attributes. The file name is the key field[88, 92]. (f)*
cf field, file, filesystem.

Distance. *See distance function.*

Distance function. *A **distance function** over M is a function d that maps pairs of elements (points) of a set M into $\mathbb{R}^{\geq 0}$. $d(x, y)$ is called the **distance** from x to y . A metric is a type of distance function. See definitions section 4.2.1. (k)*
signed distance function, space.

Document. *The fundamental “information unit” of information retrieval. (r)*
cf information retrieval, information unit.

Drill down operator. *A dimensional data mart function that expand reports to show more detail by adding further domains (table columns)[75]. (w)*

cf data mart, dimensional data mart, drill up, slice and dice.

Drill up operator. *A dimensional data mart function that summarise reports by removing domains (table columns)[75]. (w)*

cf data mart, dimensional data mart, drill down, slice and dice.

Dynamic dimension. *A type of implemented information space dimension. In a dynamic dimension the coordinates and/or distances are automatically determined, usually based on the information elements in the information space. Hence adding information units to the information space can change existing classifications. See scenario 1, section 3.3. (k, o)*

cf access path consistency, automatic dimension, dimension, information element, information space, information unit, manual dimension, static dimension.

Enumerative classification [scheme]. *A type of classification scheme where all classes within the scheme are enumerated[16]. (c)*

cf analytico-synthetic classification, bottom-up classification, top-down classification.

Explicit classification.

...a purposeful classification, which may be either hierarchical or non-hierarchical, to facilitate retrieval... Most explicit classification schemes are constructed to encompass a particular point of view—either that of the classifier or that of a typical or most frequent user[93]. (c)

cf implicit classification.

Extension and intention. *The extension of a class is the set of objects in that class, while the intension is the set of attributes that are shared by the objects in that class. Both these terms are also used in formal concept analysis with the same meaning[16]. (c)*
cf specific subject.

Field. *A field is the basic element of data. An individual field contains a single value... It is characterised by its length and data type... Depending on the file design, fields may be of fixed or variable length ...[88]. (f)*
cf data, file, record.

File. *An interface that allows access to information without the need to explicitly consider details such as the data's location on disk, block length, process blocking, security, encoding, ... etc. Files are identified by name. Many operating systems support two-part file names with each part separated by a period. The second part, called the file extension, indicates the type of the file. A File has a number of attributes, such as name, type, size, location, etc.[88, 92]. (f)*
cf directory, filesystem.

Filesystem. *“a structure for keeping and locating data in files ...[66].” The purpose of a filesystem is to make it easy to find and access these data files. (f)*
cf data, directory, file.

Fringe subject. *A fringe subject is a subject that is not really part of a (specialised) collection, but can be usefully included in the classification scheme nevertheless. Statistics, for example, is a frequent fringe subject[16]. (c)*

Granularity (of information space). *The larger and more complex the information elements of an information space, the **coarser** the granularity of the space. The smaller and simpler the elements of an information space the **finer** the granularity. See scenario 1, section 3.3..* (k, o)
cf information element, information space.

Homonym. *A word that has a number of different—unrelated—meanings.* (r)
cf polyseme.

Hospitality. *A measure of the degree to which a classification scheme can accommodate new classes[78].* (c)
cf access path consistency.

Implicit classification.

The assignment of words (index terms, descriptors, etc.), whether on the basis of human judgement or by a priori rules carried out by machine, to represent a document... in essence a non-hierarchical classification of the document. This classification of the document is unintentional since the emphasis is on the selection of index terms[93].
(c, r)

cf explicit classification.

Index. *Mathematically, an **index** is a structure that attaches information units to points in space. A space with an index is called an information space. Information units attached to points can be referred to as **information elements** of the information space. See definitions sections 8.6.2 and 8.7. (k, o)*

cf information space, information unit, space

*More traditionally, an **index** is an alphabetised list of key terms with page numbers to refer the reader to the point in a document at which information pertaining to the term is found[78]. In computer and information science **index** is often used more generally—a data structure to speed up searches[7]. (c, k, r)*

Indexing. *The process of creating an (information science type) index (especially in information retrieval.)*

cf index.

Indexing consistency. *An information retrieval system is index consistent if, when documents are added or removed from the collection, the terms that index each document remain unchanged, and the same documents (in the collection) are retrieved by the same queries. See section 4.2.3. (r, o)*

cf access path consistency, document, information retrieval system.

Information. *Structured data in context. (d, k, r)*

cf data, information element, information retrieval, information unit.

Information element. *An **information unit** that has been attached to a point in an information space. The information unit becomes an **information element** of the information space after it is attached. See the discussion scenario 1, section 3.3 and the definition section 8.7. (k, o)*

cf index, information space, information unit, space.

Information island. *A collection of documents that is isolated from other documents—about similar things—because of the special terminology used. See section 2.6.2. (r, o)*
cf document, information swamp.

Information retrieval. *A topic about modelling user information needs with queries, and then matching these queries with “documents” in a collection. Abbreviated IR. (r)*
cf document, information retrieval system.

Information retrieval system. *A mechanism—usually a computer program—that accepts queries and outputs sets of documents that it deems to be “relevant” to these queries. See section 2.6. (r, o)*
cf document, information retrieval.

Information space. *Mathematically, an information space is a space, called a classification space, and an index. Distances between points in the classification space are used to give distances between the information elements of the information space. See definition section 8.7. (k, o)*
cf classification space, distance function, index, information element, signed distance function, space.

Information swamp. *A collection of documents—about dissimilar things—that are often retrieved together because they are indexed by keywords that are homonyms or polysemes. Also called “infogluts[90]”. (r, o)*
cf document, homonym, information island, polyseme.

Information unit. *The basic unit of information in a Knowledge Library. Each unit of information has a classification in the library. Equivalent to information retrieval's "documents" and relational data's "atomic values".* **Information units** can be research papers, web sites, math equations, digital photographs, quotations, questionnaires, algorithms, numbers, (k, o)

cf information, information element, information space, information retrieval, Knowledge Library.

the Internet. *A worldwide computer network. Computers attached to the internet, called "hosts", communicate with one another by sending "packets" of information. Smaller computer networks are connected together using dedicated packet-switching computers called "gateways" or "IP routers" or "Intermediate Systems". The internet protocol suite specifies how information is transmitted between computers in the network[94].* (i)

Introduction. *See* classification.

Join.

- i) If (M, \leq) is a partially ordered set, the **join** of two elements $x, y \in M$ is the least upper bound (the supremum) of x and y and is denoted $x \vee y$. See sections 4.2.5 and 6.2.3. (d, k)
 - ii) In relational algebra, the **join** of two relations R_1 and R_2 is the composition $R_2 \circ R_1$ of R_1 and R_2 [24]. See section 4.2.6. (d, k)
 - iii) When sets are generalised so that each element x has an associated multiplicity or “membership”, union can be generalised in two ways to produce two different operators merge and join. When two (generalised) sets are **joined**, the multiplicity of each element common to both (generalised) sets is a supremum—usually the maximum—of the multiplicities of the element in each operand (generalised) set. If the operands are sets, join is equivalent to union. See section 6.2.3. (k, o)
 - iv) The join form of union is defined for L -collections in section 6.3. (k, o)
- cf merge.

k Nearest neighbour query. Given an information space (M, d, \mathcal{I}) where $\mathcal{I} = (J, m)$, a point $x \in M$ and a limit $k \in \mathbb{N}_1$, the **k nearest neighbour** (abbreviated **k -NN**) **query** $q_{k-NN}(x, k)$ is any (minimal) set $Y \subseteq M^I$ where $\sum_{\{y \in Y\}} m(y) \geq k$, for all $y \in Y$ and $y' \in M^I - Y$, $d(x, y) \leq d(x, y')$. That is, k -NN query $q_{k-NN}(x, k)$ “retrieves” a minimal set of points (with at least k information units attached) “nearest” to x . If each point in M can have at most one information unit attached, $|Y| = k$ simplifies $\sum_{\{y \in Y\}} m(y) \geq k$. (m)

cf Candidate set, range query, ranked query, retrieved set.

Knowledge Library. An idealised system that provides a range of functionality for the organisation and presentation of information. Each **Knowledge Library** is based on an information space which models the relationships between information units in the library. See section 1.6 and chapter 3. (k, o)

cf information space, information unit.

Knowledge precedence. *If the information elements of an information space are used to model knowledge, precedence dimensions can be used to model the knowledge requirements of information elements. See section 3.3. (k, o)*

cf information space, information element, knowledge requirement, precedence dimension.

Knowledge requirement. *Consider two information elements, a and b , in a Knowledge Library. If it is necessary to know a , before one can understand b , we say that a is a **knowledge requirement** of b . Conversely, b is knowledge dependent on a . See section 3.3. (k, o)*

cf information space, information element, Knowledge Library, knowledge precedence, precedence dimension.

Knowledge space. *If the information elements of an information space model knowledge, then a region in the information space can be used to model the knowledge of a user. If a region in information space is used to model the knowledge of a user it is called the user's **knowledge space**. See section 3.3. (k, o)*

cf information element, information space.

Manual dimension. *A type of implemented information space dimension. In manual dimensions, information elements must be manually attached to coordinates—manual dimensions are not associated with algorithms to automatically classify information units. See section 3.3. (k, o)*

cf automatic dimension, dimension, dynamic dimension, index, information space, information unit, static dimension.

Map. *A representation—by way of a graphical display, summary statistics and text—of a region (set of points) in information space. A **map** might display the coordinates in each dimension currently of interest, an indication of which coordinates the user has selected, and a list of information elements in the region selected by the user. Useful **summary statistics** include various counts of information elements, various distances including mean and/or median distances between information element classifications and points selected by the user, proportions indicating the size of the region selected, and the classification of each information element in the region selected by the user. **Text** includes the names of dimensions and coordinates and the titles of information elements. The relationships between coordinates may also be outlined using text. Maps can represent large regions, such as an entire information space, and small regions, such as a single point in an information space. The map of an information element is the same thing as the map of the point to which the information element is attached—it displays, at least, the coordinates of the point. See section 3.3. (k, o)*
cf classification, dimension, information element, information space, space.

Merge.

- i) When sets are generalised so that each element x has an associated multiplicity or “membership”, union can be generalised in two ways to produce two different operators merge and join. The **merge** operator, is often represented by the symbol \oplus . When two “generalised sets” are merged, the multiplicity of each element common to both generalised sets is a function of the sum of the multiplicities of the element in each operand generalised set. See section 6.2.3. (k, o)*
- ii) The merge form of union is defined for L -collections. See section 6.3. (k, o)*
cf join.

Meta Information. *See* classification.

Monothetic classification [scheme]. *A type of classification scheme wherein classes are defined by necessary and sufficient attributes. Objects that belong to a (monothetic) class possess these attributes*[79]. (c, r)
cf polythetic classification.

n -dimensional space. *A space made up from n spaces, called dimensions. See definitions section 4.4.(k)*
cf dimension, space.

Operational database. *A database that records and retrieves the everyday transactions of an organisation*[75]. (d, w)
cf database.

Outline. *See* classification.

Overview. *See* classification.

Point. *See* space.

Polyseme. *A word that has a number of different—but related—meanings.* (r)
cf homonym.

Polythetic classification [scheme]. *A type of classification scheme wherein classes are not defined by necessary and sufficient attributes. Each member of a (polythetic) class may possess only a portion of all the attributes possessed by all the members of that class*[79]. (c, r)
cf monothetic classification.

Precedence dimension. *A type of information space dimension. A distance greater than 0 between coordinates a and b indicates a is knowledge dependent on b and so information elements attached to coordinate a are dependent on information elements attached to b —effectively assigning a partial ordering to information elements. See section 3.3. (k, o)*

cf dimension, distance, information element, information space, knowledge precedence, knowledge requirement.

Precision. *A measure of the effectiveness of an information retrieval system. Given that D is the set of retrieved documents and T is the set of documents in a collection that are relevant to a query, $\text{precision} = \frac{|D \cap T|}{|D|}$. (r)*

cf document, information retrieval system, recall.

Range query. *Given an information space (M, d, \mathcal{I}) , a point $x \in M$ and a limit $t \in \mathbb{R}$, a **range query** $q_{\text{range}}(x, t)$ is the set of all points $\{y \in M^I | d(x, y) \leq t\}$. That is, the range query $q_{\text{range}}(x, t)$ “retrieves” all points $y \in M$ (with information units attached) where $d(x, y) \leq t$. See chapter 10. (m)*

cf Candidate set, k nearest neighbour query, ranked query, retrieved set.

Ranked query. *A ranked query is a k Nearest neighbour (k -NN) query where the resulting set Y is ranked according to $d(x, y)$ for each $y \in Y$. (m)*

cf Candidate set, k Nearest neighbour query, range query, ranked query, retrieved set.

Recall. *A measure of the effectiveness of an information retrieval system. Given that D is the set of retrieved documents and T is the set of documents in a collection that are relevant to a query, $\text{recall} = \frac{|D \cap T|}{|T|}$. (r)*

cf document, information retrieval system, precision.

Record. *A **Record** is a collection of related fields that can be treated as a unit... depending on design, records may be of fixed or variable length. A record will be of variable length if some of its fields are of variable length or the number of fields may vary...[88].* (d, f)
cf field.

Region. *A set of points in information space. Users specify regions in information space by **selecting** coordinates. A region specified by a user in this way is called the user's **position** in the information space.* (k, o)
cf coordinate, information space, point.

Relational database. *A database based on the relational model [of data].* (d)
cf database.

Retrieved set. *The set of points “retrieved” by a range (or k -NN, or ranked) query.* (m)
cf Candidate set, k Nearest neighbour query, range query, ranked query.

Signed distance. *See signed distance function.*

Signed distance function. *A **signed distance function** over M is a function d that maps pairs of elements (points) of a set M into \mathbb{R} . $d(x, y)$ is called the **signed distance** from x to y . The “sign” can be used to indicate direction, in which case $d(x, y) = -d(y, x)$. Each signed distance function has a corresponding distance function $|d(x, y)|$. See definitions section 4.2.1.* (k, o)
cf distance function, space.

Static dimension. *A type of information space dimension. In a static dimension the coordinates and/or distances are not automatically determined. Hence adding information units to the information space can not change existing classifications. See section 3.3. (k, o)*
cf attach, automatic dimension, coordinate, dimension, dynamic dimension, information space, information unit, manual dimension.

Strong Classification. *In strong classification, the classification of information specifies the relative “position” of the information in a “universe of knowledge.” See section 1.3. (c, k, o)*
cf classification, weak classification.

Slice and dice. *The use of arbitrary dimensional constraints to aggregate “fact table” data in a star-join dimensional data mart[75]. (w)*
cf data mart, dimensional data mart, drill down, drill up.

Space. *Mathematically, a **space** is a set M over which a (signed) distance function d is defined. Elements of M are called **points** in the space. See definitions section 4.2.1. (k, o)*
cf distance function, signed distance function.

Specific subject. *“...the subject whose extension and intention exactly coincide with the contents of the book...[76].” (c)*
cf extension and intention.

Star-join schema. *A technique for using relational databases to implement dimensional data marts. The schema consists of a single fact table and a number of dimension tables. The primary key in the fact table is composite and consists of a number of domains that are foreign keys. Each of these foreign keys is the primary key in one of the dimension tables. Domains in the fact table that are not foreign keys are generally numeric and their values can be meaningfully summed*[75]. (d, w)

cf database, data mart, dimensional data mart, relational database.

Summary. *See classification.*

Surrogate. *A surrogate is a “stand-in” for a book. Books may be arranged on the shelves in one way, while the surrogates of books may be arranged in catalogues in another way*[16]. Reitz uses “Bibliographic record” in[78]. (c)

cf collocation.

Top-down classification [scheme]. *A type of hierarchical classification scheme where the root of the classification hierarchy corresponds to the entire universe, with ever smaller subdivisions towards the leaves of the tree*[16]. (c)

cf analytico-synthetic classification, bottom-up classification, enumerative classification.

Weak classification. *Classification to simplify the retrieval of information. Also called indexing. See section 1.3 (c, k, o)*

cf classification, indexing, strong classification.

Bibliography

- [1] Merriam-webster online dictionary, 2004. URL: <http://www.m-w.com>.
- [2] CSIRO library network, 2004. URL: <http://voyager.its.csiro.au/>.
- [3] ISSN international centre, 2006. URL: www.issn.org.
- [4] Benjamin Bustos A, Gonzalo Navarro B, Edgar Chavez C, Ciudad Universitaria, and Mich Mexico. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters*, 2003.
- [5] Rakesh Agrawal, A. Gupta, and Sunita Sarawagi. Modeling multidimensional databases. In *Proc. 13th Int. Conf. Data Engineering, ICDE*, pages 232–243. IEEE Computer Society, 1997.
- [6] Cristian Mendoza Alric and Norma Edith Herrera. Center selection techniques for metric indexes. *Computer Science and Technology*, 7:98–104, 2007.
- [7] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, New York, 1999.
- [8] Rudolf Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1:173–189, 1972.
- [9] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifier (uri): Generic syntax. <http://tools.ietf.org/html/rfc2854>.
- [10] T Berners-Lee, J Hendler, and O Lassila. The semantic web. *Scientific American*, pages 1–18, 2001.

- [11] Tim Berners-Lee and Mark Fischetti. *Weaving the Web*. Harper, San-Francisco, 1999. ISBN 9780062515872.
- [12] Paul Beynon-Davies. *Database Systems*. Palgrave MacMillan, New York, 3 edition, 2004.
- [13] H.E. Bliss. *The organization of knowledge in libraries and the subject-approach to books*. H.W. Wilson company, New York, 3 edition, 1939.
- [14] Bliss classification association. 2001.
URL: <http://www.sid.cam.ac.uk/bca/bchist.htm>.
- [15] Christian Böhm, Stefan Berchtold, and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.
- [16] Vanda Broughton. *Essentail Classification*. Neal-Schuman, New York, 2004.
- [17] Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A Course in Metric Geometry*, volume 33. American Mathematical Society, Providence, Rhode Island, 2001.
- [18] V. Bush. As we may think. *The Atlantic Monthly*, 1945. URL: www.theatlantic.com/doc/194507/bush.
- [19] George Cain. *Introduction to General Topology*. Addison-Wesley, Sydney, 1994.
- [20] G. Cantor. Beiträge zur begründung der transfiniten mengenlehre [contributions to the founding of the theory of transfinite numbers]. *Mathematische Annalen*, 46:481–512, 1895. Translation from German.
- [21] Donald D. Chamberlin and Raymond F. Boyce. Sequel: A structured english query language. pages 249–264. ACM SIGFIDET Workshop on Data Description, Access and Control, 1974.

- [22] E. Chavez, G. Navarro, R. Baeza-Yates, and J. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.
- [23] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *23rd VLDB Conference*, pages 426–435, 1997.
- [24] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.
- [25] E. F. Codd, S. B. Codd, and C. T. Salley. Beyond decision support. *Computerworld*, 27(30):87–89, 1993.
- [26] E. F. Codd, S. B. Codd, and C. T. Salley. Providing olap (on-line analytical processing) to user-analysts: an it mandate, 1993. Technical report.
- [27] D. Connolly and L. Masinter. The ‘text/html’ media type. <http://tools.ietf.org/html/rfc2854>.
- [28] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genet*, (25):25–29, 2000. URL:<http://www.geneontology.org>.
- [29] C.J. Crouch, D.B. Crouch, and K.R. Nareddy. The automatic generation of extended queries. Number 13, pages 369–383. ACM SIGIR conference, 1989.
- [30] CiteSeer.Continuity Database. Most cited authors in computer science - august 2006 (citeseer.continuity), Accessed August 2007. URL: <http://citeseer.ist.psu.edu/allcited.html> Zadeh is number 275 in this list.
- [31] Anindya Datta and Helen Thomas. The cube data model: a conceptual model and algebra for on-line analytical processing in data warehouses. *Decision Support Systems*, 27(3):289–301, 1999.

- [32] M.-P. Dubuisson and A.K. Jain. A modified distance for object matching. volume 1, pages 566–568. the 12th IAPR International Conference on Computer Vision & Image Processing, 1994.
- [33] M.-P. Dubuisson, A.K. Jain, and W.C. Taylor. A vision-based vehicle matching system. pages 266–271. Intelligent Vehicles '94 Symposium, October 1994.
- [34] E.A.Fox. Composite document extended retrieval. Number 8, pages 42–53, Montreal, 1985. ACM SIGIR conference.
- [35] J. Blake et. al. Creating the gene ontology resource: Design and implementation. *Genome Research*, 2001. URL: www.genome.org/cgi/doi/10.1101/.
- [36] R. Fielding, UC Irvine, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol—http/1.1. <http://tools.ietf.org/html/rfc2854>.
- [37] International Organization for Standardization. Website, accessed August 2007. URL: <http://www.iso.org>.
- [38] A.C. Foskett. *The universal decimal classification*. Clive Bingley, London, 1973.
- [39] Eugene Garfield. A tribute to Calvin N. Mooers, a pioneer of information retrieval. *The Scientist*, 11(6):9–12, 1997. URL: [www.garfield.library.upenn.edu/commentaries/tsv11\(06\)p09y19970317.pdf](http://www.garfield.library.upenn.edu/commentaries/tsv11(06)p09y19970317.pdf).
- [40] L.M. Garshol. Metadata? thesauri? taxonomies? topic maps! making sense of it all. *Journal of Information Science*, 30(4):378–391, 2004.
- [41] J. Gehrke. International ISBN agency, accessed September 22, 2003. URL: www.isbn-international.org.
- [42] J.A. Goguen. *l*-fuzzy sets. *Journal of Mathematical Analysis and Applications*, 18(1):145–174, 1967.

- [43] Gisli Hjaltason and Hanan Samet. Index-driven similarity search in metric space. *ACM Transactions on Database Systems*, 28(4):517–580, 2003.
- [44] Gisli Hjaltason and Hanan Samet. Index-driven similarity search in metric space. *ACM Transactions on Database Systems*, 28(4):517–580, 2003.
- [45] David K. Hsiao. ACM transactions on database systems: aim and scope. *ACM Transactions on Database Systems*, 1(1):1–2, 1976.
- [46] Chun-Che Huang, Tzu-Liang (Bill) Tsengb, Ming-Zhong Lic, and Roger R. Gungd. Models of multi-dimensional analysis for qualitative data and its application. *European Journal of Operational Research*, 174(2):983–1008, 2006.
- [47] American Library Association’s ALCTS/LITA/RUSA Machine-Readable Bibliographic Information Committee in conjunction with Network Development and MARC Standards Office Library of Congress. The marc 21 formats: Background and principles, 1996. URL: <http://www.loc.gov/marc/96principl.html>.
- [48] Wikimedia Foundation Inc. Wikipedia, 2001. URL: <http://en.wikipedia.org/>.
- [49] Bill Inmon. The problem with dimensional modeling. *DM Review Magazine*, 2000. URL: www.dmreview.com/article_sub.cfm?articleId=2184.
- [50] William H. Inmon. *Building the Data Warehouse*. John Wiley and Sons, 2005.
- [51] American National Standards Institute. Website, accessed August 2007. URL: <http://ansi.org>.
- [52] Thomson ISI. ISI highlycited.com, Accessed August 2007. URL: <http://isihighlycited.com/> This implies that Zadeh is in the top 250 cited authors in his field.

- [53] B Javidi. *Image Recognition and Classification: Algorithms, Systems, and Applications*. CRC Press, 2002.
- [54] W. Kent. *Data and Reality: Basic Assumptions in Data Processing Reconsidered*. Elsevier Science, New York, 1978.
- [55] David M. Kroenke. *Database Processing*. Macmillan, New York, 1992.
- [56] F.W. Lancaster. *Information retrieval systems: characteristics, testing and evaluation*. Wiley, New York, 1968.
- [57] Chang Li and Xiaoyang Sean Wang. A data model for supporting on-line analytical processing. In *CIKM*, pages 81–88, 1996.
- [58] Kwan-Ho Lin, Baofeng Guo, Kin-Man Lam, and Wan-Chi Siu. Human face recognition usign a spatially weighted modified hausdorff distance. pages 477–480, Hong Kong, May 2001. International Symposium on Intelligent Multimedia, Video and Speech Processing.
- [59] Christian Lindig. Concept-based component retrieval. In J. Köhler, F. Giunchiglia, C. Green, and C. Walther, editors, *Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs*, pages 21–25, 1995.
- [60] Paul Love, Joe Merlino, Jeremy Reed, Craig Zimmerman, and Paul Weinstein. *Beginning Unix*. Wiley, Indianapolis, 2005.
- [61] D. Maddison. The tree of life web project, 2001. URL: <http://tolweb.org/tree/>.
- [62] Davide Maltoni, D. Maio, Anil K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer Professional Computing. Springer, 1st edition, 2003.
- [63] G. Mazzola, G. Milmeister, and J. Weissmann. *Comprehansive Mathematics for Computer Scientists*, volume 1. Springer-Verlag, Berlin, 1998.

- [64] J. Mills. *The universal decimal classification*. Rutgers state university, New Jersey, 1964.
- [65] P. Mockapetris. Domain names—implementation and specification. <http://tools.ietf.org/html/rfc1035>.
- [66] John Muster. *Introduction to UNIX and Linux*. McGraw-Hill Osborne, Berkeley, 2003.
- [67] Gonzalo Navarro. Searching in metric spaces by spatial approximation. *The VLDB Journal The International Journal on Very Large Data Bases*, 11(1):28–46, 2002.
- [68] Library of Congress. The cataloging in publication program, accessed May 2, 2002. URL: <http://cip.loc.gov/cip>.
- [69] Office of declassification. history of classification and declassification, July 1996. URL: <http://www.fas.org/irp/doddir/doe/history.htm>.
- [70] B.I. Palmer. *Itself an education (six lectures on classification)*, chapter The socio-historical background to library classification, pages 7–15. The library association, Kent, 1971.
- [71] Z. Pawlak. Rough sets. *International Journal of Computer and Information Science*, 11:341–356, 1982.
- [72] Z. Pawlak. *Rough sets: Theoretical Aspects of Reasoning About Data*. Kluwer, Netherlands, 1991.
- [73] P.J. Phillips, Hyeonjoon Moon, S.A. Rizvi, and P.J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1090–1104, 2000.
- [74] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Upper Saddle River, NJ, USA, 1993.

- [75] Margy Ross Ralph Kimball. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley and Sons,, New York, 2 edition, 2002.
- [76] S.R. Ranganathan. *Elements of library classification*. Asia publishing house, Bombay, 1962.
- [77] E. Rayner. Developing a method of algorithm classification for computer algorithm library applications, 2003. Honours Thesis.
- [78] Joan M. Reitz. *Dictionary for Library and Information Science*. Libraries Unlimited, Portsmouth, 2004. URL: <http://lu.com/odlis/index.cfm>.
- [79] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, 1979.
- [80] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Sciences*, 27:129–146, 1976.
- [81] R.E. Rubin. *Foundations of library and information science*. Neal-Schuman, New York, 1998.
- [82] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.
- [83] G. Salton. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 37(2):98–108, February 1994.
- [84] G. Salton and M.H. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
- [85] M. Shapiro. The choice of reference points in best-match file searching. *Communications of the ACM*, 20(5):339–343, May 1977.
- [86] Amit Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pages 1–9, 2001.

- [87] American Mathematics Society. 2000 mathematics subject classification, 2000. URL: <http://www.ams.org/msc/>.
- [88] Willaim Stallings. *Operating Systems—Internals and Design Principles*. Prentice Hall, New Jersey, 4 edition, 2001.
- [89] Janet Stevenson. *Dictionary of Library and Information Management*. Peter Collin Publishing, Teddington, 1997.
- [90] E. Svenonius. The epistemological foundations of knowledge representations. *Library Trends*, 52:571–587, 2004.
- [91] A. Kenneth Swanson. Development and management of a computer-centered data base: part 4: A computer-centered data base serving usaf personnel managers., 1963. URL:<http://stinet.dtic.mil> Accession Number:AD0662956.
- [92] Andrew Tanenbaum. *Modern Operating Systems*. Prentice Hall, New Jersey, 2nd edition, 2001.
- [93] D.E. Taulbee. Classification in information storage and retrieval. pages 119–137. the 20th national conference, 1965.
- [94] the Internet Engineering Task Force. Requirements for internet hosts—communication layers. <http://tools.ietf.org/html/rfc1122>.
- [95] Carolyn Begg Thomas Connolly. *Database Systems: A practical approach to design, implementation, and management*. Addison-Wesley, Sydney, 4 edition, 2005.
- [96] UDC Consortium. Universal decimal classification consortium (UDC), 2004. URL: <http://www.udcc.org/>, accessed September 2004.
- [97] Panos Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In *Statistical and Scientific Database Management*, pages 53–62, 1998.
- [98] H. G. Wells. *World Brain*. Meuthuen & Co., 1938.

- [99] R. Wille. *Ordered sets*, chapter Restructuring lattice theory: an approach based on hierarchies of concepts, pages 445–470. Reidel, Dordrecht-Boston, 1982.
- [100] R. Wille. *Formal Concept Analysis*, volume 3626/2005 of *Lecture Notes in Computer Science*, chapter Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies, pages 1–33. Springer, Berlin / Heidelberg, 2005.
- [101] Edward O. Wilson. The encyclopedia of life. *Trends in Ecology and Evolution*, 18:77–80, 2003. URL: <http://www.eol.org/>.
- [102] S.K.M. Wong, W. Ziarko, V.V. Raghavan, and P.C.N. Wong. On modeling of information retrieval concepts in vector spaces. *ACM Transactions on Database Systems*, 12:299–321, 1987.
- [103] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the 4th Annual ACM-SIAM Symposium in Discrete Algorithms*, pages 311–321, 1993.
- [104] L. A. Zadeh. Fuzzy sets. *Inf. Control*, 8:338–353, 1965.
- [105] Hubert Zimmermann. OSI reference model—the OSI model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.