Department of Spatial Sciences

Spatial Semantic Search with Online Agents and Weighted Ontologies

Elizabeth-Kate Gulland

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

July 2017

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material that has been accepted for the award of any other degree or diploma in any university.

Signature: .....

Date: 17 July 2017

Abstract

Information retrieval (IR) is a process of matching an information need to the contents of one or more data resources. This basic concept holds true for all types of search, including Web and spatial searches. Difficulties arise at two points: determining the true information need from a user request, and matching this need to data content.

These issues arise partly because both user and data source have contexts that affect how they interpret information. The goal of this thesis is to improve relevance of online searches, particularly for retrieval of online spatial datasets, by automatically extracting and comparing user and data contexts.

Two aspects of context are of particular interest within this research: terminology and location. Their impact is particularly noticeable in text queries, such as made familiar by search engines including Google, Bing and Yahoo. Simple text-entry interfaces are easier to learn and use than more complex forms that allow users to explicitly specify context about a query.

The flexibility of natural human languages makes textual mismatches between intent and results a common problem. For example, consider the paired terms 'vaccinate'/ 'inoculate', which refer to the same concept, and 'Uluru'/'Ayers Rock', which refer to the same place. A key term search for `"inoculated at Ayers Rock"` would not find data described as `"vaccinated at Uluru"`.

Existing methods applied to the search synonym problem include natural language processing (NLP) techniques and Semantic Web ontologies. NLP can be used to compare text document contents without relying upon exact term matches, and ontologies record meaning as networks of linked concepts, such as terms.

This research aims to semantically index datasets during a search, by using local textual and spatial context in addition to crowdsourced and other public natural language resources to estimate dataset relevance to an information request. For this purpose, methods were explored for the automatic production of contextualised weighted ontologies of terms and spatial proximity measures. A framework for local software search agents with public search interfaces was also explored.

This research contributes to contextual search in four ways, by 1) developing automated

methods to produce a weighted ontology of terms from local content and crowd-sourced text resources, 2) adapting queries to rank text matches by using a weighted ontology of terms, 3) developing automated spatial proximity measures to compare user and data locations, and 4) encapsulating data content and context within online search agents so that they can respond with ranked results to queries from sources with no prior knowledge of the data.

The overall goal is to automatically produce a confidence level in the relevance of a data source and its records to a user query, so that multiple results can be ranked and compared in response to a request for information.

Keywords

Semantic search; contextual search; spatial search; crowd-sourcing; term-relatedness; natural language processing; topic modelling; weighted ontologies; proximity measures; search agents; Web services.

Preface

The research reported in this thesis was supported by the Australian Primary Health Care Research Institute (APHCRI) at the Australian National University (ANU). APHCRI was supported by a grant from the Australian Government Department of Health. The information and opinions contained in this thesis do not necessarily reflect the views or policy of APHCRI or the Australian Government Department of Health.

This work has been supported by the Cooperative Research Centre (CRC) for Spatial Information CRCSI, whose activities are funded by the Australian Commonwealth's Cooperative Research Centres Programme.

Acknowledgements

Many thanks to my supervisors, Dr Simon Moncrieff, Professor Bert Veenendaal, and Emeritus Professor Geoff West at Curtin University, and Mr Paul Konings at the Australian National University (ANU), and to my financial supporters APHCRI and CRCSI.

I'd like to thank Simon, David and many others at Curtin who gave me practical advice about researching, writing, conferences, travel, presenting, and more throughout my studies. Thanks also to the CRCSI P3 gang of PhD researchers and the room 214 posse for their support.

Thank you to my parents, Valerie and David, family and friends who supported me and put up with my absences and distraction during my studies. Melinda helped me keep my senses of humour and proportion (and had great writing tips besides), and fencers from ECU Cavaliers Fencing Club and beyond made it easier to keep my balance — there's nothing like sparring with swords to focus the mind!

Thanks to all the creators of, and contributors to, open source software projects. Many of these projects were of enormous help in my work, including Python libraries as well as applications and frameworks such as Wikipedia Miner, Django, and Weka.

Relevant publications

A number of presentations and papers have been made based on the work presented in this thesis during the project. They are listed below for reference.

Journal articles

Gulland, E.-K., Moncrieff, S., West, G., 2016. Distributed agents for online spatial searches. Spatial Information Research. 24(3), 191–202.

Moncrieff, S., Turdukulov, U., Gulland, E.-K., 2016. Integrating geo web services for a user driven exploratory analysis. ISPRS Journal of Photogrammetry and Remote Sensing. 114, 294–305.

Conference papers

Gulland, E.-K., Moncrieff, S., West, G., 2015. Automated Calculation of Term Relatedness Weights for Semantic Searches. In: the 2015 IEEE/WIC/ACM International Conference on Web Intelligence, Singapore, 6–9 December, IEEE. 1: 313–316.

Reed, T., Gulland, E.-K., West, G., McMeekin, D., Moncrieff, S., 2016. Geographic Metadata Searching with Semantic and Spatial Filtering Methods. In: GEOProcessing 2016: the Eighth International Conference on Advanced Geographic Information Systems, Applications, and Services, Venice, Italy, 24–28 April. 85–92.

Moncrieff, S., Gulland, E.-K., 2015. Dynamic Styling for Thematic Mapping. In: Free and Open Source Software for Geospatial Conference, Seoul, Korea, 16–18 September. 53–67.

Table of Contents

Abstract i

Acknowledgements iii

Relevant publications iv

List of Figures viii

List of Tables x

Table of Code Listings xi

Abbreviations xii

Glossary xiv

1 Introduction 1

1.1 Problem description 1

1.2 Aims and objectives 3

1.3 Approach 3

1.4 Contribution and significance 4

1.5 Thesis outline 6

2 Background 7

2.1 Context 7

2.1.1 Spatial context 10

2.2 Natural language 15

2.2.1 Text sources 16

2.2.2 Natural language processing 19

2.2.3 Term-relatedness 24

2.3 Semantic Web 27

2.3.1 Ontologies 27

2.3.2 Weighted ontologies 31

2.4 Search 33

2.4.1 Information retrieval 35

2.4.2 Spatial search 37

2.5 Software agents . . . 45
2.5.1 Web services . . . 47
2.6 Making and evaluating matches . . . 51
2.6.1 Machine learning . . . 51
2.6.2 Evaluating results . . . 56
2.7 Summary . . . 62

3 Matching terms **64**
3.1 Introduction . . . 64
3.2 Evaluating term-relatedness . . . 65
3.2.1 Relatedness measures . . . 66
3.2.2 Evaluating measures . . . 72
3.3 Discovering related terms . . . 85
3.3.1 Binary classifiers . . . 85
3.3.2 Estimating relatedness . . . 91
3.4 Categorising and clustering terms . . . 98
3.5 Summary and discussion . . . 109

4 Matching texts **111**
4.1 Introduction . . . 111
4.2 Document matching . . . 112
4.2.1 Indexing . . . 112
4.3 Virtual queries . . . 115
4.4 Topic modelling . . . 120
4.5 Summary and discussion . . . 122

5 Matching locations **124**
5.1 Introduction . . . 124
5.2 Retrieving spatial context . . . 125
5.2.1 Spatial overlap index . . . 126
5.3 Measuring proximity . . . 131
5.4 Summary and discussion . . . 140

6 Managing contextual searches **143**
6.1 Introduction . . . 143
6.2 Search coordination . . . 144

6.2.1 Combining measures 148
6.3 Encapsulating search and context 150
6.4 Search agent design 153
6.4.1 Implementation 156
6.5 Case study 162
6.5.1 Extracting context from a query 163
6.5.2 Search process 164
6.5.3 Results 166
6.6 Summary and discussion 168

7 Conclusions and recommendations **170**
7.1 Conclusions 170
7.2 Recommendations 174

References **176**

Appendix A Test Term Set: MixMed50 **196**

Appendix B Test Term Set: WHO-105 **204**

Appendix C DataAgent — Request Parameters **206**

Appendix D DataAgent — Sample Output **207**

Appendix E DataAgent — Code **208**

List of Figures

2.1 Perth boundaries . . . 12
2.2 Bounding box and convex hull (England) . . . 13
2.3 Bounding box over a POI dataset with an outlier . . . 14
2.4 Document as a term vector . . . 20
2.5 Plate notation of LDA model . . . 23
2.6 Relative Google hit counts for two terms . . . 25
2.7 Ontology example . . . 29
2.8 Proximity measures — overlapping . . . 42
2.9 Proximity measures — disjoint . . . 42
2.10 Overlapping regions . . . 44
2.11 Decision tree example . . . 54
2.12 Example of hierarchical grouping of terms. . . . 56
3.1 Chapter 3 summary . . . 64
3.2 Relatedness boxplots — Google-based measures . . . 74
3.3 Histograms of GR measures . . . 74
3.4 Relatedness scatterplots — Google-based measures . . . 75
3.5 ROC for individual relatedness measures . . . 77
3.6 ROC for individual and combined relatedness measures . . . 79
3.7 Relatedness scatterplots — mixed terms . . . 80
3.8 Relatedness scatterplots — health terms (ignoring unknowns) . . . 81
3.9 Relatedness scatterplots — health terms (zeroing unknowns) . . . 82
3.10 Test set relatedness boxplots . . . 83
3.11 Relatedness scatterplots — health terms (with specialist measure) . . . 84
3.12 Relatedness linear regression (CRW) . . . 93
3.13 Regression decision tree . . . 94
3.14 Regression boxplots (CRW) . . . 97
3.15 Relatedness probability scatterplot (*MixMed*-50) . . . 97
3.16 MLP - relatedness category probability (*MixMed*-50) . . . 103
3.17 Extract of *MixMed*-50 1-NGD weighted graph . . . 104
3.18 *MixMed*-50 clustering with CRW . . . 107
4.1 Chapter 4 summary . . . 111
4.2 *MixMed*-50 document similarity . . . 114

4.3 BOW query results by type (WHO-105) 117
4.4 LDA query results by type (WHO-105) 121
5.1 Chapter 5 summary 124
5.2 WA wheatbelt regions 128
5.3 Region overlap and containment 129
5.4 Proximity versus distance 132
5.5 North Perth regions 133
5.6 Bus stops near North Perth 135
5.7 Proximity versus distance for points near North Perth 137
5.8 Public toilets near 'North Perth' suburb 138
5.9 Public toilets near 'Perth North' Public Health Network region 139
6.1 Chapter 6 summary 143
6.2 Contextual search system using agents 145
6.3 Detail of a search agent 154
6.4 UML diagram of DataAgent classes 157
6.5 Search application results 163
6.6 Search application warning messages 167

List of Tables

3.1 Wikipedia relatedness measure performance (*MixMed*-50) 77
3.2 Google relatedness measure performance (*MixMed*-50) 78
3.3 Combined relatedness measure performance (*MixMed*-50) 79
3.4 SVM binary classification over *MixMed*-50 86
3.5 SVM confusion matrices 87
3.6 SVM confusion matrices - health terms 87
3.7 MLP binary classification over *MixMed*-50 88
3.8 MLP confusion matrices 88
3.9 MLP confusion matrices ($T = 0.5$) - health terms 89
3.10 MLP confusion matrices ($T = 0.4$) - health terms 89
3.11 MLP binary classification over health terms 90
3.12 Logistic regression coefficients of *MixMed*-50 relatedness 94
3.13 PCA results over *MixMed*-50: four attributes 95
3.14 PCA results over *MixMed*-50: three attributes 96
3.15 Medical Subject Headings (MeSH) classes for *MixMed*-50 99
3.16 Decision tree tests (*MixMed*-50) 100
3.17 Weka decision tree over *MixMed*-50 101
3.18 Multiclass MLP results over *MixMed*-50 102
3.19 Extract of *MixMed*-50 1-NGD weight matrix 105
3.20 DBSCAN clustering results (preset ϵ) 106
3.21 DBSCAN results (estimated ϵ) 107
4.1 Query results by type 116
4.2 Query targets found, by type 117
4.3 Query and augmented query similarity to *MixMed*-50 articles 119
5.1 Proximity measures 134
5.2 Feature counts at proximity thresholds 136
5.3 Feature counts from dataset context at proximity thresholds 139
C.1 DataAgent request parameters 206

Table of Code Listings

2.1 RDF with SKOS example 30
5.1 Updating a polygon overlap index 128
5.2 Spatial SQL intersection query 130
6.1 RESTful WFS request 159
6.2 RESTful WFSAgent request 160
6.3 BoundaryAgent model code extract 161
6.4 RESTful BoundaryAgent request 161
D.1 JSON results from a search agent 207
E.1 DataAgent Django models 208
E.2 Search agent handler classes 216
E.3 WFSAgent URL 221
E.4 WFSAgent Django View 221

Abbreviations

ABS	Australian Bureau of Statistics
API	Application Programming Interface
BOW	Bag-of-words model
CRW	Combined Resources Weight
DBSCAN	Density Based Spatial Clustering of Applications with Noise
F_1	F-measure
FOI	Feature of interest
GIR	Geographic information retrieval
GIS	Geographic Information System
GML	Geographic Markup Language
GUI	Graphical user interface
HTTP	HyperText Transfer Protocol
ICD-10	International Classification of Diseases and Related Health Problems, 10th Revision
IP	Internet Protocol
IR	Information retrieval
JSON	JavaScript Object Notation
KVP	Key-value pair
LDA	Latent Dirichlet Allocation
MBR	Minimum bounding rectangle
MeSH	Medical Subject Headings
MLP	Multilayer Perceptron
NGD	Normalized Google Distance
NLP	Natural language processing
OGC	Open Geospatial Consortium
OSM	OpenStreetMap
PCA	Principal Component Analysis

POI	Point of interest
QCI	Query Context Integrator
RDF	Resource Description Framework
REST	Representational State Transfer
ROC	Receiver operating characteristic
ROI	Region of interest
SDI	Spatial Data Infrastructure
SNOMED-CT	Systemized Nomenclature of Medical Terms – Clinical Terms
SRS	Spatial reference system
TF-IDF	Term frequency-inverse document frequency model
UMLS	Unified Medical Language System
URI	Uniform Resource Indicator
Weka	Waikato Environment for Knowledge Analysis
WFS	Web Feature Service
WHO	World Health Organisation
WLM	Wikipedia Link-based Measure
WSDL	Web Service Description Language
XML	eXtensible Markup Language

Glossary

Agent — A small piece of software that can carry out a specific task autonomously and can communicate with other agents. In this thesis, agents carry out search tasks.

Corpus — In NLP, a corpus is a set of text documents. Its plural form is 'corpora'.

Deep Web — Part of the World Wide Web that cannot be indexed by traditional search engines, such as databases accessed via online services or Web forms.

Google — In this paper, Google refers to the Web search engine that indexes online resources, rather than the proprietary company that owns it.

Hit count — The estimated number of resources (such as web pages) indexed by a particular search term. Returned by search engines such as Google.

Homonym — (In lexicology) A word or phrase with the same spelling as another but different meaning: 'bank' (financial or river). Some definitions also include words which differ in either spelling (homophones: 'to'/'two'/'too') or pronunciation (homographs: 'bow'/'bow' for decorative knot/obeisance). See also polyseme.

Hypernym — (In lexicology) A word or phrase that is a broader definition of another: 'disease' is a hypernym of 'influenza'. The property is called *hypernymy*. The opposite is a hyponym.

Hyponym — (In lexicology) A word or phrase that is a narrower definition of another: 'influenza' is a hyponym of 'disease'. The property is called *hyponymy*. The opposite is a hypernym.

Machine learning Machine learning is a type of artificial intelligence that allows computer programs to automatically or semi-automatically learn patterns from training data.

Meronym (In lexicology) A word or phrase that describes part of another: 'wheel' is a meronym of 'car'. The property is called *meronymy*. The opposite is a holonym.

Ontology A collection of entities or concepts that are linked to each other using subject-predicate-object triples and are typically described using the Resource Description Framework (RDF).

Polyseme (in lexicology) A word or phrase with the same spelling as another but different meaning, where both have a similar origin: a computer 'mouse' was named after the rodent 'mouse'. The property is called *polysemy*. See also homonym.

Semantic Web An extension of the Web, facilitating machine-to-machine communication of data, that makes use of linked ontologies.

Shapefile A format for disseminating geospatial vector data, consisting of a set of files for geometry, attribute, and other related information such as SRS. An open specification, originally developed by Esri and now a de facto standard used by many applications[1].

Synonym (In lexicology) A word or phrase that has an identical or very similar meaning to another: 'flu' is a synonym of 'influenza' (they are *synonymous*). The property is called *synonymy*. The opposite is an antonym.

Synset A set of terms with similar meanings.

Term a word or brief phrase with a distinct meaning (within a given context). See also 'token'.

Token In NLP, a token is a word or phrase, such as 'doctor' or 'gen-

[1]Sustainability of Digital Formats: Planning for Library of Congress Collections. 2013. https://www.digitalpreservation.gov/formats/fdd/fdd000280.shtml, accessed 24/3/2017.

	eral practice' that represents an individual concept.
Topic	A group of terms that are logically related.
Toponym	(In lexicology) A word or phrase that is the label for a location — a place name. The property is called *toponymy*.
Web	The World Wide Web (WWW) links online global resources including webpages, multimedia, and RDFs via Uniform Resource Indicators (URIs).
Web service	An online resource that returns a result in response to parameters passed over the Internet in a service request.
Weighted ontology	An ontology, e.g. of vocabulary terms, where linked items are associated with a relationship likelihood weight and can therefore contribute to non-deterministic semantic searches. Fuzzy ontologies are weighted ontologies that also incorporate fuzzy logic rules.
Wikipedia	An online encyclopedia whose content is crowd-sourced (that is, it can be edited by any member of the public).
WordNet	A lexical database of English language terms, including semantic relationships between them, developed at Princeton University.

1 Introduction

Searching for answers on the Web has been an essential part of its appeal since its inception (Berners-Lee, 1989). Searches can take many forms, including answering questions, finding informational webpages, tracking down multimedia and connecting to people.

Finding datasets for use in further analysis or reporting carries additional complications, not least because much online data is on the Deep Web that is not indexed by search engines like Google, Bing and Yahoo (Madhavan *et al.*, 2009). With the rise of Web services and applications and an ever-increasing volume of online data, the main issue is less whether data is available and more which dataset should be used.

Judging whether a dataset suits an information need depends upon more than matching query text to data contents: the *contexts* of both user and data are also important. Location and terminology affects outcomes — should a query `"rooms in Perth"` find medical clinics in Western Australia (WA)? Or hotels in Scotland? Resources such as thesauri and ontologies on the Semantic Web can disambiguate many concepts but still miss local subtleties.

This thesis presents methods for describing data source context in the form of spatial proximity measures and weighted ontologies of terms, where ontologies can be thought of as a network of linked concepts. It also proposes online software agents to encapsulate search actions with data content and context. Such an agent can respond to a query with a 0.0...1.0 estimate of the likely relevance of its data and, optionally, a list of data records that can also include relevance weights. This allows potential data sources to be ranked for usefulness in offline or Web applications and services.

1.1 Problem description

Determining what data to analyse is an important first step when processing, visualising or reporting on information. Comparison of online datasets is complicated by contextual differences between data and user, where context can include spatiotemporal setting (location and time), language preferences (including specialist terminology),

user perspective (such as purpose, profession, and expertise level), and technological requirements.

Context for both user and data is important when matching an information need to data. This is a key consideration for searches from simple interfaces, such as a single query text input. The problem is how to *automatically* extract context from both user and data, and then how to compare them.

Two aspects of context are of particular interest in this research: language and location. Even persons speaking the same language, such as English, use different terminologies. Consider a health-related question posed by the general public, town planners, medical personnel, and health researchers: each group is likely to use different vocabularies to describe a similar problem.

Links between concepts can be described in ontologies, by defining links between terms using subject-predicate-object Resource Description Framework (RDF) triples, for example `"Uluru same-as Ayers Rock"`. This poses problems for links of variable strength, such as `"Perth is-similar-to Fremantle"`. How similar are these places? The triple only records the presence of a link, not its strength.

Hence a second search problem is how to extract lexical context from a data source and build it into a resource, such as an ontology, to draw upon in searches. Such a resource also needs to take into account differing strengths of links between terms from different perspectives.

Searches for — or relative to — a specified place depend upon how that place is described in context. A single place can be described by different names and varying boundaries, depending on its currency and intended use. User and data context also affects how queries should be interpreted, for example what is considered 'near' in a query such as `"restaurants near me"`.

Spatial dataset metadata such as its extent is useful in judging its relevance to a user query but does not include record-level detail, such as individual feature geometries. A more accurate search necessitates searching within the resource itself — in other words, looking beyond search indexes and public metadata into the Deep Web.

A final problem is how to build the knowledge necessary for accessing multiple types of data into online searches. Geospatial data, for example, can be sourced from Spatial

Data Infrastructures (SDIs), geoportals, spatial Web service, databases, and files described in different formats. Finding, accessing, and comparing data from such diverse sources is a complex process (Rautenbach *et al.*, 2013).

1.2 Aims and objectives

The aims of this research are to automatically infer linkages between the lexical and spatial contexts of a user query and dataset to estimate a confidence level in its relevance, to use these links as a semantic index for ranking datasets, and to return ranked data records from within a dataset.

These aims lead to the following objectives:

1. Automate the production of a weighted ontology of terms (that is, a network of interconnected terms) for a data resource, incorporating relatedness strengths between terms. As part of this objective, crowd-sourced text resources will be investigated as sources of term-relatedness weights.

2. Rank dataset records relative to an information request, using lexical context including weighted ontologies of terms.

3. Combine spatial context from user, query, and data source locations, including feature-level details from the Deep Web, to inform a spatial search.

4. Develop software agents that, in response to an information request using a common format, link data and user context to return a confidence level in the relevance of the data source. These agents need to be able to use data record level detail rather than relying solely on public metadata (as used in current spatial searches, such as in Web Feature Services (WFS)). The search agent template must also allow for retrieval of a list of records where this is feasible.

1.3 Approach

The approach addressing the research aims was separated into three main tasks:

- Development of lexical and document weights.

- Development of spatial proximity measures.
- Development of a framework of online search agents incorporating relatedness weights.

A ground truth set of paired terms with manually-assigned relatedness strengths was developed for testing. The test set was mixed-domain: general and medical. Semantic measures of term-relatedness were calculated based upon different crowdsourced and other publicly-available text resources. These were compared individually and in combination to the ground truth weights, using machine learning methods for regression, classification, and clustering.

Corpora of Wikipedia article texts were compiled for testing, and relatedness between each document determined using natural language processing (NLP) methods. Search results for queries and virtual queries (adding terms strongly related by the combined measure to query terms) were compared against target articles.

A non-deterministic spatial proximity measure, based on distances and polygon containment, was tested with point of interest (POI) datasets and sample regions. Deep Web record access was used to calculate weights and enable the return of a subset of features from a single query.

A framework for online search agents was developed, incorporating a core of required behaviours, input parameters, and output formats to facilitate contextual searches. The framework was tested with a set of specialised agent types in a case study Web application interpreting text queries to visualise found features on a map and return their place names.

Text query elements including target feature type (such as 'hotel' or 'bus stop'), target place name, and/or spatial operator (such as 'near' or 'within') were extracted. Search results were tested for success or failure, and returned Deep Web records were compared to expected feature results.

1.4 Contribution and significance

The main contributions of this thesis are:

- Methods to estimate term-relatedness weights from a combination of crowd-sourced and other natural language resources, and methods for testing possible measure combinations against a ground-truthed subset of terms.

- Description of a Combined Resources Weight (CRW) measure of term-relatedness for general terminology, which can be updated with specialised terminology measures for domain-specific vocabularies.

- An approach to automatically build locally-relevant, weighted terminological ontologies using crowd-sourced resources. Also, the use of NLP methods to update such ontologies based on local texts.

- Development of a load-time procedure to develop a spatial index of polygons across multiple data sources.

- Creation of a non-deterministic spatial proximity method for ranking spatial features by distance from a region or other spatial feature of interest.

- Development of a framework for online search agents that encapsulates local content and context. A hierarchy of agents is supported, in that an agent can call on one or more other such agents as data sources. They can also be extended to add functionality and cater for new types of data resource. Their efficacy was demonstrated in a Web application that coordinated access to and results from multiple search agents.

The novelty and significance of this research is that it:

- Automates the production of a weighted ontology of terms from crowd-sourced data.

- Describes methods to automatically incorporate multiple perspectives on data into online searches, including searches of spatial data sources.

- Incorporates semantic indexing of a data resource into relevance ranking, including combinations of lexical and spatial indexes. These indexes have potential for knowledge discovery.

- Develops an innovative, extensible framework of hierarchical online software agents that encapsulate local context and facilitate searches by web applications and ser-

vices with no knowledge of data source context.

1.5 Thesis outline

Chapter 2 reviews previous research in the fields of user and data source context, particularly as it relates to information searches; natural languages and natural language processing, the Semantic Web and ontologies (including weighted ontologies), search and information retrieval, software agents and Web services, and finding and evaluating data matches, including via machine learning algorithms. It also introduces terminology and concepts used throughout this thesis.

Chapter 3 presents research into development and testing of a semantic index of terms, relative to data source context. Term-relatedness scores are developed based on crowd-sourced resources Google and Wikipedia, and manually collated expert knowledge sources for general and medical terminology. Calculated relatedness weights are compared to ground-truth weights for a set of general and medical terms. This chapter investigates lexical context at the micro level — relatedness between individual terms.

Chapter 4 extends research from the previous chapter into the macro level — relatedness between documents. It uses NLP methods to compare documents sourced from Wikipedia articles to text queries. This research makes use of the semantic index developed in Chapter 3 to add context to queries by augmenting them with related terms. Chapters 3 and 4 in combination are applied to produce lexical and document weights for search result ranking.

Chapter 5 presents the effects of user and data location context on spatial data searches. It proposes and tests a non-deterministic spatial proximity measure for ranking data features and data sources by their relevance to a query feature of interest (FOI).

Chapter 6 proposes the use of online search agents to encapsulate search capabilities with data source content and context. A framework of search agents with a common set of basic capabilities and responses is described, catering for a hierarchy of agents using one or more agents to source data. The use and coordination of agents is investigated with a case study Web application searching for spatial features.

Final conclusions and recommendations are discussed in Chapter 7.

2 Background

This chapter outlines previous research in fields relevant to this thesis, including elements of user and data source context relevant to spatial semantic search.

Uses of the Semantic Web to describe natural language context are reviewed, along with natural language processing (NLP) methods for estimating term-relatedness strengths for incorporation into weighted ontologies. Previous applications of crowdsourced resources, including Wikipedia and Google, in NLP research are also outlined.

Work on Web search and geographic information retrieval is reviewed, including the use of software agents and Web services to encapsulate search behaviour with data. Other, non-NLP machine learning techniques used in the research for classification, regression, categorisation and evaluating outcomes are also described.

2.1 Context

The Macquarie Dictionary defines *context* as:

> "the circumstances or facts that surround a particular situation, event, etc."[2]

The concept of context is important in a number of research fields, with varying but overlapping definitions in linguistics, psychology and — importantly to this research — computer science. This wide scope and flexibility makes an exact, directly usable definition difficult to define (Bradley & Dunlop, 2005), as detail depends upon applications, users and tasks or goals. A general computing definition is:

> "any information that can be used to characterize the situation of an entity, where an entity can be a person, place, or physical or computational object." (Abowd *et al.*, 1999).

Computer applications, systems, and services described as *context-aware* make use of information about the system, environment, data, or user in order to fulfil a task. As defined by Dey (2001):

[2] "context": *Macquarie Dictionary Online*. 2017. Macquarie Dictionary Publishers, an imprint of Pan Macmillan Australia Pty Ltd. www.macquariedictionary.com.au, accessed 17/03/2017

> "A [computer] system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

Within this thesis, context refers to supplementary information relating to a data resource, user, or query including elements such as vocabulary, spatial location, and technology. Contextual elements outside the scope of this research include time and date, price, and user age or occupation.

Perera *et al.* (2014) outlined and compared six ways to describe context for context-aware systems: key-value pairs (KVPs), markup, such as eXtensible Markup Language (XML), relationship graphs (e.g. recorded in databases), object oriented code, logical rules, and ontologies.

Bhogal *et al.* (2007) described four ways that context is used in information retrieval (IR): personalisation, which uses query history to judge relevance for individuals, link analysis (for example, between webpages), language models, which apply natural language processing (NLP) techniques, and ubiquitous computing, which considers the environment that a query originated from (such as the location of a mobile device). Ruthven (2011) also emphasised the importance of context for determining the information need of an IR query.

An example use of ubiquitous computing is a surveillance system that dynamically adjusts privacy levels in viewed footage, depending upon the current situation of the subject (Moncrieff *et al.*, 2008). Ubiquitous, context-aware systems include mobile applications integrating environmental data from device sensors (Abowd *et al.*, 1999), which may publish their readings as Web services (Perera *et al.*, 2014). All of these context-aware systems continuously monitor sensor and other data to update an application or service on-the-fly, although use of near real-time sensor data is not explored in this research.

Context is used by software agents interacting with the Semantic Web. For example, Huang & Webster (2004) combined user context with two online sources of information — a news stream in Really Simple Syndication (RSS) format and a semantic summary of web site content — in an agent-based system searching for IR matches to a user query.

Computing, linguistic and psychological perspectives on context can contribute to IR related problems in a number of ways, including:

- In usability testing, for the selection of users and tasks to test (Lee, 2013). Here, context refers to the choice of users, their goals, and how the test application is being used.

- In systems and application design, including mobile technologies (Bradley & Dunlop, 2005; Abowd *et al.*, 1999).

- To find related terms and determine how strongly they are related (Zhang *et al.*, 2012; Gulland *et al.*, 2015).

- In visualisation of IR results, including geovisualisation of spatial data (Moncrieff *et al.*, 2016; Tomaszewski & MacEachren, 2012) and representation of spatiotemporal perspectives (Hadlak *et al.*, 2010).

- In user modelling, context describes the system being acted upon, in order to correctly interpret a user profile and actions (Delfos *et al.*, 2013).

- Context can be stored within search histories, for improved information seeking behaviour in the long term or within teams of collaborators (Komlodi & Lutters, 2008).

An important aspect for this research from the context-aware definition is the idea of *relevance*, where context and relevance are closely associated (Bradley & Dunlop, 2005). This is allied with the IR focus on the relevance of results.

Context of data resources, though present, is often overlooked in relation to search problems. The context of a resource is dependent upon a number of factors, including their purpose, intended audience, and how they were collated (Bhogal *et al.*, 2007). Computing context in general can be drawn from sources as diverse as sensor readings, search histories, timestamps, user actions, and data resource content and metadata.

Mismatches between a request for information and results are more likely when perspectives or contexts differ between user and data provider. For example, a medical layperson seeking specialist health information, or a hotel-booking system misinterpreting a place name based on the current location of a user.

The term 'context' has been used to differentiate non-spatial attributes within geospatial datasets — for example, public health data is described in Joshi *et al.* (2012) as consisting of geographic, time, and *context* dimensions. Within this research, however, location is considered part of context. This is a similar approach to that taken in adaptive Geographic Information System (GIS) application research (Delfos *et al.*, 2013; Mac Aoidh *et al.*, 2009; Petit *et al.*, 2008).

2.1.1 Spatial context

Spatial context of a user

The physical location of a user is an important part of their context and can affect what spatial data is considered local (Petit *et al.*, 2007). This is of particular interest in location based services (LBSs) using dynamic locations from mobile devices, for example in targeted marketing. An example from Viktoratos *et al.* (2015) considered location alongside other aspects of user context including age, occupation and gender.

Within this thesis, an *origin* point describes the current location of a user. Potential sources include Global Positioning System (GPS) coordinates from a mobile device (Yi *et al.*, 2009), a computer Internet Protocol (IP) address (Jones *et al.*, 2008), or other possibilities such as member profile settings or web application preferences.

Context also affects perception of distances. Relevant issues include choice of transport mode, such as walking, cycling, driving, or taking public transport, and associated context: Willis *et al.* (2004) demonstrated how elements such as age and mobility affect walking speed, which could in turn influence the perception of nearness.

Queries with "geographic intent" include spatial and non-spatial components (Yi *et al.*, 2009; Jones *et al.*, 2008), although the spatial component may be implied — queries `"restaurants near me"` or even `"restaurant"` imply that the current position of the user is the target location. The theme of a query also affects spatial search parameters, such as maximum distance from the origin (Cai, 2007; Jones *et al.*, 2008) or query region size (Yi *et al.*, 2009). For example, airports are expected to be much further apart than lamp posts.

A *region of interest (ROI)* defines the boundary of a search area. Approaches to defining a query ROI, for example in Web applications Google, Bing, or Apple Maps and Open-

StreetMap (OSM)[3], include: 1) using the current map view, 2) accepting a manually-entered filter from the user, such as a rectangular area, 3) recording a user's default search location to help narrow initial results, and 4) extracting place names from text-based queries.

A place name (also toponym) can refer to different places (London in England, France, Canada, United States of America (USA) and so on); and the same place can have different names, including historical or multilingual names (Uluru/Ayers Rock, New Zealand/ Aotearoa) or spelling variations (Yailoup/Yarloup/Yarloop).

Resources for disambiguating toponyms in text queries include knowledge-based, machine learning, and spatial distribution approaches (Zaila & Montesi, 2015), and NLP methods — specifically, for named entity recognition (Lehmann *et al.*, 2015; Ballatore *et al.*, 2013; Malo *et al.*, 2010).

In addition to the reuse and variability of place names, a single named entity can be described by different geometries because of changes over time, different spatial data collection methods, and varying purposes, e.g. a city as a point of interest (POI), central business district boundary, or broader region such as a suburb, electorate or census district.

An example of different boundaries for a single named entity is illustrated for Perth, WA in Figure 2.1. Note that, as well as gross differences in region size and shape, there are also misaligned borders between federal and state electorates.

Spatial context of the data

Data sources have their own spatial context including toponyms, geometry, such as an *anchor* (a single representative point) and extent, and, in some cases, a *feature type*.

'Feature type' is used in this thesis to describe a general theme rather than individual labels. Examples include 'road', 'hotel', 'bus stop' and 'flu rate', where the latter refers to numeric values per region. Feature type should not be confused with a dataset name, such as the FeatureType parameter in the Web Feature Service (WFS) standard, or

[3] http://wiki.openstreetmap.org/wiki/Main_Page, accessed 16/1/2017.

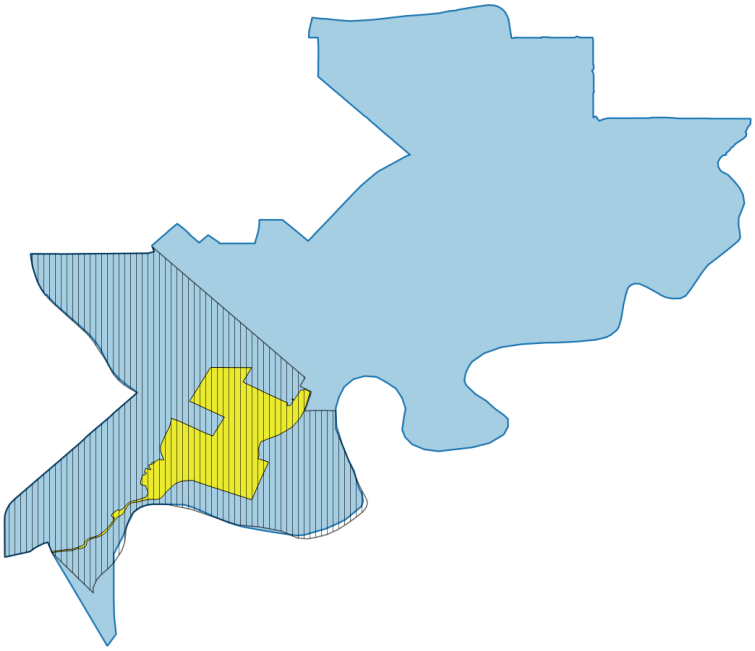


Figure 2.1: Regions labelled 'Perth' in WA. Blue: Australian federal electorate 2016[a]. Vertical bars: WA state electorate 2015[b]. Yellow: WA suburb 2016[b].

[a]data © Commonwealth of Australia (Australian Electoral Commission) 2017, accessed 13/1/2017.
[b]data © PSMA Australia Limited licensed by the Commonwealth of Australia under Creative Commons Attribute 4.0 International licence (CC BY 4.0), accessed 13/1/2017. .

geometry type as used in GIS software[4].

The most apparent geometry-based information in a dataset is the collection of spatial features it contains, whether as point coordinates, more complex geometries, or addresses. In the latter case, a spatial search operation would use a geocoder to translate addresses into coordinates (Waller & Gotway, 2004).

Any spatial data resource can be described by a spatial footprint — a geometry describing the collective location of its features (Ballatore *et al.*, 2013). The extent of a dataset is a geometry containing all of its spatial features such as a minimum bounding rectangle (MBR) or convex hull — the smallest possible container that has no indentations.

The term coverage is not used in this thesis to refer to such a footprint, although it can define "the extent to which something is covered"[5] in a spatial sense e.g. "global coverage" (Ballatore *et al.*, 2013). 'Coverage' was not used because of possible confusion

[4]http://desktop.arcgis.com/en/arcmap/10.5/manage-data/geodatabases/feature-class-basics.htm, accessed 1/3/2017.
[5]"coverage": *Macquarie Dictionary Online*. 2017. Macquarie Dictionary Publishers, an imprint of Pan Macmillan Australia Pty Ltd. www.macquariedictionary.com.au, accessed 2/3/2017

with other meanings, including ontology or search browser scope (Garla & Brandt, 2012; Kozareva & Hovy, 2013; Madhavan *et al.*, 2009), or structured spatiotemporal data such as raster images[6].

In this thesis, unless specified otherwise, the extent and anchor point of a dataset are represented by its MBR and MBR centroid, respectively. An anchor can be used for visualisations, to position a label or represent a dataset as a point on a map. It also provides a target for distance measures between places (Jones *et al.*, 2008).

The spatial extent of a dataset can be described explicitly in metadata, such as the boundaries for England, United Kingdom (UK) illustrated in Figure 2.2. This separate storage allows for quick checks of a dataset against a known location. This is particularly the case for MBRs, as features need only be compared against maximum and minimum coordinates.

Although MBRs are efficient to store and process, they can misrepresent a set of spatial features by overestimating their scope and distribution (Larson & Frontiera, 2004). For example, the MBR around England in Figure 2.2 also covers all of Wales and the Isle of Man plus parts of Scotland, Ireland, Northern Ireland, and France for which there would be no data.

Figure 2.2: England, UK: MBR/bounding box (red outline) and convex hull (black dashed outline).
Map image from Google Maps (11 January 2017).

[6]ISO 19123:2005 https://www.iso.org/standard/40121.html, accessed 2/3/2017.

Outlier features can lead to a false impression the size and extent of a dataset. Consider a POI dataset that contains a single outlier feature, as illustrated in Figure 2.3. This shows the variation in centroid-based anchor points when calculated from the MBR of all points (C), or convex hulls with (B) or without (A) the outlier point. Datasets with outliers, two or more feature clusters, or a concave footprint could define a more appropriate anchor point manually or with methods such as clustering algorithms.

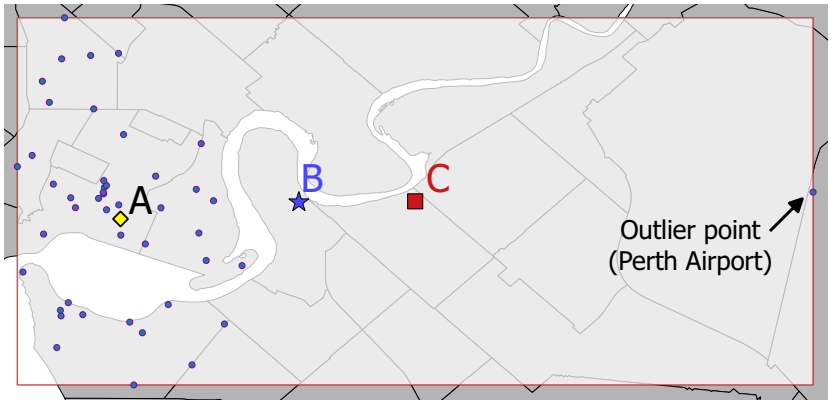


Figure 2.3: Bounding box over a POI dataset[a] with an outlier at right, showing centroids of non-outlier features (A), all features (B), and MBR (C).

[a]WA public toilets whose *Town* attribute contains the text 'Perth', extracted from Australian Department of Social Services (DSS). http://data.gov.au/dataset/national-public-toilet-map, accessed 18/3/2016.

Regardless of the approach used, any pre-calculated extent or anchor point location needs to be maintained as features in the dataset are updated. This means that selection and maintenance of appropriate spatial metadata depends upon the context of the data source, and ideally should be encapsulated with the data rather than implemented by an outside application.

Some spatial dataset providers use metadata templates to encourage data consistency. For example, the Australia and New Zealand Land Information Council (ANZLIC) provides instructions for metadata templates, including a MBR extent[7]. This can lead to datasets with extents much larger than their contents if the default extent is large, such as a nation or state.

Feature density describes the average sparsity of geometrical features within a spatial dataset. Spatial distribution is a measure of the separation of points, for example the average of nearest neighbour distances (Diggle, 2013). This measure is computationally

[7]ANZLIC Report "Creating a template metadata file for cloning for ANZMet Lite v1.0." 2009. http://www.anzlic.gov.au/resources/metadata, accessed 7/3/2017.

expensive as each point is compared to all possible neighbours, so it could either be precomputed and recorded with other data context, or approximated by a simpler representation.

Technical aspects of feature geometry storage, such as the spatial reference system (SRS), resolution, and precision used to define coordinates, also influence options for comparison with other spatial features. Capabilities including available spatial operations also differ across different spatial software and data formats.

In some cases, access restrictions also form part of data context. For example, access may be limited to paying customers or individual records may be blocked to protect privacy. Encapsulating search behaviours alongside the data can alleviate potential issues by providing access to values calculated from data records, without releasing unfiltered content (Moncrieff *et al.*, 2016).

Locally-relevant toponym resources can be sourced from text labels, metadata descriptions, and logs of multi-user queries over the data. Yi *et al.* (2009) used NLP methods to develop "city language models" relating feature type terms and place names, based on query logs over a set of data. They also used a combination of query logs and spatial distances between origin and anchor points to train classifiers differentiating between local, neighbourhood, and other spatial queries. Zaila & Montesi (2015) produced a refinement of the public GeoNames[8] ontology of place names, using Wikipedia and WordNet as sources, and developed a toponym disambiguation algorithm to assist in ranking text documents.

2.2 Natural language

Language is an important part of searches, using text queries and data content as a starting point to match results to an information need.

Natural describes human languages, used for general communication, in contrast to *artificial* languages such as programming languages and XML which have very strict grammatical rules (Zhang *et al.*, 2015). Living languages like English are flexible and change over time and, although they use grammatical structures and rules, texts written in natural languages are considered unstructured data.

[8] http://www.geonames.org, accessed 2/12/2016.

The flexibility of natural languages introduces a number of challenges for automating the processing and interpretation of these texts. The field of study related to these problems is natural language processing (NLP).

Texts contain *terms*, where a term represents a concept and can be a single word (e.g. 'disease') or a phrase of two or more words (e.g. 'infectious disease') (Zhang *et al.*, 2012). Terms are also referred to as *n-grams*, where n represents the number of words within a single term. For example, 'hospital' is a *uni*gram, 'nursing home' is a *bi*gram, and 'Remote Health Centre' is a *tri*gram. In the field of NLP, *token* is also used to refer to a term of interest.

A single term may represent more than one concept, depending upon context such as knowledge domain. Polysemes are terms that have the same spelling but different meanings.

> "... different people with different knowledge sets tend to have different perspectives on the categorization of terms and their linkages and relations." (Li *et al.*, 2014b)

2.2.1 Text sources

A number of text sources have been used in NLP research, including crowdsourced resources, document repositories, and vocabularies. Within this thesis, the words 'terminology' and 'vocabulary' are used interchangeably to describe a lexicon of terms.

Terminologies come in a variety of formats that include databases, ontologies or plain text listings. Different formats are suited for different purposes, for example thesauri can be better suited to text mining than terminological ontologies in some circumstances (Tsujii & Ananiadou, 2005). This helps to explain why some terminology resources, such as WordNet, are available in multiple formats.

Crowdsourced resources are built and maintained by members of the public who explicitly or implicitly update information. Wikipedia, for example, allows anyone to edit an existing encyclopaedic article or add a new one. Google indexes crowdsourced information, such as public webpages, for Web searches.

Test sets of human judgements of relationships between terms or documents have been

constructed for testing NLP tasks. Examples include paired terms in both general (Finkelstein *et al.*, 2002; Agirre *et al.*, 2009) and specialist terminology, notably the biomedical domain (Pakhomov *et al.*, 2010, 2011; Pedersen *et al.*, 2007; Petrakis *et al.*, 2006), as well as full text datasets such as produced by Text REtrieval Conference (TREC)[9] for testing information retrieval (IR) tasks (Büttcher *et al.*, 2010).

Google

Google maintains a search index over publicly available webpages and other crowd-sourced resources on the Web. Results from Google Search include an estimate of the number of indexed resources found that contain the query text - its *hit count* (Cilibrasi & Vitanyi, 2007).

Wikipedia

Wikipedia is a crowdsourced resource where members of the public create and contribute to online articles on a wide variety of subjects. There are many language variants available, with calculations in this research based on the Simple English and English versions. Each article can be seen as descriptive text about a particular concept, labelled with the article title. Wikipedia uses *redirect* links to cater for synonyms. For instance, in the Simple English wiki, the 'whooping cough' article is a redirect link to the 'pertussis' article.

Articles may include *infoboxes* — small summary tables of facts, typically presented at the top right-hand corner. Infoboxes comply with templates that loosely define a format shared by articles with a common theme. For example, infoboxes on geographic location articles can include coordinates, time zone(s), map, list of languages, and statistics such as area and population. Another example is the navigation template, listing related topics at the bottom of an article.

Hyperlinks and infoboxes contribute to the semi-structured nature of Wikipedia that caters for automatic extraction of concepts. It has been widely used within areas of research including artificial intelligence and NLP (Hovy *et al.*, 2013; Medelyan *et al.*, 2009). Hyperlinks between articles mean that Wikipedia is a graph of information, which Franzoni *et al.* (2014) took advantage of in order to find hidden related concepts from

[9]TREC. http://trec.nist.gov, accessed 24/3/2017.

a starting point. The DBpedia structured data resource, for example, is sourced from Wikipedia infoboxes, producing ontologies that form part of the Linked Open Data (LOD) project (Lehmann *et al.*, 2015).

Wikipedia has been used to extract semantic links between concepts in other text corpora, for example by Malo *et al.* (2010) who developed semantic rules based on Wikipedia that were applied to a document-filtering task based on a text query.

WordNet

WordNet is a manually-compiled lexicographic resource for the English language (Miller, 1995; Tangi, 1998). It defines synsets — sets of synonymous terms — and relationships between them, such as hypernym (type-of) ↔ hyponym (is-a), e.g. 'dog' ↔ 'poodle'. Terms can occur in more than one synset, such as 'bank' that can describe a financial institution or land at the edge of a river.

The original WordNet inspired the development of other such *wordnets*, for example in other languages (Dragoni, 2015) or taxonomies, i.e. hierarchies of concepts in a particular domain (Kozareva & Hovy, 2013).

Biomedical and health

Communication within the biomedical and health knowledge domain relies heavily upon specialist terminology and jargon. It is important that medical knowledge, statements and diagnoses are comparable across different practitioners and experts. For this reason, a number of national and international classification systems of medical vocabularies have been produced.

Biomedical terminology resources include the Medical Subject Headings (MeSH)[10] (Petrakis *et al.*, 2006), Systemized Nomenclature of Medical Terms – Clinical Terms (SNOMED-CT)[11], Unified Medical Language System (UMLS)[12], incorporating these and other health vocabularies, and the International Classification of Diseases and Related Health Problems, 10th Revision (ICD-10) taxonomy of terms (Hadlak *et al.*, 2010).

[10] http://www.nlm.nih.gov/mesh/meshhome.html, accessed 1/2/2015.
[11] http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html, accessed 1/2/2015.
[12] http://www.nlm.nih.gov/research/umls/, accessed 7/5/2015.

The ICD-10[13] (World Health Organisation, 2006) is a hierarchical framework for defining medical conditions and is in use around the World. Trained medical coder personnel translate medical statements into formal codes in systems such as ICD-10, e.g. J10.0 "Influenza with pneumonia, seasonal influenza virus identified". The 11th edition is currently under development. ICD-10 is also available as an ontology[14].

Highly specialised language, such as in the health domain, can make it more difficult for non-experts to find relevant information. The rich semantic detail available in authoritative resources, including ontologies, is typically compiled manually by experts and can be problematic for non-experts to interpret and use. Consider using biomedical vocabularies to incorporate medical context into published webpages, data records, and other documents:

> "And while the medical community has invested significant effort in building rich structured ontologies to describe medical knowledge, such structure is today typically available only 'behind the scenes' rather than shared in the Web using standard markup."[15]

2.2.2 Natural language processing

Natural language processing (NLP) is about human-computer interaction: the application of computer algorithms to the interpretation of natural human language. It has been applied to a wide variety of problems, including IR, machine translation, word sense disambiguation, speech recognition, and question answering (Medelyan *et al.*, 2009; Cai, 2007).

In NLP, a *document* is a self-contained text that can be a short phrase (such as a query) or a longer piece of writing (such as an article or story). A *corpus* (or *corpora* in the plural) is a set of documents. For example, Wikipedia can be treated as a corpus containing many article documents.

A corpus defines a *dictionary* of *tokens* (terms) used in its documents. A dictionary may limit the number of words in a term, for example to unigrams only or unigrams and bigrams. Dictionaries usually ignore *stop words* — common words that do not

[13] http://apps.who.int/classifications/icd10/browse/2014/en, accessed 29/10/2013.
[14] https://bioportal.bioontology.org/ontologies/ICD10, accessed 23/1/2017.
[15] http://schema.org/docs/meddocs.html, accessed 5/12/2016.

discriminate well between documents, such as [‘the’, ‘and’, ‘or’, ‘it’] — and may also exclude rare terms, such as those that appear only once across the entire corpus.

Texts (both documents and corpora) can be described statistically, for example using a bag-of-words (BOW) model. This model ignores grammar, punctuation, digits and stop words. Instead, it describes a document by a set of tokens and the number of times they appear. That is, a document is represented as a vector of term frequency (TF) counts, for all terms identified in the corpus dictionary.

For example, the text:

> “Fairy tales are more than true; not because they tell us that dragons exist, but because they tell us that dragons can be beaten.”[16].

can be represented as a BOW document, using a TF vector (Figure 2.4) or a sparse vector of tokens and counts:

[(fairy, 1), (tale, 1), (true, 1), (tell, 2), (dragon, 2), (exist, 1), (beat, 1)].

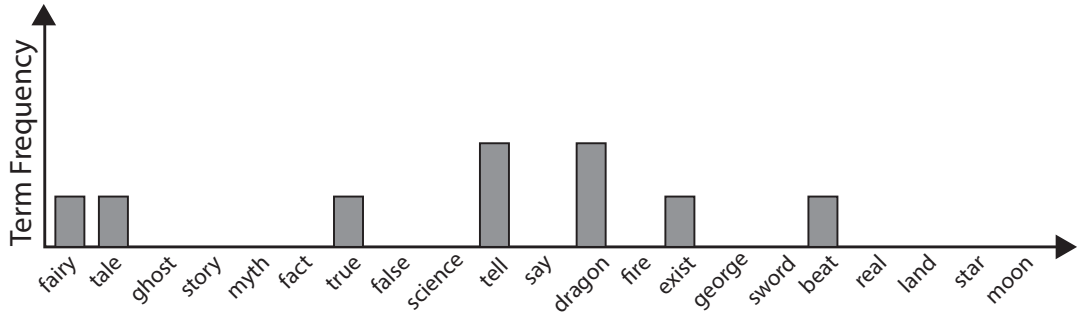


Figure 2.4: Representation of a document as a bag-of-words term frequency (TF) vector

The final token ‘beat’ is a *lemma* of the original term ‘beaten’ — that is, it is the shortest possible logical form of the word. This lemma would be used for all possible versions of the term: [‘beats’, ‘beaten’, ‘beatable’, . . .]. *Stemming*, by comparison with lemmatising, uses predefined rules to remove suffixes without reference to word meaning, so ‘flies’ would become the stem ‘fli’. Corpus dictionaries can be set up to include only lemmatized or stemmed versions of tokens rather than multiple terms for the same underlying concept.

[16]G. K Chesterton, quoted in Coraline (2002) by Neil Gaiman

The simple BOW model is affected by document size because the number and frequency of terms is likely to be larger in longer documents, making documents of different sizes difficult to compare (Li *et al.*, 2014b). To compensate for this, a term frequency-inverse document frequency (TF-IDF) model can be used.

Inverse document frequency (IDF) of a term t is a measure of its rarity within a corpus. Common words such as 'the', which are less useful as discriminators of document meaning, would have a very low IDF value. The IDF measure used in this research is described in Equation 2.1, where D is the set of documents (i.e. the corpus) and n is the number of documents d in D.

$$\mathrm{IDF}(t) = \log_2 \left(\frac{n}{\sum_{i=1}^{n} \mathrm{TF}(t, d_i)} \right) \text{for } d_i \in D \tag{2.1}$$

for corpus D, document d and term t

The TF-IDF measure of term t in document d is a multiplication of TF and IDF, as shown in Equation 2.2. This converts the integer TF vector into a vector of decimals. TF-IDF values are commonly normalised to the $0.0 \ldots 1.0$ range. The measure has been used in many NLP applications, for example to match terms to DBpedia resources (Lehmann *et al.*, 2015).

$$\text{TF-IDF}(t, d) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t) \tag{2.2}$$

for document d and term t

Similarity between two documents can be calculated as the distance between their vectors. In order for similarity values to be comparable, they are calculated over the same type of vector. Each vector of term counts refers to the same list of terms in the same order. Documents being compared are also described with the same model — for example, either BOW or TF-IDF but not a mixture of both.

Non-corpus documents can be compared with corpus documents, as long as they too can be described by the same type of vector. If a document term is not in the corpus dictionary, it is not included in the vector.

Cosine similarity is a common method used in NLP to quantify document similarity

(Li *et al.*, 2014b). It is calculated across two document vectors A and B over n terms as shown in Equation 2.3. A value of 1 means that the vectors are identical. A document similarity index can be precomputed over a corpus by calculating the cosine similarity between every pair of documents and recording these values in a matrix.

$$\text{sim}(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.3}$$

for document vectors A and B

Alternative equations based on Equation 2.3 have been formulated for situations where it is not suited, for example cosine similarity of TF-IDF over a corpus of documents with a wide variety of sizes (Büttcher *et al.*, 2010). In this research, cosine similarity was used without alteration for consistency in comparative results.

There are a number of other NLP models used for analysis of natural language texts, including for estimation of the distribution of themed *topics* across documents in the corpus.

Topic modelling

Topics group logically related items, such as terms in NLP, together. In a corpus of documents, the number of topics is smaller than the number of terms, meaning that descriptive document vectors are shorter than equivalent term-based vectors like TF-IDF. A term can be in more than one topic, due to lexical properties including synonymy, polysemy, and hypernymy.

Topic modelling automates the detection of topics in a corpus. A popular topic modelling algorithm is Latent Dirichlet Allocation (LDA), which estimates probabilistic weights for terms in topics, and for topics in documents (Blei *et al.*, 2003). It generates weight vectors for a predefined number (k) of topics.

LDA was built upon principles from Latent Semantic Indexing (LSI) — also referred to as Latent Semantic Analysis (LSA) — and probabilistic pLSI. LSI uses Singular Value Decomposition (SVD) to reduce the number of dimensions in document vectors, in comparison to the number of terms (Deerwester *et al.*, 1990) and pLSI describes

topics as mixture models of terms (Hofmann, 1999). 'Latent' refers to hidden variables, such as topics, that are assumed to be present but are not directly observable.

Topic models such as LDA consider three levels: items (denoted as 'words') in a topic, topics in a document, and documents in a corpus. LDA generates probabilities for words in a topic and topics in documents, based upon a Bayesian model. It treats a document as a mixture (i.e. a set of probabilities) against a collection of latent topics, and topics as probabilistic distributions against words in the corpus vocabulary.

Figure 2.5 illustrates the LDA generative process in plate notation, for a corpus D containing M documents, each of which consists of N words taken from a vocabulary of V words. In this figure, α and β are prior Dirichlet parameters for document-topic and topic-word distributions respectively, θ is the topic mixture for document $i \in 1..M$ chosen from the Dirichlet distribution $\mathrm{Dir}(\alpha)$, and z is the set of N topics for document i (Blei *et al.*, 2003).

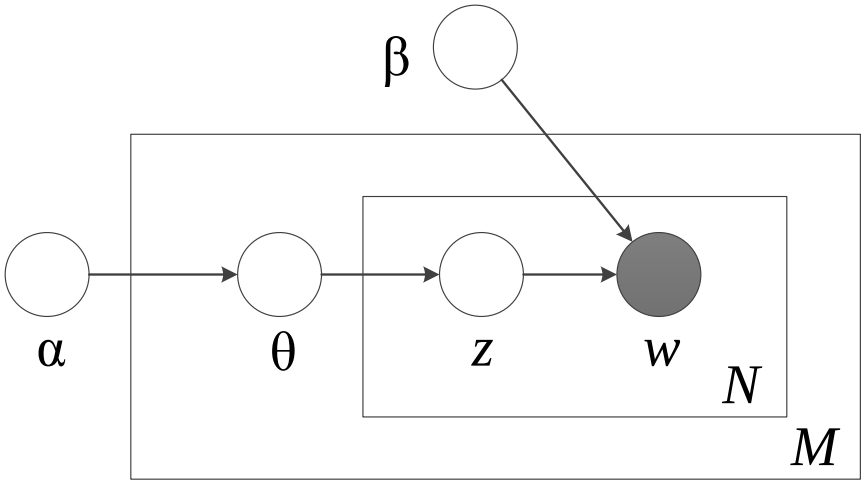


Figure 2.5: Plate notation of the LDA model, after Blei *et al.* (2003). The outer plate represents documents $1..M$ and the inner plate represents topics $1..N$ for document i. Only w_{ij} (the jth word in the ith document) is directly observable: all other variables are latent.

After a number of iterations, a trained LDA model represents each topic as a probability distribution for all vocabulary words, and each document as a probabilistic vector of topic weights. New documents containing words from the corpus's vocabulary can be converted by the same LDA model to vectors of topic weights. Document vectors produced by a LDA model can be compared with cosine similarity.

Selecting the number of topics k to allocate whilst training a LDA model over unstruc-

tured text documents is notoriously difficult, with a number of proposed heuristics including Symmetric Kullback-Leibler (KL) divergence (Arun *et al.*, 2010), topic coherence (Mimno *et al.*, 2011), Bayesian models (Mimno & Blei, 2011), and rate of perplexity change (Zhao *et al.*, 2015).

Li *et al.* (2014a) investigated alternative methods of sampling topics and words in order to improve efficiency of LDA and other topic modelling algorithms.

As intended by its developers Blei *et al.*, LDA is not limited to text documents: topics can describe groups of any related, discrete items. An example of a LDA application beyond text-based document topics is the detection of repeated user behaviours in search logs (Li *et al.*, 2017).

In the spatial field, text-based LSI has been used to detect topics in geospatial data in order to improve the recall rates for IR over spatial datasets (Vockner *et al.*, 2013; Li *et al.*, 2014b). Adams & McKenzie (2013) combined topics from a text-based LDA on travel blogs containing locations to automatically produce models of places.

2.2.3 Term-relatedness

Semantic *similarity* of a pair of terms measures their synonymy — that is, how closely their meanings match. However, related terms are not necessarily synonymous: consider 'flu' and 'vaccination', where an interest in one term could imply interest in the other. Semantic *relatedness* also encompasses other types of semantic relationships including polysemy or homonymy, hypernymy/hyponymy, and meronymy (definitions of these relationships can be found in the Glossary).

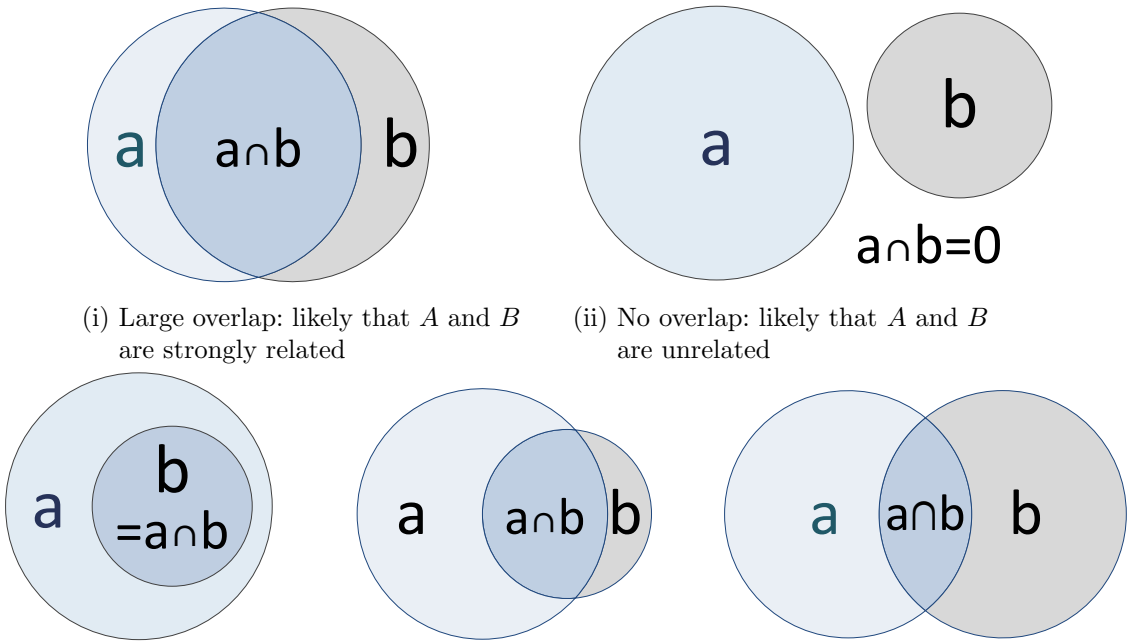
Relatedness between terms is normally calculated as symmetric, although this is not always the case in natural language. In hierarchical term definitions, the relationship from a hyponym to a hypernym is usually perceived as stronger than the reverse — consider `"measles"` to `"disease"`, for example. Complications are also caused by polysemy, where one term can refer to more than one concept (Zhang *et al.*, 2012).

A semantic *distance* measures the semantic relatedness between two terms, where closely related terms have a small distance: perfect synonyms have a semantic distance of zero. This is the inverse of a semantic relatedness *weight* that increases with

relatedness (Budanitsky & Hirst, 2006; Cilibrasi & Vitanyi, 2007): perfect synonyms have a weight of one for relatedness weights in the range 0.0...1.0.

Judging how closely two terms are related involves an element of subjectivity due to the flexibility of natural languages. Consider: is 'doctor' more or less related to 'hospital' than 'x-ray' is? How closely related is 'house' to 'mansion'? Nevertheless, statistical methods can be used to calculate a paired-term relatedness weight that can be tested against subjective human judgements.

Statistical measures of relatedness between paired terms calculated in Google are based upon individual and combined hit counts of the terms. To compare two terms A and B to each other, it is necessary to also consider the intersection of the two: those resources that are indexed by both A and B. Consider Figure 2.6, where a and b represent hit counts for paired terms A and B respectively and $a \cap b$ represents their intersection: the hit count for resources containing both terms A and B. The hit count for the total number of pages referred to by the combination of A and B is the union: $a \cup b$, calculated as $a + b - (a \cap b)$.



(i) Large overlap: likely that A and B are strongly related

(ii) No overlap: likely that A and B are unrelated

(iii) All B resources are also indexed by A

(iv) Most B resources are also indexed by A

(v) A and B are equally related

Figure 2.6: Representation of relative Google hit counts for two terms A and B.

The Normalized Google Distance (NGD) measure proposed by Cilibrasi & Vitanyi (2007) is calculated as shown in Equation 2.4, where a and b are hit counts for terms A and B, and N is a normalising factor approximating the total number of pages in-

dexed by Google. NGD is a symmetrical measure: that is, the weight value is the same for terms $A \rightarrow B$ and $B \rightarrow A$, irrespective of their relative hit count magnitudes. In the scenarios described in Figures 2.6iii and 2.6iv, for instance, the relatedness weight $W_{ab} = W_{ba}$.

$$\mathrm{NGD}(A, B) = \frac{\max\left(\log(a), \log(b)\right) - \log(a \cap b)}{\log(N) - \min\left(\log(a), \log(b)\right)} \tag{2.4}$$

for terms A,B, hit counts a,b, and normalising factor N

The Wikipedia Link-based Measure (WLM) (Milne & Witten, 2008) combines two measures: a vector similarity measure based on TF-IDF (Gabrilovich & Markovitch, 2009), replacing terms with Wikipedia article links, and a statistical distance measure based on NGD, replacing hit counts with the number of inter-article hyperlinks.

The WLM measure has been used in a variety of experiments including quantifying term-relatedness weights for document filtering (Malo *et al.*, 2010), detecting hierarchical links between concepts (Neuman *et al.*, 2013), and discovering relationship strengths between terms (Jabeen, 2014).

A disadvantage of the WLM is that it presumes each test term is a Wikipedia article label, such as a title or hyperlink text that links to the article (Milne & Witten, 2013). This causes limitations when a concept does not match to an article label in the version of Wikipedia used for calculations.

An example of an alternative way to measure term-relatedness based on Wikipedia is via graph path-finding methods, based on article hyperlinks (Franzoni *et al.*, 2014).

Google and Wikipedia are not the only general language resources used to measure term-relatedness in NLP research, however. Other sources include the original WordNet and other wordnets (Petrakis *et al.*, 2006; Kozareva & Hovy, 2013) and domain-specific resources, for example in the biological and medical field (Hassanpour *et al.*, 2014).

A number of measures of similarity or relatedness between biomedical terms have been developed, many of which use resources such as MeSH and/or SNOMED-CT. Garla & Brandt (2012) compared similarity measures based on different biomedical resources including SNOMED-CT and UMLS. The efficacy of biomedical-specific measures such as UMLS::Similarity and UMLS::Relatedness on health texts is discussed in previous

research including McInnes *et al.* (2009, 2013).

Relatedness-estimating techniques designed for general vocabulary are also applicable to specialist domains, for example by combining results across different terminological resources (Petrakis *et al.*, 2006). An example using term measures across domains is outlined by Dong *et al.* (2010), who used ontologies to compute term similarity and disambiguate medical queries from users lacking expert knowledge in the field. Rinaldi (2009) also used ontologies as sources for calculating relatedness weights in his semantic IR system for cross-domain queries.

Term-relatedness has also been applied to the problem of interpreting meaning in spatial data, for example in Janowicz *et al.* (2011) and Li *et al.* (2014b).

2.3 Semantic Web

In general terms, *semantics* within logic and linguistics research refers to *meaning*[17]. The Semantic Web describes the incorporation of semantic meaning into the Web so that computers can extract information from and communicate directly with each other with little or no guidance from human users (Berners-Lee *et al.*, 2001).

Structured information is a necessary part of this automation, and *ontologies* are a common way to provide information in a machine-readable format. Other structured and semi-structured formats are also used for automated information extraction, for example Wikipedia (Hovy *et al.*, 2013).

Section 2.3.1 describes ontologies, which define relationships between concepts, and research into their use. Section 2.3.2 extends this concept into weighted ontologies by including relationship strengths with concept links.

2.3.1 Ontologies

Ontologies are used to describe semantic concepts in a particular domain, including how they relate to each other and what facts are known about them (Gruber, 2009). For example, ontologies can be used to describe people (FOAF[18]), biomedical information

[17] "semantics": *Macquarie Dictionary Online*. 2017. Macquarie Dictionary Publishers, an imprint of Pan Macmillan Australia Pty Ltd. www.macquariedictionary.com.au, accessed 4/4/2017

[18] Friend of a Friend ontology http://xmlns.com/foaf/spec/, accessed 9/5/2017.

such as genetics (GO[19]) (Stevens & Lord, 2009), and geospatial features (Zhao *et al.*, 2012).

The term 'ontology' is also sometimes used in NLP research to refer to a vocabulary, such as a hierarchical taxonomy or terminology used within a particular knowledge domain (Petrakis *et al.*, 2006). In this thesis, unless specified otherwise, ontologies refer to resources used to disseminate machine-readable information over the Semantic Web.

Conceptually, an ontology is a directed, unweighted graph of concepts, typically described by the Resource Description Framework (RDF) model. Within this model, a concept or entity is assigned a Uniform Resource Indicator (URI) defining where to find it on the Web and links between concepts are described using subject-predicate-object triples. An example triple is `"pertussis is-a bacteria"` where '`pertussis`' is the subject and '`bacteria`' is the object. Entities can be grouped into classes and may also be assigned properties, which do not have separate URIs. More information about ontologies and RDF is available at the World Wide Web Consortium web pages[20].

A number of formatting standards can be used to define RDF ontologies, including RDF/XML and Web Ontology Language (OWL) (W3C, 2014, 2012). Valid classes and properties for a RDF ontology are described by schema (RDFS). RDF data can be queried using SPARQL Protocol and RDF Query Language (SPARQL).

Ontologies can describe any type of concepts, including terms in a vocabulary or thesaurus. The Simple Knowledge Organization System (SKOS)[21] schema defines class and property types that are useful for terminologies, including concepts, labels and semantic relationships. Using a common framework such as SKOS simplifies connections between ontologies, for example by adding recognised types of predicate that link to related term concepts in different ontologies.

Figure 2.7 illustrates a set of triples from a hypothetical *medexample* ontology. An extract from a RDF/XML document defining it is shown in Listing 2.1. This extract describes a hierarchy of three medical concepts — `"pertussis"`, a type of `"bacteria"` that is itself a type of `"pathogen"`.

[19]Gene Ontology `http://www.geneontology.org`, accessed 9/5/2017.
[20]RDF `https://www.w3.org/standards/techs/rdf#w3c_all`, accessed 20/4/2017.
[21]`https://www.w3.org/TR/skos-primer`, accessed 20/4/2017.

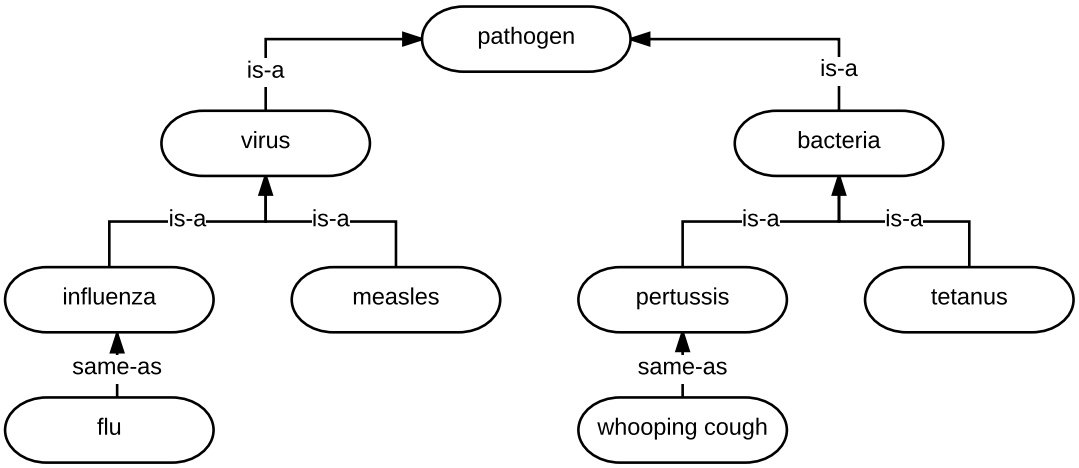


Figure 2.7: An ontology of terms, with concepts represented by ellipses and predicates by the edges that link them.

Note in the RDF/XML listing that lines 2–4 define *namespaces*: abbreviated references to the base URI for an ontology, such as `skos` for SKOS and `me` for the medexample ontology. This example illustrates both properties (such as the `prefLabel` property '`pathogen`' in line 6), and entities (such as the concepts at lines 5–11 and 20–26).

Listing 2.1 demonstrates semantic relationships such as hyponym and hypernym between terms (`skos:broader` and `skos:narrower`, in lines 9, 10 and 23). More general semantic relationships can be represented by a `skos:related` property, as shown in lines 24–25. However, although a link can be made, its strength is not defined: '`cough`' is a more important symptom of pertussis than '`fever`', but this cannot be determined from this RDF.

Terminology ontologies vary in size, and many are domain-specific (Kozareva & Hovy, 2013), as illustrated for health in Listing 2.1 (for information about existing health-related ontologies, refer to Section 2.2.1).

In text-based search problems, the issue of mismatches between a query and resource metadata is an ongoing issue. This applies regardless of the wide variety of searchable resources, such as databases, webpages, online articles, spatial data, Web services, or text files.

This problem can be alleviated by using terminological ontologies to augment query terms. This effectively supplements the available metadata to increase the likelihood

Listing 2.1. Extract from hypothetical medexample (`me`) ontology, using SKOS, in RDF/XML format.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
       ns#"
3           xmlns:skos="http://www.w3.org/2004/02/skos/core#"
4           xmlns:me="http://www.me.com#">
5    <skos:Concept rdf:about="http://www.me.com#pathogen">
6      <skos:prefLabel xml:lang="en"> pathogen </skos:prefLabel>
7      <skos:prefLabel xml:lang="it"> patogeno </skos:prefLabel>
8      <skos:definition> An agent, such as a micro-organism, that
           causes disease </skos:definition>
9      <skos:narrower rdf:resource="http://www.me.com#bacteria" /
           >
10     <skos:narrower rdf:resource="http://www.me.com#virus" />
11   </skos:Concept>
     ...
     ...
20   <skos:Concept rdf:about="http://www.me.com#pertussis">
21     <skos:prefLabel> pertussis </skos:prefLabel>
22     <skos:altLabel> whooping cough </skos:altLabel>
23     <skos:broader rdf:resource="http://www.me.com#bacteria" />
24     <skos:related rdf:resource="http://www.me.com#cough" />
25     <skos:related rdf:resource="http://www.me.com#fever" />
26   </skos:Concept>
     ...
50 </rdf:RDF>
```

of a successful match. Suitable manually-maintained terminological ontologies exist in a number of fields. However, finding a relevant ontology — if one exists — is itself an obstacle.

Automated generation of resource-based ontologies allows for metadata extension without finding and linking to a relevant external ontology. To automatically build a terminological ontology for a resource, linked pairs of terms are sought within relevant texts, such as the articles, tables, pages, or documents of the resource. Kozareva & Hovy (2013) approached this problem by looking for hierarchical 'type-of' patterns in text sources, starting at a high-level general term such as 'animal'.

RDF ontologies are examples of structured datasets that can be automatically linked together. Public ontologies GeoNames and DBpedia, for example, are often linked with other data sources (Lehmann *et al.*, 2015).

Ontology matching is the process of aligning two or more ontologies by adding relationships between them, for example by building RDF triples where the subject and object

are from different ontologies. Manual techniques are slow and can miss or mis-label connections, and hence automated methods are a goal in IR systems. Using the Semantic Web to contextually match user queries and data sources is, in essence, developing a large, universal resource on-the-fly (Caliusco & Stegmayer, 2010).

Potential disadvantages of ontologies for conveying information include:

- Complex processing can be required to extract information from ontology formats (Perera *et al.*, 2014).
- Conflicting demands — *lightweight* ontologies tend to be more widely used but *complete* ontologies support more activities (Wang *et al.*, 2012).
- Domain specificity. Whilst there are a number of *upper* ontologies designed to apply across many fields of knowledge (Sah & Wade, 2016), *domain* ontologies may be applicable only in a subset of topics. Examples of domain ontologies include geospatial (Zhao *et al.*, 2012; Li *et al.*, 2011) and biomedical (Pesquita *et al.*, 2014, 2009; Garla & Brandt, 2012). Linked data approaches can help to offset this problem in many cases.
- Binary links between concepts. Defining a link as a subject-predicate-object triple does not facilitate specification of its strength or likelihood. Also, absence of a link does not mean no such link exists — only that it is not documented.

The latter item can cause problems in IR, as a search across binary links is deterministic. That is, the same question posed to the same ontologies would return the same answers, whereas search results should be ranked by relative strengths (Janowicz *et al.*, 2011). This disadvantage of traditional ontologies can be allayed by using *weighted* or *fuzzy* ontologies.

2.3.2 Weighted ontologies

An issue with standard ontologies is that they document the presence of a link, but make no allowance for relationships with an element of uncertainty, such as terms with a high probability — but not a certainty — of being related. Including weighted links has the potential to cater for more flexible ranking of search results.

Ontology matching techniques also have an element of uncertainty, particularly over a wide scope encompassing ontologies from different domains. Including probabilities for the existence of inter-ontology relationships can improve the validity of linking two or more ontologies (Zhang *et al.*, 2015). Weighted links are also useful within automatically-produced ontologies, such as for geospatial metadata (Sun *et al.*, 2014).

Text-based standards for recording ontologies such as RDF/XML are not well suited to weighted or probabilistic relationships: they were designed to note the presence of a relationship, not its strength. However, different approaches to add weights to ontological links exist, for example in Lacasta *et al.* (2010, Chapter 2). Lacasta *et al.* defined concept links as entities so that a probability weight could be attached as a property.

Cesarano *et al.* (2003) used a Semantic Knowledge Base (SKB) in their IR web browser to rank retrieved webpages relative to the knowledge domain of a query. The SKB contained semantic networks — weighted, directed graphs of concepts — for each domain of interest. In essence, these are weighted ontologies.

Bast *et al.* (2015) developed methods to estimate how strongly an entity belongs to categories, for example, individual persons and professions, based on information in Wikipedia. This resulted in a weight for an ontology-like triple, such as their example `"Tim Burton has-profession Director"`, although their paper focussed on weight calculations.

Combining fuzzy logic principles with ontologies is an ongoing area of research (Tho *et al.*, 2006). Managing fuzzy relationships is complex and requires some type of reasoner. A number of fuzzy reasoners have been developed in recent years, including some specialising in fuzzy ontologies, e.g. ONTOSEARCH2, fuzzyRDF and fuzzyDL, each developed from 2009 onwards (Bobillo & Straccia, 2016).

Ontologies of terms have been used for a number of search tasks, including as a source of alternative terms for query expansion (Jimeno-Yepes *et al.*, 2010; Bhogal *et al.*, 2007). Further background about text and spatial searches is outlined in the next section.

2.4 Search

The greater the volume of available data, the harder it is to find *relevant* data. Search has therefore long been of interest in the computing industry and solutions have been applied to many different information sources, including databases, email systems, collections of academic articles and webpages (Büttcher *et al.*, 2010). Search is a broad term in computing, relevant to tasks including browsing and information retrieval, with specialisations including Web search, and semantic search.

Online searches are often initiated through a Web browser interface to a search engine such as Google, Yahoo or Bing. User searches have different motivations, including navigating to a specific, known target, browsing to finding information about a particular topic, or discovering where to activate an online transaction (Broder, 2002).

Search engines typically use hyperlinks to index public webpages, but cannot index data that is accessed on-the-fly, such as via Web forms or services. These additional online information sources are referred to collectively as the *Deep Web* (Bergman, 2001).

Other sources of structured and semi-structured online data from outside the Deep Web include tabular data, knowledge sources such as DBpedia (providing formatted, crowd-sourced information from Wikipedia), visualisers allowing users to interact with multiple data sources, and discussion forums (Madhavan *et al.*, 2009).

Users can be assisted in searches by providing them with access to categories within the data, such as topics or themes (Hearst, 2006). Automated clustering methods, described in Section 2.6.1, are one way to detect related items to group together.

Faceted searches allow users to refine their search results by selecting categories. Facets are mutually exclusive, logical filters that can be applied to the data, such as make and colour for vehicles. They can also be nested hierarchically, e.g. vehicle make and model. Facets are particularly useful for exploratory searches (Lehmann *et al.*, 2015). Text classification techniques can be useful for discovering facets to apply to a set of data (Gabrilovich & Markovitch, 2009). Hahn *et al.* (2010) made use of Wikipedia links and elements to add facets to search.

Traditional text-based search uses keyword or NLP searches over documents, such as webpages, which are often pre-indexed by a search engine using additional informa-

tion such as hyperlinks. *Semantic* search focusses on the meaning behind the text, and is associated with the Web, and the Semantic Web in particular (Uren & Motta, 2006).

Semantic search can specifically investigate ontologies (Ding *et al.*, 2004; Lei *et al.*, 2006; Blanco *et al.*, 2013) or use ontologies to narrow the context of a query or add meaning to other search results (Guha *et al.*, 2003). An example of the latter is Entity Linking (EL), where persons, places, or other entities are identified in natural language text and linked to a knowledge base such as an ontology or Wikipedia (Yamada *et al.*, 2014). Ontologies can also be applied to data mining and knowledge discovery tasks, either generally or in specific domains (Ristoski & Paulheim, 2016).

Potential issues with semantic search include low precision and high recall such that a large number of irrelevant results are returned, difficulties updating ontologies to keep pace with new information, and misunderstandings between ontology designers and users, due to flexibility in natural languages (Dong *et al.*, 2008).

One way the Semantic Web can be used in searches is *query expansion*, where an initial text query is augmented with related terms, such as found in many ontologies. Bobed & Mena (2016), for example, matched query terms to semantic concepts in ontologies to produce disambiguated queries describing goals more precisely. The user then selected the most appropriate of these queries to use in a search.

Query expansion aims to reduce the impact of a query-information need gap (Crabtree *et al.*, 2007) by adding new, relevant terms to a textual query before searching. This can disambiguate meanings of the original terms, add context, and potentially increase the number of useful results. Query expansion tends to be most effective on shorter queries that cannot include as much context (Jimeno-Yepes *et al.*, 2010). New terms can be selected manually, automatically, or semi-automatically (Bhogal *et al.*, 2007). A similar process can be used to select the correct word sense or named individual from a set of possibilities (Zhang *et al.*, 2012; Minkov *et al.*, 2006).

Related terms can be derived from existing lexical resources, such as WordNet, ProBase[22] knowledge taxonomy (Wang *et al.*, 2017) or terminological ontologies (Bhogal *et al.*, 2007). Related terms can also be discovered using NLP techniques (Section 2.2.2).

[22]Microsoft Concept Graph, built on ProBase. https://concept.research.microsoft.com, accessed 10/4/2017.

2.4.1 Information retrieval

Conceptually, information is processed from data, which are facts without context that are both more plentiful and less valuable than information. Conversely, knowledge is a level above information, as described in Rowley (2007). Both data and knowledge have equivalent areas of research: data retrieval focusses on querying structured data such as databases, and knowledge retrieval works with processed information, such as semantic ontologies (Section 2.3) and large geospatial data sources (Guo & Mennis, 2009).

Information retrieval (IR) is concerned with finding and extracting relevant information from a large set of data, frequently text documents in a natural language such as English (Büttcher *et al.*, 2010). IR is a common search task, starting with a text query or example and returning a set of results ranked by relevance to this input. It is also used to search non-textual data such as multimedia (Ren & Bracewell, 2009).

IR search results are ranked by likely relevance to the *information need* of a user, i.e. the goal they seek when entering a query (Büttcher *et al.*, 2010; Stojanovic, 2005). Relative ranking is used rather than returning a single 'best' result due to the subjective nature of interpreting a request: ultimately, the user chooses from the highly-ranked options provided to them.

The subjective nature of IR requests is caused by differences in context between user and data (Ruthven, 2011). Queries are typically too short to fully express an information need and can use a different vocabulary than the data being searched.

A number of methods to weight results for ranking have been developed, including probabilistic methods that judge likely utility (Robertson, 1977) and PageRank for webpages (Page *et al.*, 1999), which underpinned early successes of Google search and was the first of many hyperlink-based ranking algorithms used by search engines (Wu *et al.*, 2008). PageRank makes use of hyperlinks between webpages to rank their relative importance. These values are not probabilistic — they are used for relative ordering, not as stand-alone estimates of worth.

Ranking search results by relevance or popularity is not the only approach to search, and is not suited to all situations. Noel *et al.* (2014), for instance, developed a Web browser with an emphasis on novel results for sensitive and unpopular topics. They made use of user context and a source of concepts (DBpedia) to expand the initial

query and direct a Web crawler towards potentially useful data sources.

Tasks related to IR, though not necessarily classic search problems, include information extraction (IE), information filtering (IF), summarising or categorising information, and question answering (Büttcher *et al.*, 2010).

Information extraction focusses specifically upon the extraction of structured information from unstructured data such as text (Huang & Webster, 2004; Freitag, 2000). Named entity recognition is an example of IE. One approach used by IE and some NLP applications to extract meaning from text is detecting parts of speech, such as nouns and verbs (Budanitsky & Hirst, 2006; Pedersen *et al.*, 2007; Gabrilovich & Markovitch, 2009), although this is outside the scope of this thesis.

In information filtering, a large volume of items is reduced to a smaller set by removing less relevant items. In contrast to search, which attempts to return the best one or more results, filtering aims to reduce information overload for data explorers. *Recommender systems*, for example, suggest items with similar user ratings or content to a previously-selected item (Vockner *et al.*, 2013; Herlocker *et al.*, 2000). Jansen *et al.* (2006) tested recommender agents as part of their investigation into using software agents for Web searches. This thesis focusses upon online search rather than filtering, particularly for retrieval of online datasets, although relevance ranking can be used for both.

The Semantic Web (Section 2.3) also contributes to the field of IR, for example, by expanding queries to better represent an information need in context (Bhogal *et al.*, 2007; Stojanovic, 2005) or augmenting metadata from ontologies (Ren & Bracewell, 2009). In the opposite direction, Jimeno-Yepes *et al.* (2010) made use of context to refine terminological ontologies.

The primary target for search in this thesis is online datasets, including spatial feature sets. Online data is available from the Shallow and Deep Web, where the Shallow Web includes static webpages that can be directly indexed by traditional Web search engines such as Google and Yahoo via hyperlinks.

Data in the Deep Web, by contrast, is underrepresented in these indexes (He *et al.*, 2007). Madhavan *et al.* (2009) described deep data as hidden behind Web forms although Bergman (2001) described it more generally as being accessed dynamically: for example, extracted from online databases upon request. Hence in this thesis, Deep Web

is taken as inclusive of data retrieved from Web services, as outlined in Chun & Warner (2008).

Processing techniques for IR include statistical methods and machine learning algorithms (Section 2.6). IR systems are frequently paired with IE methods that build relevant indexes from unstructured data to improve search efficiency (Jimeno-Yepes *et al.*, 2010).

Evaluations of IR systems often use set collections of documents such as Text REtrieval Conference (TREC) test sets. Test collections include a list of test topics, a corpus of documents to search, and expected responses per topic (Carterette *et al.*, 2012). Each TREC dataset is designed to evaluate a common task, such as spam detection or finding a webpage. This can result in poor performance results for novel search methods judged against TREC (Büttcher *et al.*, 2010).

2.4.2 Spatial search

Location is an important component in a wide range of data and is also important for correctly interpreting the information need of a user query. Data records can include spatial information as text (addresses or named features), point locations, more complex geometries, or some combination of these. Some of this detail may be hidden behind firewalls or otherwise inaccessible to regular search browsers.

Due to the democratisation of spatial data and analysis, driven partly by the expansion of spatial tools onto the Web (Reichenbacher, 2001), spatial expertise of users cannot be assumed (Skarlatidou *et al.*, 2011).

Initial research in IR, Web search, and semantic search focussed on text-based and multimedia data. Spatial searches have been treated as an add-on to these more traditional searches (Ballatore *et al.*, 2013; Janowicz *et al.*, 2011; Ruthven, 2011). Early research in particular focussed on data retrieval rather than IR; for example, finding datasets relevant to offline solution of a spatial problem.

Tasks in geographic information retrieval (GIR) are similar to those in IR, including *ad hoc* record retrieval and question answering, but take special consideration of geographic information. An important component is the identification of toponyms (place names) in

text. GIR is typically focussed on geographical texts, such as ranking relevant webpages, Wikipedia articles (Santos *et al.*, 2008), or text passages (Bilhaut *et al.*, 2003).

As a result, test collections for evaluation of GIR systems, such as GeoCLEF[23], are also focussed on spatial texts. Palacio *et al.* (2010) proposed a GIR evaluation framework that combined spatial, temporal, and topical search indices separately and in combination. They also proposed the addition of geographic resources, such as georeferenced entities from the corpus, within a test collection.

In this thesis, GIR is taken to include searches for spatial datasets and features as well as texts.

The first stage in GIR is the same as for all IR — where can potentially useful data be found? Selection of data sources and datasets is an important research area in Spatial Data Infrastructures (SDIs) and spatial data mining (Guo & Mennis, 2009; Li *et al.*, 2011). The next stages include extracting, ranking, and processing the data needed to respond to the original request. This research focusses on the extraction and ranking steps of this process.

Although location was described in early Semantic Web proposals (Berners-Lee *et al.*, 2001; Egenhofer, 2002), the importance of semantics and spatial ontologies in geospatial data searches is a relatively recent focus (Janowicz *et al.*, 2010; Li *et al.*, 2011; Zhang *et al.*, 2010a,b; Li *et al.*, 2014b; Sun *et al.*, 2014; Reed *et al.*, 2016).

Spatial searches have challenges beyond text-based searches, including how geometries can be accessed and manipulated. Methods to compute interactions between complex geometries have been researched for spatial databases and GIS, whilst GIR approaches are more recent and often focus on simpler geometries such as points and MBRs (Larson & Frontiera, 2004).

Complex spatial operations for a target geometry include listing n nearest neighbour features and finding all features within a set distance from the target. Results are typically still binary, however: a feature is either one of the n neighbours or it is not, and either within the buffered geometry or not.

Queries with spatial operations such as within, near, and overlap require specialised processing. Some spatial queries need to join two or more information datasets from

[23]GeoCLEF www.uni-hildesheim.de/geoclef/, accessed 6/6/2017.

one or more sources, such as named boundary regions and spatial features (Egenhofer, 2002). For example, a text query `"bus stops in Perth"` may need access to separate information sources for regions and bus stops. Given this query, keyword and NLP techniques are likely to ignore 'in' as a common stop word, when it should be retained as a spatial operator.

Regardless of these complexities, spatial searches still strive for the same output as any other search — successfully matching an information need to results from data sources. This means that a successful spatial search relies upon the (spatial) context of user, query, and data.

Data sources

Technologies used to store and interface with spatial data influence how spatial records can be accessed and tested. Some spatial Web services can be filtered by polygon, some by bounding box, and some cannot be spatially filtered at all. The WFS standard, for example, ensures that all WFS services respond to a GetCapabilities request with a list of available operations that may include optional filters such as BBOX, Within and Intersects (OGC, 2005, 2010). If any of these capabilities are not listed by a service, it cannot be assumed that it can apply such a spatial filter.

Calculations used for operations such as distance, buffering, and spatial indexing also differ depending upon software and settings used. There is a wide range of technologies for storing, maintaining and publishing spatial data online, including:

- Spatial database solutions such as PostGIS[24], and Oracle Spatial[25],
- Online spatial platforms such as Esri ArcGIS Online[26], and MapServer[27],
- Web mapping applications such as OSM,
- Web services such as those complying with Open Geospatial Consortium (OGC) standards[28] including WFS, Web Map Service (WMS), Web Coverage Service (WCS) and Web Processing Service (WPS), and

[24] http://www.postgis.net, accessed 11/1/2017.
[25] https://www.oracle.com/database/spatial, accessed 11/1/2017.
[26] https://www.arcgis.com/home, accessed 11/1/2017.
[27] http://www.mapserver.org, accessed 11/1/2017.
[28] http://www.opengeospatial.org/standards/, accessed 31/1/2017.

- Spatial data catalogues, portals, and online SDIs such as Data.gov.au[29], INSPIRE[30], and GeoNetwork[31].

Each technology has different capabilities and limitations, and carries its own technical and syntax requirements. The range of technologies is also changeable: it is subject to future expansion and modifications.

Even if a single technology were assumed — for searches within a single organisation, say — usage can change between different versions of the same software or standard. As an example, different WFS versions expect requests with different XML formatting, and same-version WFS services can implement different capabilities (Zhang *et al.*, 2010b). Encapsulating processing with data, so that queries do not have to include technical details, is one way to approach this problem of technological disparities (Gulland *et al.*, 2016).

Formats for transferring spatial data are used across different data source types. Standard data formats include shapefile — a set of related files for geometry, attribute and other related data, often combined in a zipfile — and text-based formats including Geographic Markup Language (GML) and Geographic JSON (GeoJSON).

For efficiency, spatial search operations rely upon the use of spatial indexes built with spatial data structures such as R-trees. Spatial indexing is an integral part of spatial databases such as Oracle Spatial and PostGIS (Güting, 1994).

Online spatial data catalogue solutions such as CKAN[32] allow data managers to manually add place names to the metadata description for a dataset, as well as add keyword tags to facilitate faceted search. The latter can be used to help users filter datasets by place name. Similarly, data records — including in Web services such as WFS — use data attributes to store values that can include place names and/or unique administrative codes.

Public resources including the global GeoNames, public OSM tool Nominatim[33] and the Australian Geocoded National Address File (G-NAF)[34] can be searched for matching

[29] http://data.gov.au, accessed 11/1/2017.
[30] http://inspire.ec.europa.eu/, accessed 11/1/2017.
[31] http://www.geonetwork.nl, accessed 11/1/2017.
[32] http://ckan.org, accessed 11/1/2017.
[33] http://wiki.openstreetmap.org/wiki/Nominatim, accessed 7/3/2017.
[34] Public Sector Mapping Agencies (PSMA) https://www.psma.com.au/products/g-naf, accessed 17/2/2017.

or alternative toponyms and geometries. However, commercial agreements are required for full access in some cases. Depending on a naming resource's purpose, it may have currency, accuracy or scope limitations, such as excluding place names that are historical, local, or in alternative languages. Place name repositories also do not include unique codes, such as used in censuses or by various government departments.

Toponyms can also be collected from data content. Jones *et al.* (2008), for example, used text analysis to assign probabilities of being location-related to query terms, based on query logs, corpus frequency, user location, and region population. Yi *et al.* (2009) also used query logs and text analysis, to extract city language models (Section 2.1.1).

Proximity and distance measures

A range of spatial relationships between geometries can be used to measure proximity or distance. Egenhofer & Franzosa (1991) formally defined nine topological relationships between regions, including overlapping, containment (i.e. with no part touching, overlapping or beyond the test region's boundary), and equality. Results from topological operators are binary, e.g. A either overlaps B, or it does not. Buffering is another example, such as the DWITHIN(A,B,x) operator, implemented by some Web Feature Services, that returns *True* if B is within x units of A.

Features, particularly if disjoint (not touching or intersecting), can be described more flexibly by other spatial operators such as intervening distance, including between region centroids (Jones *et al.*, 2008). Palacio *et al.* (2010, citing G. Andogah) described three general, non-binary measures of spatial similarity — containment, overlap, and distance.

- Containment includes the proportion of features within a reference ROI.
- Overlap is the proportion of spatial features, such as polygon extents, that overlap each other (e.g. for the yellow convex hull compared to the blue 'Perth Airport' polygon in Figure 2.8, the area of the red diagonally-barred region i divided by the combined areas).
- Distance measures include:
  - point-to-point straight line distance, e.g. between feature set centroids ($d1$ in Figure 2.8) or bounding box centroids ($d2$ in Figure 2.8),

– shortest distance between features in each dataset ($d3$ in Figure 2.9), and

– network distance following paths, such as roads, between nodes.

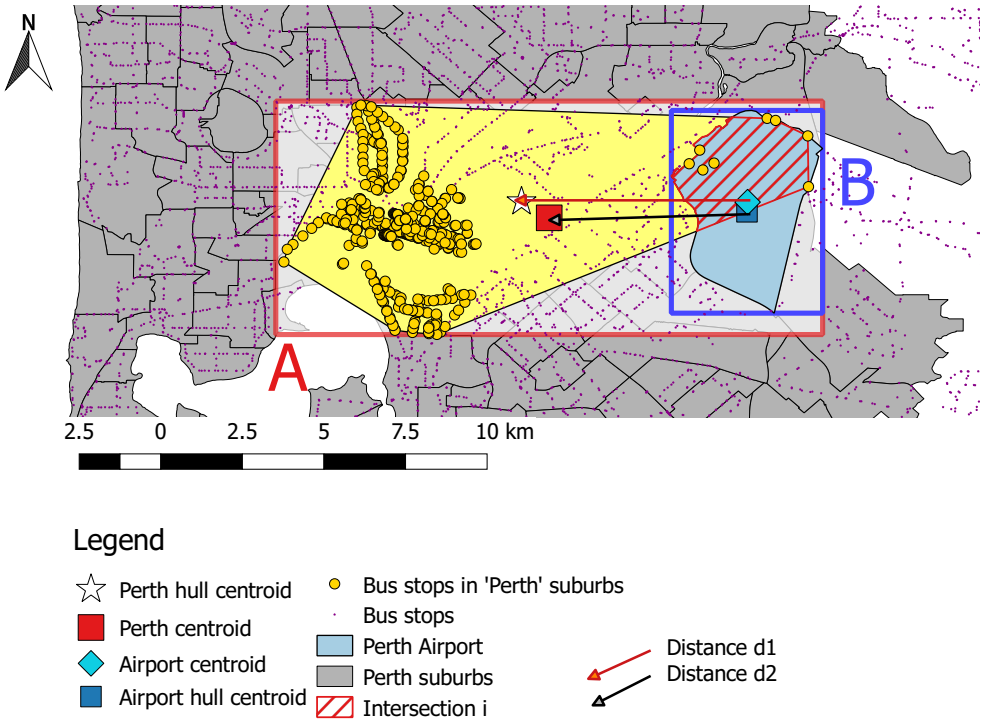


Figure 2.8: Example proximity measure between 'Perth' bus stops point set (in yellow convex hull, with MBR A) and 'Perth Airport' suburb (in blue, with MBR B), including centroid distances $d1$ and $d2$ and intersection area i.

Figure 2.9: Example proximity measure between 'Park' bus stops point set (in purple convex hull, detail) and 'Perth Airport' suburb (in blue), including the nearest feature distance $d3$.

Elements contributing to selection of a distance measure include geometry types (point, line, polygon, multipoint, etc), whether the Earth curvature needs to be accounted for (i.e. for larger distances), if conversions between spatial reference systems are needed, availability of other resources (e.g. road networks), the effect of region sizes on statistics such as population rates (Bachi, 1962), and what is appropriate for the feature type or theme involved.

There is a wide variety of calculations for geospatial distance between spatial features. Three examples, each for two-dimensional point geometries, are described here to illustrate how context affects the selection of a formula (Waller & Gotway, 2004).

- Euclidean distance (Equation 2.5) for points that are fairly close together, e.g. projected coordinates that use (or can be converted to) the same spatial reference system (SRS).
- Great-circle distance (GCD) (Equation 2.6, where r is the radius of the Earth, ϕ is latitude and λ is longitude) for geographic, i.e. unprojected, coordinates that are widely separated over the curving surface of the Earth.
- Network-based distances such as the City-Block distance that follows path segments in a road or other such network.

$$\text{Euclidean } d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.5}$$

$$\text{GCD } d = r \times \arccos\left(\sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos(|\lambda_1 - \lambda_2|)\right) \tag{2.6}$$

for distance d, Earth radius r, and latitude, longitude (ϕ, λ)

Network distances are valid for destinations on a road network such as hotels and hospitals, but not for values associated with regions like disease rates. Accurate distances are calculated using shortest-path algorithms and require access to appropriate network data. The Minkowski metric is a geometrical measure used to approximate network distances that can be quickly calculated without constant reference to network data or path-routing algorithms (Janowicz *et al.*, 2011; Shahid *et al.*, 2009).

Ranking of aspatial features in a dataset by their distance from a target is used for spatial operations including finding nearest neighbours (Hjaltason & Samet, 1995). However, distance values alone are of limited use for comparing geometries from multi-

ple datasets, given the variety of possible calculation methods, SRS used, and units of measurement. A spatial *ranking* method is more appropriate for GIR, with proximity values in the same range (assumed 0.0...1.0 unless specified otherwise). Spatial ranking of geometries in relation to a feature of interest (FOI) can be calculated based on distance or region overlap as well as other attributes such as size and shape (Larson & Frontiera, 2004).

Overlap measures include simple overlap (true if there is any overlap, otherwise false) that is easily calculated for MBRs, but cannot rank results. Overlap extent measures range from zero (not similar) to one (equivalent) and are calculated from region areas. Larson & Frontiera cited three examples, illustrated in Equation 2.7 (Hill, 1990), Equation 2.8 (Walker *et al.*, 1992), and Equation 2.9 (Beard & Sharma, 1997) for regions Q (query, i.e. a ROI) and R with overlapping area O (Figure 2.10).

$$\text{range} = 2\frac{O}{Q+R} \tag{2.7}$$

$$\text{range} = \min\left(\frac{O}{Q}, \frac{O}{R}\right) \tag{2.8}$$

$$\text{range} = \begin{cases} \frac{R}{Q}, & \text{if } Q \text{ contains } R \\ \frac{\frac{O}{Q}}{(1-\frac{O}{R})+1}, & \text{if } Q \text{ and } R \text{ overlap} \\ \frac{Q}{R}, & \text{if } R \text{ contains } Q \end{cases} \tag{2.9}$$

for region areas Q, R and their overlap area O

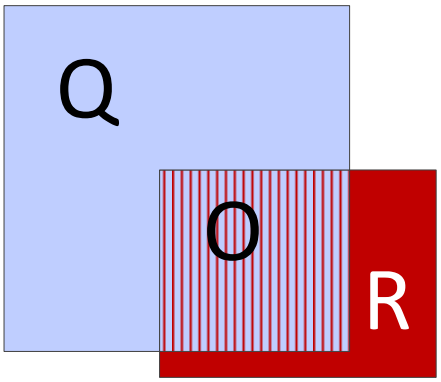


Figure 2.10: Overlap between query region Q and test region R.
Area of Overlap $O = Q \cap R$.
Area of Union $U = Q \cup R = Q + R - O$.

Larson & Frontiera also described a probabilistic measure, based on the logistic regression model. This measure used area ratios between the query and test region and also took into account the ratio of land versus water covered by each region.

These scores have a zero weight for adjacent or disjoint features. Alternatively, features can be ranked by distance from a target location, a process that can be made more efficient with the use of spatial indexes (Hjaltason & Samet, 1999).

Application- and environment-specific nearness methods have also been defined, for example the Public Transport and Walking Access Index developed by Mavoa *et al.* (2012) for Auckland, New Zealand. Specific details are outside the scope of this thesis, other than to note that proximity calculation methods can be closely related to the data they serve, in order to inform searches upon that data.

Contextual information, including technical capabilities at the data access point, available data formats, and metadata such as SRS, also affects proximity calculations. In the latter case, spatial features described with different projections need to be mapped to a common SRS before they can be compared (Vaccari *et al.*, 2009).

An issue with all types of contextualised searches, including spatial ones, is how to relate specialised search tasks to what is known about a specific data source. One approach to manage this problem is encapsulating data access and search behaviour together in self-contained agents that can communicate with each other.

2.5 Software agents

Software agents are self-contained programs that react to what they perceive and can communicate with other agents in order to solve tasks, such as comparing a search query against the content and context of a data source.

Research into agents, beginning in the field of artificial intelligence, has been active since the twentieth century and the subsequent wide range of fields and applications mean that a precise definition is problematic — the scope and purpose of an agents depends upon its problem space (Wooldridge & Jennings, 1995; Nwana, 1996).

Within computer science, software agents are sometimes also referred to as intelligent agents (Sengupta & Sieber, 2007), although this is also used in a broader sense in artifi-

cial intelligence for robotic, human, and other agents (Russell & Norvig, 2010). Nwana (1996) defined seven software agent types: collaborative, interface, mobile, information, reactive, hybrid, and smart, where agents can incorporate aspects of two or more types. Software agents may also be described by their primary purpose, e.g. search, semantic, or broker agent (Li & Horrocks, 2004).

Agents have been used to search for a variety of data sources including spatial (Gulland *et al.*, 2016; Sengupta & Sieber, 2007) and semantic (Çelik & Elçi, 2006; Cesarano *et al.*, 2003; Kerschberg *et al.*, 2001). Jansen *et al.* (2006) categorised search agents by the IR stage they conformed to: discovering online resources, query adaptation, or filtering and managing results. Potential advantages of an agent-based approach to search include flexibility, provenance reporting, and data privacy management (Moncrieff *et al.*, 2016).

Features of agents include: 1) autonomous behaviour[35], 2) perception of the current environment or situation, relevant to their purpose, and 3) communication with other agents via messages (Wooldridge & Jennings, 1995; Nwana, 1996). Agents interact through messages, which are specified in standard formats to allow for independence from implementation detail (Li & Horrocks, 2004).

Cooperating agents negotiate with other agents as required, and in some cases incorporate learning over time (McNeill, 2013). Discovery of relevant and useful agents is an important component of complex Multi-Agent Systems (MAS) (Viroli *et al.*, 2006; Jansen *et al.*, 2006). The purpose is often to discover Web services that online agents can use for communication.

Fully autonomous agents seek other agent(s), without needing outside direction, by querying agent capabilities. MAS can also use partially autonomous agents, with a middleware layer to provide some or all of the interaction capabilities (Weyns, 2010). Agents can be designed specifically for a coordination task, for example 'middle agents' (Viroli *et al.*, 2006) or 'broker agents' (McIlraith *et al.*, 2001).

Agents are a useful complement to the Semantic Web, with its goal of facilitating automated communication between computers without prior knowledge of format, content, or capabilities (Shadbolt *et al.*, 2006). Agents can fill an important niche within an

[35]The level of independence varies by agent purpose, for example Semantic Web agents generally receive more direction than fully autonomous agents (Caliusco & Stegmayer, 2010).

automated chain of online processes, defined in ontologies (Hendler, 2001).

Semantic agents can contribute to online search tasks. Çelik & Elçi (2005, 2006), for example, developed a Semantic Search Agent (SSA) to discover Semantic Web services appropriate to query terms, using an ontology of related terms. Cesarano *et al.* (2003) also used intelligent search agents and the Semantic Web for IR, specifically for on-the-fly data mining.

Geospatial agents have been designed and used in problems including data retrieval, contextual visualisation, location based services, and models of pedestrian or wildlife movement (Sengupta & Sieber, 2007). Simulation problems incorporating location have made use of agent-based modelling and simulation (ABMS) (Macal & North, 2009). Examples of resources supporting development and use of such spatial agents include NetLogo[36] and Repast[37].

In the geospatial domain, online software agents have also been used variously to match terminology context between data requesters and providers (Huang & Webster, 2004), geocode addresses (Hutchinson & Veenendaal, 2013), incorporate data context into multi-source data searches (Gulland *et al.*, 2016), provide specific geospatial processes that can be chained together into more complex solutions (Yue *et al.*, 2007; Zhao *et al.*, 2012), manage contextualised geovisualisations for user-lead data exploration (Moncrieff & West, 2013; Moncrieff *et al.*, 2016), and as part of a spatial decision support system (Sengupta & Bennett, 2003).

2.5.1 Web services

A Web service is a public interface that can initiate and respond to online messages. Services are provided and requested by *entities*, which may themselves be services (McNeill, 2013). Whilst the interface is public, implementation detail is hidden, for example within an agent.

The power of Web services is realised through interaction of multiple services, extending the older concept of MAS (Hendler, 2001; Viroli *et al.*, 2006; Sengupta & Sieber, 2007). Web services are primarily for machine-to-machine communication, rather than direct access by users. Mobile cloud services — use of Web services from mobile devices — is

[36] https://ccl.northwestern.edu/netlogo/, accessed 1/5/2017.
[37] https://repast.github.io, accessed 1/5/2017.

a practical example. Lee (2013), for example, used Web services for computer-intensive image processing in a mobile device application that analysed local imagery.

There are a number of approaches for orchestrating multiple services, depending upon the intended purpose. *Chaining* involves a sequential set of steps, where results from one service task can be used as inputs to the next in sequence. *Composition*, by comparison, is more complex: each service can call upon other services hierarchically and, with distributed services, some steps can be carried out simultaneously (Rautenbach *et al.*, 2013). A practical example is orchestration of geoprocessing services to solve complex spatial tasks (Zhao *et al.*, 2012; Supavetch & Chunithipaisan, 2011).

A Web application is one example of a service coordinator or mediator (Lu & Sterling, 2000), combining results from multiple services to complete a task. Coordinators can make use of a Web service management framework such as Service-Oriented Architecture (SOA), which involves three types of entity: service requestors, service providers, and discovery agencies (Manoj & Ghosh, 2006).

An alternative to SOA is a Resource-Oriented Architecture (ROA), which focusses on managing access to distributed sources of data in multiple formats and domains rather than online services providing specified functionality. In ROA, data is published online directly by providers and accessed via RESTful requests, for example using SPARQL endpoints to query RDF ontologies. Linked data can be exposed in this way, whilst still hiding management detail within agents (Janowicz *et al.*, 2013).

Any system requiring Web service access needs to be able to communicate with it and discover its capabilities and requirements, regardless of implementation detail. This is achieved through the use of one or more standards or protocols.

Communication protocols for sending and receiving service messages include Simple Object Access Protocol (SOAP)[38] and Representational State Transfer (REST). SOAP messages are composed in XML and transferred via HyperText Transfer Protocol (HTTP). REST is another Web interface protocol that can communicate with services via HTTP requests and responses (Fielding, 2000).

Several service description standards have been defined, including Web Service Description Language (WSDL)[39] for XML-based services, e.g. communicating with SOAP

[38] https://www.w3.org/TR/ws-arch/#SOAP, accessed 7/2/2017.
[39] https://www.w3.org/TR/wsdl, accessed 7/2/2017.

(Rautenbach *et al.*, 2013; W3C, 2007). Description and communication protocols have also been defined for services in specific domains, including business transactions via Business Process Execution Language (BPEL)[40], the Semantic Web, and geospatial data and processing.

Intelligent agents can also discover Web services described in ontologies on the Semantic Web. Standards specifically designed to describe and access semantic services include Web Ontology Language for Services (OWL-S) (Martin *et al.*, 2008; McIlraith *et al.*, 2001), and Web Service Modeling Language (WSML) (Chun & Warner, 2008; WSMO-WG, 2008). Caliusco & Stegmayer (2010) investigated discovery of ontology services using machine learning approaches such as artificial neural networks (ANNs).

Different service-describing formats can work together, for instance WSDL and WSML can be used to describe different aspects of the same service — message interface for the former and behaviour descriptions for the latter[41].

OGC[42] RESTful standards for geospatial services include WFS[43] for vector feature data and querying, WCS for raster data, WMS[44] for map imagery, and WPS for geo-processing tasks, amongst others (OGC, 2005, 2006, 2012, 2007).

Rautenbach *et al.* (2013) found that BPEL and WSDL were insufficient for orchestrating OGC spatial services, due to incompatibilities of formats and the need to manually write WSDL files, and hence used a workflow script to produce a Web service capable of using other services.

Geo-semantics research investigates improving spatial IR by linking to both semantic and spatial Web services (Janowicz *et al.*, 2010). de Andrade & Baptista (2011) identified metadata issues in SDI that could have an adverse impact on information discovery, including the OGC services they used. The issues included imprecision, missing data, and generalisation caused by resource- or dataset-level descriptions being provided without feature-level information. Their proposed solution combines spatial services with semantic information derived from them.

Farazi *et al.* (2011) approached the geo-semantic problem by developing local knowledge

[40] https://www.oasis-open.org/committees/wsbpel, accessed 7/2/2017.
[41] https://www.w3.org/Submission/WSMO-related, accessed 3/4/2017.
[42] http://www.opengeospatial.org/standards/, accessed 31/1/2017.
[43] http://www.opengeospatial.org/standards/wfs, accessed 31/1/2017.
[44] http://www.opengeospatial.org/standards/wms, accessed 9/2/2017.

for use in service queries; specifically, a faceted ontology that split domain knowledge such as named regions and building types into a number of hierarchical subtrees. They used their faceted ontology to adapt queries over GeoNetwork services.

Data discovery

The utility of both Web services and the Semantic Web depends partly upon the ability to automatically discover existing services that can carry out required tasks (McIlraith *et al.*, 2001), such as returning datasets relevant to a query. Data discovery commonly investigates services via known interaction formats such as WSDL.

In Bone *et al.* (2014), a list of appropriate search keywords for geospatial services and webpages was maintained from query logs to assist in matching user queries to relevant spatial data services. They used a non-Semantic Web crawler to find spatial datasets, although using ontologies to expand queries was a stated future direction.

Bogdanović *et al.* (2015) noted in their research into geospatial data discovery that mapping between multiple service interface standards caused limitations to reasoning across all sources: combining service description formats for geospatial and semantic data aspects reduced the consistency of outcomes.

In geospatial research, data discovery focusses mainly on finding datasets, usually based upon metadata descriptions, rather than on retrieval of features within those datasets. Zhang *et al.* (2010b) is an exception that retrieves records via specialist OGC services. Their spatial feature discovery system combined spatial and semantic services. A complication they faced was incompatibility between the SOAP-based semantic and REST-based spatial service descriptions.

Yue *et al.* (2007) used a SOA-based online system to construct geospatial Web services, with OGC and pre-existing semantic services, describing their "geospatial semantics schema" in OWL-S. Results from systems combining semantic and spatial services are affected by measures used to determine what concepts are "semantically similar" (Janowicz *et al.*, 2011).

Gao *et al.* (2009) developed a SOA-based online system for processing spatial health data, using WFS and WPS services. Best *et al.* (2007) used OGC Web services in an online system for modelling marine mammal habitats, but noted that more work was

needed to map between OGC and SOAP.

Tasks throughout this thesis, including for contextual search agents, require judgements of relationship strengths — between contexts, terms, data records, spatial features, and datasets. The next section discusses measurement and evaluation methods for this purpose.

2.6 Making and evaluating matches

Machine learning methods are applied across many fields, including NLP (Section 2.2.2). The aim of many techniques investigated in this section is to produce comparative similarity or relevance measures between items. This has applications beyond information retrieval (IR), for example to group similar items or match ontologies. Selection of relevant machine learning measures needs to be based upon the context where they are used, including application purpose and knowledge domain (Janowicz *et al.*, 2011).

The following sections include background on machine learning approaches for determining and using weight values (Section 2.6.1) and measures to judge their efficacy (Section 2.6.2). Machine learning approaches described include classification, regression, and clustering.

2.6.1 Machine learning

Machine learning algorithms learn to detect and predict patterns in input data. Approaches include supervised and unsupervised methods, where the former uses input data labelled with expected outcomes for use in training. Unsupervised algorithms learn patterns from unlabelled data without reference to expected outcomes. Once trained, the algorithm can be applied to similar items from outside the training set.

Training data includes a set of attributes (also called measures or features) for a set of observations. For example, attributes in BOW documents are frequency counts for each corpus token.

Individual algorithms are designed to solve specific problem types, including regression models that predict the value of a dependent variable based on other attributes, classifiers that learn how to allocate items to different logical categories, clustering algorithms

that use patterns in the input data to find groups of similar items, and dimensionality reduction to simplify inputs by decreasing the number of attributes used to predict results.

Machine learning methods can be combined together, for example using dimensionality reduction to improve the efficiency of clustering algorithms (Jain, 2010), or topic modelling to provide inputs for classification (Blei *et al.*, 2003).

Algorithms used in this thesis include:

- dimensionality reduction — Principal Component Analysis (PCA),
- regression — linear regression, logistic regression, regression decision trees,
- classifiers — decision trees, Support Vector Machine (SVM), Multilayer Perceptron (MLP),
- clustering — Density Based Spatial Clustering of Applications with Noise (DBSCAN), and
- topic modelling — Latent Dirichlet Allocation (LDA) (Section 2.2.2).

Principal Component Analysis (PCA) uses linear algebra to transform a matrix of n observations and p attributes into a matrix with k components for each observation. A component is an eigenvector built from a transformation of the original, raw attribute values, associated with an eigenvalue that measures its variance. Components are orthogonal to each other, as they are calculated to maximise variance from each other. For dimension reduction, lower-variance components are ignored, giving a new $n \times k$ dataset where $k \leq p$ (Johnson & Wichern, 2002; Shlens, 2014).

As the first component calculated with PCA always accounts for the largest possible proportion of variance within the raw data (for that combination of measures), this calculated value can sometimes be used as a rough estimate of the relationship strength between items.

Regression and Classification

Regression analysis is used to predict relationships between observations or items, based upon their attributes. The estimate produced for one item can be a single numerical or

categorical value, where the latter is an example of classification.

Classification algorithms assign observations to categories, based on their attribute values. Binary classifiers select between two classes (e.g. *related* and *unrelated* term pairs) and multi-class classifiers assign items to one of three or more categories (e.g. *animal*, *vegetable*, *mineral*).

Linear regression estimates a weight W for each observation, based on a linear combination of observation attributes and a common set of weights. The resulting equation can be used to predict weights for new items in the same feature space, as shown in Equation 2.10 for n attributes $[a_1, ...a_n]$. Using supervised learning, the attribute weights $[w_1, ...w_n]$ are selected such that the sum of squared differences between a known label X and predicted weight W is minimised across the training dataset (Witten *et al.*, 2016).

$$\text{Predicted weight } W = w_0 + \sum_{i=1}^{n} (w_i a_i) \tag{2.10}$$

$$\text{for attributes } a_{1..n} \text{ and attribute weights } w_{0..n}$$

In some circumstances, such as when there is a non-linear dependency between attributes, logistic regression is a more suitable model. It transforms the target value by building a linear model using a logit transformation, which is trained by maximising the log likelihood instead of the linear sum of squared differences. Logistic regression can be used to calculate the probability of membership for a class.

Regression analysis can be used for binary classification, for example by assigning training values of one for *related* and zero for *unrelated* paired terms in a test set. In this case, the calculated weight estimates the likelihood of relatedness between terms in a pair. Binary logistic regression can be used to estimate the probability that an observation belongs to one of the two outcomes.

Decision trees describe a series of attribute-based rules that predict a class. Where results are from a continuous scale, a *regression* tree can be used to predict a representative value. Rules are specified at each internal tree node for values of one or more categorical or numerical attributes. As a result, each tree node describes a subset of items from the training data.

A three-class decision tree is shown in Figure 2.11 for a pair of regions (A,B), classifying by potential utility of spatial features in B relative to query region A.

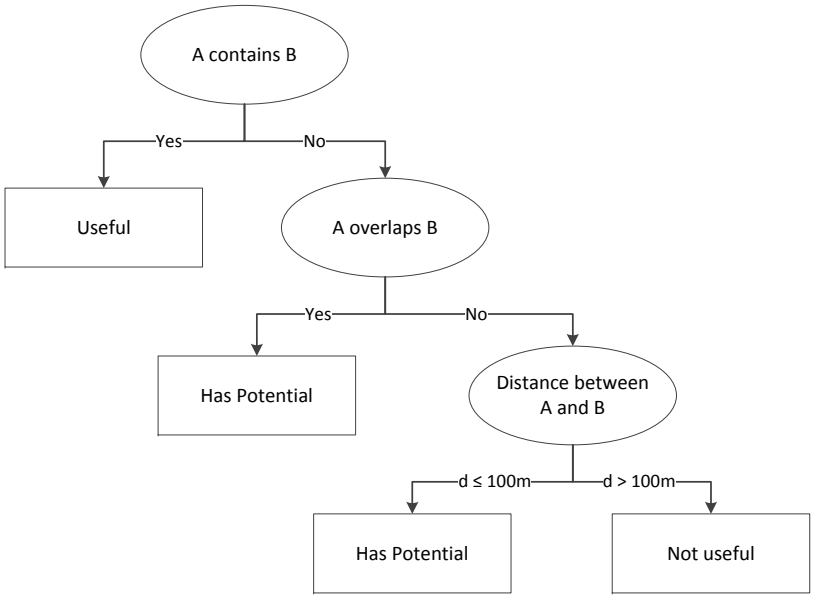


Figure 2.11: A decision tree example for paired regions A and B, classifying by potential usefulness of B records to query region A. Classes are specified in leaf nodes (rectangles).

Supervised machine learning methods for constructing decision trees take a divide-and-conquer approach to dividing the training data into buckets. Leaf nodes contain items that cannot be further subdivided, either because all items in that bucket are equivalent (i.e. in the same class) or because a limit (such as maximum tree depth) has been reached. One way to build regression trees is to minimise the error between known X labels and calculated W values within the group of items in each node.

Overfitting is a potential problem for decision and regression trees, where the model is so closely matched to the training data that it does not generalise well to other data. Methods such as pruning or other adjustments can be applied to reduce this issue (Wu *et al.*, 2008).

A Support Vector Machine (SVM) is a supervised binary classifier that determines an optimal linear hyperplane separating category items by the largest possible margin (Cortes & Vapnik, 1995). For data that is not linearly separable, alternative classification functions can be substituted (Wu *et al.*, 2008).

Although SVM is best suited to two classes, it can be adapted to multi-classification using one-against-all testing. For example, for a four-class model with classes A–D, four

SVMs would be required: A versus B+C+D, then B versus A+C+D and so on. Results can be combined by cross-validating results against each other.

Cilibrasi & Vitanyi (2007) used SVM to test if NGD values could be used to predict the WordNet Connectivity (WN) category of a term, finding agreement rates greater than 75%. However, they were not investigating the discovery of new categories, such as may be expected for locally specialised texts.

Multilayer Perceptrons (MLPs) are supervised models that can be used for classification; they are a type of artificial neural network (ANN) that use a number of input values connected to output nodes (such as a list of possible classes) via one or more hidden weighted layers. The training process adapts these internal weights to the data to improve the accuracy of the final results.

MLP uses backpropagation to classify items based on their data attributes. That is, it calculates results from existing weights, then edits the weights to reduce the difference between expected and actual outputs. Hidden inner layers contain neurons that apply an activation function to transform the weights. Examples of activation functions that can be used in a MLP model include sigmoids and softmax (Mikolov *et al.*, 2013).

Clustering

Clustering algorithms, which can be supervised or semi-supervised, group items together based on similarities in their attributes. A wide range of algorithms are available for clustering data, including k-means clustering (Jain, 2010). This family of algorithms aims to minimise variance within each cluster, and is best-suited to clusters that are similar in size and separated from each other. The expected number of clusters n is provided to the algorithm as an input.

The Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is better able to adapt to uneven cluster sizes and shapes. It does not require prior knowledge of n, but does need to know the minimum number of points per cluster (m) and the maximum distance between points within the same cluster (epsilon ϵ), and clustering results are sensitive to these input parameters (Ester *et al.*, 1996).

Clustered items may be physically close, such as houses separated by small distances. Other measures such as semantic distance can also be used — for example, to group

books with similar themes or terms with related meanings. The distance measures can be used as edge weights on a weighted graph of terms, where items joined by small edge values are grouped into strongly related clusters.

Term categories can serve a number of purposes: 1) validate relatedness measures for use in building terminological ontologies, 2) discover outliers that are not related to other terms, 3) augment link strengths for terms within the same cluster, and 4) contribute to a hierarchical taxonomy of terms that can further group and connect terms. A taxonomy can accommodate search filters, e.g. to limit a search to terms within a disease-related term cluster as in the Figure 2.12 example.

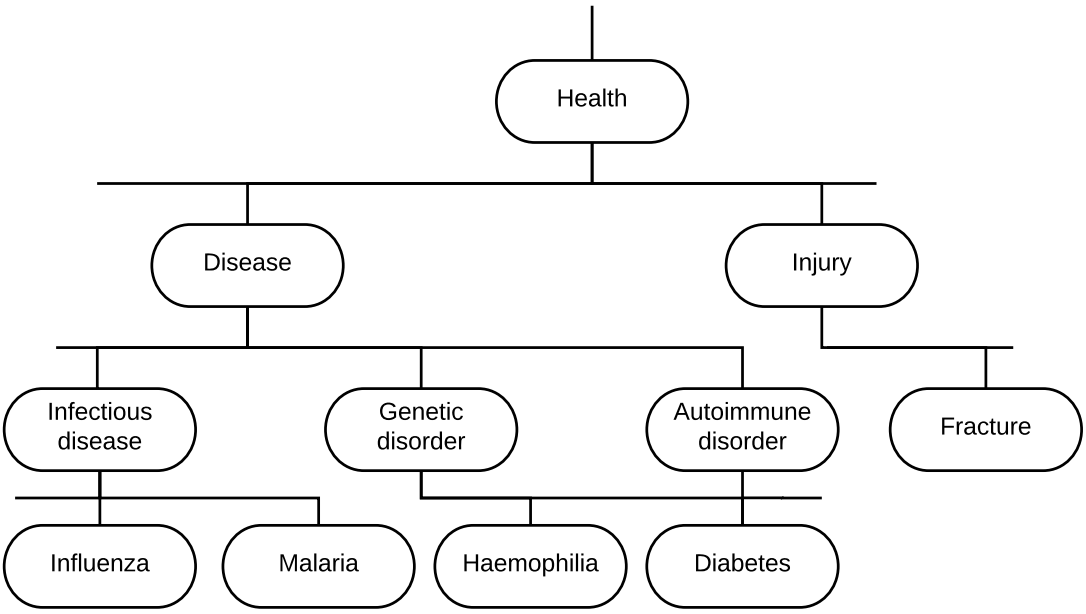


Figure 2.12: Example of hierarchical grouping of terms.

Chen *et al.* (2012) described a multidimensional clustering approach for overlapping groups, using multiple properties for each item. Chen *et al.* gave an example of data about persons that naturally groups into both gender- and nation-based clusters, from the same set of attributes.

2.6.2 Evaluating results

An important component of machine learning is evaluating their suitability for their intended purpose. Labelled training data for supervised learning is particularly useful for this purpose, for both nominal categories (classes) and numeric weights.

Training and testing can be combined to reduce result bias and over-fitting by dividing labelled training data into a number of subsets, called *folds*. One fold is kept aside for testing, with the rest used for training. This is repeated for each fold, and test results from each training session are averaged.

The following sections describe metrics used to evaluate the performance of machine learning and IR results against expected outcomes. These include statistical classification methods that compare assigned categories to test labels, IR ranking results, error rates for numeric results, and clustering measures that compare results within and between clusters.

Statistical classification measures

Statistical measures quantify the effectiveness of classification into either satisfying (positive) or not satisfying (negative) a condition, such as membership of a particular class or category. The following equations use a binary classification example of *related* (size R) and *unrelated* (size U) items.

Classification statistics are calculated based on the numbers of true positives (TP), false positives (FP), and true and false negatives (TN and FN). True positives and negatives are correctly classified; false positives and negatives have been assigned to the incorrect category. Statistical measures for binary classification, or per class for multiclass problems, include:

- True positive rate (TPR) (also called recall or sensitivity): the proportion of target items that were correctly classified (Equation 2.11),
- False positive rate (FPR): the proportion of non-target items that were incorrectly classified as the target class (Equation 2.12),
- Accuracy A: the proportion of items that were correctly classified (Equation 2.13),
- Precision P: the proportion of items identified as the target class that were correct (Equation 2.14), and
- F-measure (F_1): a balance between recall and precision (Equation 2.15).

A perfect calculated result would have TPR=P=A=F_1=1 and FPR=0.

$$\text{True positive rate } \text{TPR} = \text{Recall} = \frac{\text{TP}}{\text{R}} \tag{2.11}$$

$$\text{False positive rate } \text{FPR} = \frac{\text{FP}}{\text{U}} \tag{2.12}$$

$$\text{Accuracy } A = \frac{(\text{TP}+\text{TN})}{(\text{R}+\text{U})} \tag{2.13}$$

$$\text{Precision } P = \begin{cases} 1, & \text{if } (\text{TP}+\text{FP}) = 0 \\ \dfrac{\text{TP}}{(\text{TP}+\text{FP})}, & \text{otherwise} \end{cases} \tag{2.14}$$

$$\text{F-measure } F_1 = 2 \times \frac{(P \times \text{TPR})}{(P + \text{TPR})} \tag{2.15}$$

for true positive TP, false positive FP, true negative TN,

related (i.e. all positives) R, and unrelated U counts

The correlation coefficient ranges from -1.0...1.0 and shows the strength of a linear relationship between predicted and actual classifications. A value of zero shows no linear relationship — where there may be a non-linear relationship or none at all — and values at the other extreme show a strong negative (close to -1) or positive (close to +1) linear correlation.

A confusion matrix (Equation 2.16) compares true and false positives and negatives in a grid, with rows listing counts for labelled training data and columns listing the counts for classes as identified by the model. For binary classification, such as the related/unrelated terms problem, this results in a two by two grid:

$$\begin{bmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{bmatrix} \tag{2.16}$$

For binary classification, the Matthews Correlation Coefficient (MCC) (Equation 2.17) is a useful numerical representation of a confusion matrix. The MCC returns +1 for perfect classification and -1 for its inverse (all positive examples classified as negative and vice versa). A value close to zero represents random classification.

$$\text{MCC} = \frac{(\text{TP} \times \text{TN} - \text{FP} \times \text{FN})}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}} \tag{2.17}$$

for true positives TP, false positives FP, and true negatives TN

Calculated and ground truth results can be compared visually in a receiver operating characteristic (ROC) plot. The ROC curve summarises a measure's performance by plotting TPR against FPR. A diagonal line from (0,0) to (1,1) represents a neutral line where the TPR and FPR cancel each other out and thus is roughly equivalent to chance (random classification). In general, a line that curves closest to the ideal point at top left where TPR=1 and FPR=0, and hence has a larger area under the ROC curve (AUC) value, describes a better result.

The Kappa statistic is a measure of accuracy that accounts for chance classification. It ranges from $0.0 \ldots 1.0$, where all values larger than zero are greater than chance, and values close to one show near perfect prediction. It is calculated with reference to error rates in the classification, as shown in Equations 2.18—2.20.

$$\text{Total Accuracy tA} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \tag{2.18}$$

$$\text{Random Accuracy rA} = \frac{(\text{U} \times (\text{TN} + \text{FN})) + (\text{R} \times (\text{FP} + \text{TP}))}{(\text{R} + \text{U})^2} \tag{2.19}$$

$$\text{Kappa} = \frac{(\text{tA} - \text{rA})}{(1 - \text{rA})} \tag{2.20}$$

for true positive TP, false positive FP, true negative TN,

related (i.e. all positives) R, and unrelated U counts

Evaluating information retrieval results

An IR system is evaluated relative to an information need, in terms of the relevance and ranks of returned results and the completeness (i.e. recall) of results from the total available set of relevant items.

Precision and recall can be adapted to evaluate IR requests by comparing returned, ranked results (Res) to the set of all relevant results, whether returned or not (Rel). These are calculated in terms of the top k results returned (i.e. the k most highly ranked results), as shown in Equations 2.21 and 2.22 (Büttcher *et al.*, 2010).

$$\text{Precision@k } P@k = \frac{|\,\text{Rel} \cap \text{Res}_{1..k}\,|}{|\,\text{Res}_{1..k}\,|} \tag{2.21}$$

$$\text{Recall@k } R@k = \frac{|\,\text{Rel} \cap \text{Res}_{1..k}\,|}{|\,\text{Rel}\,|} \tag{2.22}$$

for relevant records Rel, returned results Res, and result set size k

Standard IR performance measures include Average Precision (AP) at recall levels $k = [0.0, 0.1, 0.2, ..., 1.0]$ and Mean Average Precision (MAP), i.e. the average of AP values across all queries (Larson & Frontiera, 2004). AP can be calculated from a binary (relevant/irrelevant) ground truth set of results as illustrated in Equation 2.23, adapted from Büttcher *et al.* (2010, Equation 2.23). The relevant(i) function is one if the ith-ranked result is relevant, or zero if not.

$$\text{Average precision AP} = \frac{1}{|\,\text{Rel}\,|} \cdot \sum_{i=1}^{k} \text{relevant}(i) \times P@i \tag{2.23}$$

for relevant records Rel and result set size k

Care needs to be taken when interpreting the rank of an IR result. The item calculated as the 'best' result has a rank of one; in other words, lower ranks have greater magnitude than higher ranks. Where ranking is based upon likely relevance, a higher probability would result in a higher (lower-valued) rank.

Reciprocal rank (RR) is a ratio measure that is larger for target results with higher ranks (Equation 2.24, adapted from Büttcher *et al.* (2010). That is, where a smaller value of k is required to return only relevant items.

$$\text{Reciprocal rank RR} = \frac{1}{\min\{k \mid \text{Res}[k] \in \text{Rel}\}} \tag{2.24}$$

for relevant records Rel, returned results Res, and result set size k

Error rates

Error rates are useful for evaluating numeric predictions, as they describe the difference between expected (i.e. ground truth) and calculated weights, represented here as w and w', respectively (Witten *et al.*, 2016). Formulae for calculating errors are shown in Equations 2.25 — 2.27, where n is the number of term pairs measured.

Mean Squared Error (MSE) is the average of squared errors, where the error e is the difference between calculated and true weights: $e = (w' - w)$. Root Mean Squared Error (RMSE) documents the standard deviation of the differences. Mean Absolute Error (MAE), such as reported by the Weka toolkit, is a linear measure of errors. Each of these three measures has a range that depends upon the measured values ($0.0 \ldots 1.0$ in this research).

$$\text{Mean Squared Error (MSE)} = \frac{1}{n}\sum_{i=1}^{n}\left(w_i' - w_i\right)^2 \tag{2.25}$$

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\text{MSE}} \tag{2.26}$$

$$\text{Mean Absolute Error (MAE)} = \frac{1}{n}\sum_{i=1}^{n}\left|w_i' - w_i\right| \tag{2.27}$$

for true and calculated weights w, w' and set size n

Relative Absolute Error (RAE) and Root Relative Squared Error (RRSE) are normalised over the range of values to give percentage scores for MAE and RMSE respectively. They are calculated by dividing the raw measures by the prior probabilities of data observations.

Clustering measures

Clustering algorithms can be evaluated with statistical measures including classification measures, error rates, homogeneity (including completeness and V-measure scores), Adjusted Rand Index, Adjusted Mutual Information (AMI), and Silhouette coefficient scores.

Each of the clustering metrics is a single value calculated across all items. Together, they measure how accurately items were classified into clusters by comparing results against the ground truth cluster labels for each item.

Homogeneity, completeness, and V-measure. Each of these measures is in the [0.0...1.0] range, with 1.0 being an ideal result. A homogenous cluster only contains items correctly classified as that cluster; a complete one contains all possible items of that class. The V-measure score is a harmonic mean of homogeneity and completeness.

Rand index and AMI. The adjusted Rand index [-1.0...1.0] measures similarity between ground truth and predicted class labels. A result of 1.0 is perfect, whilst a negative or zero index indicates a poor prediction (i.e. independent labelling).

Mutual information scores measure the agreement between assigned and ground truth classes, where AMI is adjusted to disregard agreement by chance. A result of 1.0 de-

scribes perfect labelling, and a value near 0.0 implies independence between the true and calculated labels.

Silhouette coefficient. The silhouette coefficient [-1.0...1.0] is an estimate of the clarity of clusters, calculated without reference to ground truth labels. This makes it useful for unsupervised methods, or if clear clusters exist that do not match the ground truth. Ideally, this will be close to 1.0, with negative values implying an incorrect allocation, and a value close to zero indicating a large overlap between clusters.

2.7 Summary

This chapter has discussed previous research related to the aims of this thesis: that is, automated creation of weighted ontologies from local content and public crowd-sourced data and using online search agents to apply local context in spatial semantic search.

The concept of context was explored, particularly as it relates to search. Location and vocabulary were the aspects of context relevant to this thesis, with both impacting on how an IR request is interpreted as an information need. Types of spatial measures were also described in terms of their potential use in contextualised search agents. Further background on natural language and natural language processing was reviewed, including research into the use of public crowd-sourced resources Google and Wikipedia in determining relationships between terms.

Next, search was reviewed from two main perspectives: use of ontologies and the Semantic Web to assist in matching concepts, and Web search and information retrieval. Types of weighted ontologies were reviewed, given their potential for supporting flexible term relationships. Spatial perspectives on semantic and Web search were emphasised.

Web services were a common theme across research into online spatial search. Given the agent approach within this thesis, the use and design of agents and Web services, particularly in search, was reviewed. Finally, machine learning methods used in this thesis were outlined, including methods to find and evaluate term matches and test IR ranking.

The next chapter will describe the approach taken to develop a weighted ontology of

terms, related to the context of a data source. This will make use of machine learning techniques on text from public, crowd-sourced and manually-collated resources including Wikipedia and Google.

3 Matching terms

This chapter investigates the automated creation of terminologies relevant to a specific data source. Such a resource can be used to automatically expand search key terms and match a query to more of its records.

The layout of this chapter is illustrated in Figure 3.1. Section 3.2 describes term-relatedness measures used and the evaluation of individual and combination measures against ground truth weights (labels 1–3). Statistical and machine learning methods to extract term pairs for ontologies and values for weighted ontologies are discussed in Section 3.3 (labels 4–5). Automated grouping of terms into categories and clusters based on term-relatedness weights is explored in Section 3.4 (labels 6–7). Section 3.5 discusses results from the chapter.

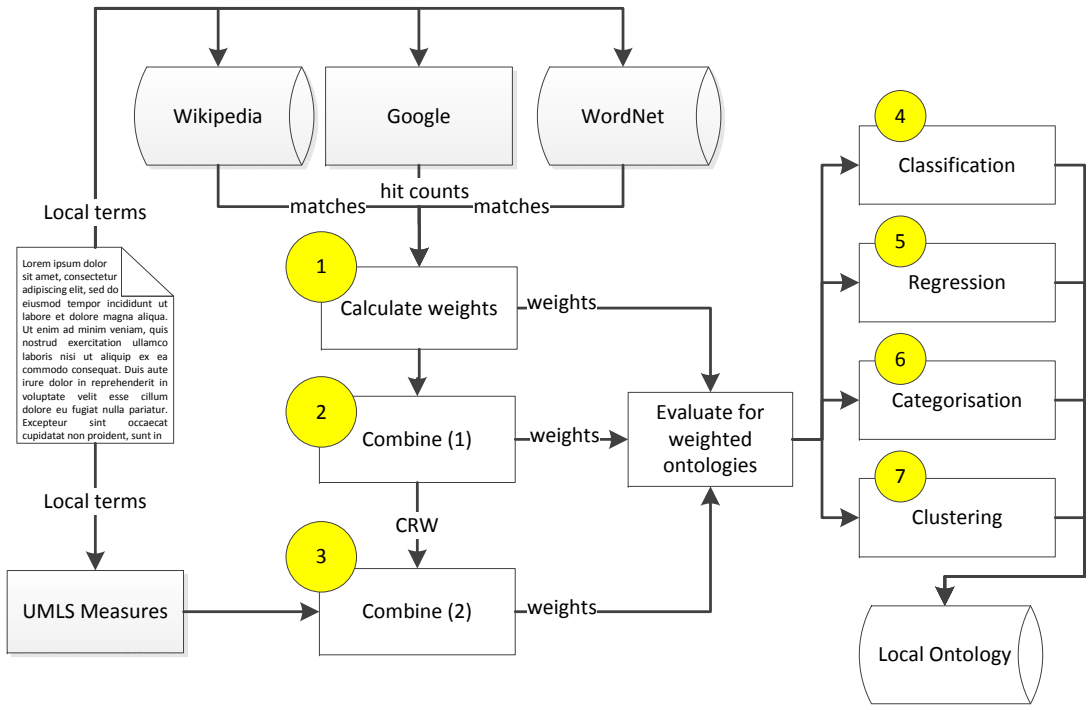


Figure 3.1: Summary of Chapter 3. The yellow circular labels are explained in text.

3.1 Introduction

Text entry is a common way to request information, with input boxes a familiar graphical user interface (GUI) element in online searches. A text query contains a set of

keywords that a user selects as relevant to the target they are interested in. However, an item can be described in many ways and these differences can lead to relevant results being overlooked, when terms used by a user differ from those used by data providers.

Keyword searches that seek synonyms — words with similar meanings, such as 'flu' and 'influenza' — of query terms can find matches that would otherwise be missed, as long as they have access to a suitable source of synonyms.

Specialised terminology, such as used in the health domain, increases the difficulty of finding relevant information for non-experts. Linked terms defined in manually-compiled terminologies or vocabularies can define specialist text and jargon, but these are often designed with a narrow scope for and by domain experts. Spatial locations are not immune from this language-based problem, given that locations can also be described in different ways (Section 2.1.1).

Measuring and recording the relationship strength between terms grants finer control over query augmentation, which can bring the domains of user and data closer together. Measurements between paired terms can be calculated for *similarity* (close in meaning, such as the synonyms 'flu' and 'influenza') or *relatedness* (often associated, such as 'hospital' and 'doctor'). This research investigates a number of relatedness measures built from the crowd-sourced Wikipedia and Google and other resources.

Relatedness measures based on different textual resources were evaluated for how well they predicted relationship strengths between sets of terms. Combinations of measures were also tested, to determine if gaps in one resource could be compensated for by matches in another.

Using relatedness weights to rank matching terms was investigated to test the effectiveness of automatically producing a weighted ontology of terms. In each case, results were evaluated against human judgements over the same term pairs.

3.2 Evaluating term-relatedness

Estimating the relatedness or similarity between terms is an ongoing natural language processing (NLP) problem, as discussed in Section 2.2.3.

It was theorised that relatedness measures from a variety of textual resources could be used to automatically build an ontology of linked terms relevant to the data source being processed. Relatedness measures were calculated based on public natural language knowledge bases including crowdsourced (Google, Wikipedia) and authoritative, manually-compiled (WordNet, health terminologies) resources.

It was further theorised that, for jargon and other specialist terminologies, relatedness measures calculated purely from general language resources could be reliable enough to build an initial ontology suitable for later refinement, although general measures were expected to be less accurate predictors of specialist term links.

This section describes the relatedness measures investigated, methods for evaluating their effectiveness in automatically building an ontology of linked terms, and results from tests against human judgements of relatedness.

3.2.1 Relatedness measures

Sources of comparative text for calculating term-relatedness included crowdsourced resources Google and Wikipedia, a public, manually compiled language resource (WordNet), and domain-specific taxonomies, in this case in the health field: the Unified Medical Language System (UMLS), incorporating a number of biomedical vocabularies. Refer to Section 2.2.1 for descriptions of the public resources used.

Relatedness measures were tested on a list of terms manually collated from World Health Organisation (WHO) information pages and Wikipedia article titles, as discussed in the *Ground truth* section below.

Google-*based measures*

Relatedness measures for terms A and B based on the Google index were calculated from their individual hit counts a and b, their intersection hit count $a \cap b$ for a search on the combination of both terms, and their union hit count $a \cup b$ for the total number of pages accessed, calculated as $a + b - (a \cap b)$.

Hit counts for individual and paired terms were extracted by a Python script that connected to the Google search engine at `www.google.com.au` with a search query of the term or pair of terms. Hit counts were extracted from the first page returned

by a successful search, which contained text in the form "*About 715,000 results (0.60 seconds)*". Other content from the Google search, including subsequent result pages and links to individual webpages, was ignored.

Pauses were built into the script for batch processing to maintain politeness and conform to Google access restrictions of approximately 5 queries per second and 200 queries per minute[45]. This increases processing time for large term sets.

Hit counts were also extracted with additional search filters for web domains of .gov (government), .org (organisation), and .edu (education) respectively to test if limiting the search to these potentially more authoritative sites would improve relatedness measure accuracy.

Three relatedness weights were measured: 1-Normalized Google Distance (NGD), Google Ratio (GR), and Google Intersection (GI). Each is based on hit counts a and b for searches on terms A and B respectively. N is a normalising factor approximating the total number of pages indexed by Google. In this research, a value of $N = 2.5 * 10^{10}$ was used, based on the hit count returned for a Google search on a common English term: 'the' or 'a': 25,270,000,000 results.

The NGD measure, described in Equation 2.4, Section 2.2, was inverted by subtracting it from one, thus converting it from a *distance* measure (smaller value showing a closer relationship) to a *relatedness* measure (higher value showing a closer relationship). This aligns 1-NGD (Equation 3.1)) with other relatedness measures.

$$\text{1-NGD}(A,B) = 1 - \frac{\max\left(\log(a), \log(b)\right) - \log(a \cap b)}{\log(N) - \min\left(\log(a), \log(b)\right)} \tag{3.1}$$

for terms A,B, hit counts a,b, and normalising factor N

It was theorised that, as some specialised terms have very low hit counts, the large NGD normalising factor N could swamp the hit counts for separate and combined term pairs, giving a false impression of low relatedness. Therefore, an unnormalised Google Ratio (GR), based on a Jaccard Index, was calculated as intersection divided by union (Equation 3.2), where the union is the individual hit counts minus the hit count of the two terms combined.

[45] https://developers.{gi}google.com/webmaster-tools/v3/limits, accessed 22/1/2017.

$$\mathrm{GR}(A,B) = \frac{a \cap b}{a \cup b} \tag{3.2}$$

for terms A,B and Google hit counts a,b

The Google Intersection (GI) measure considers the proportion of indexed resources for each term that intersects with its paired term (Equation 3.3). It aims to account for pairs of terms where one is much more common than the other.

$$\text{Intersection Ratio } \mathrm{IR}_{ab} = \begin{cases} 0, & \text{if } a \cap b = 0 \\ \dfrac{a \cap b}{a}, & \text{if } a \cap b < a \\ 1, & \text{otherwise} \end{cases}$$

$$\mathrm{GI}(A,B) = (\mathrm{IR}_{ab} + \mathrm{IR}_{ba}) \div 2 \tag{3.3}$$

for terms A,B and Google hit counts a,b

These Google-related measures are all symmetrical. That is, the weight value calculated is the same for terms $A \rightarrow B$ and $B \rightarrow A$, irrespective of their relative hit count magnitudes.

Wikipedia-*based measure*

Wikipedia Link-based Measure (WLM) values (Section 2.2.3) were extracted from an instance of the Wikipedia Miner (WM) toolkit (Milne & Witten, 2013), using a dump of the Simple English Wikipedia from April 2014[46]. The Simple English language version has a much smaller coverage than the Full English version, increasing the likelihood of a term not matching to any article title. This limitation on article-matching terms acts as a proxy for highly specialised terminology.

WordNet-*based measure*

WN was calculated by comparing paired terms in WordNet (version 3.0) via the Python nltk library to search for synonyms, hypernyms and hyponyms. These relationships are reflexive (Tangi, 1998) so, if a relationship was found from $A \rightarrow B$, the reflexive relationship from $B \rightarrow A$ was assumed and both were assigned a weight of 1.0. The

[46] http://dumps.wikimedia.org/simplewiki/, accessed 25/4/2014.

search was extended to a maximum depth of two links between synsets, i.e. terms $A \rightarrow B \rightarrow C$.

For each term, WordNet was searched for a matching synset sense. Where there was more than one synset match for a word, the first match was used. This introduces some limitations: for example, the term 'dog' has several possible noun and verb synsets but only one of these (the first returned) is a hypernym of 'poodle'. This means that a hypernym/hyponym relationship between terms is not necessarily reflexive.

For terms A and B, a weight of one was allocated if any of the following occurred:

- A and B occur in the same synset (i.e. they are synonyms),
- A is in a synset that is a hypernym of the B synset (i.e. B is a type of A), or
- A is in a synset that is a hyponym of the B synset (i.e. A is a type of B).

Where none of these three relationship types were found between paired terms, they were each assigned a weight of zero. As the number of terms in WordNet is limited, any relationship to a term not defined in WordNet was also assigned a value of zero. Similarly, these terms would have a WN of zero to themselves - that is, if term A is not found in WordNet, the weight from $A \rightarrow A$ is zero.

Biomedical and health measures

Relatedness between specialist health terms was determined using the UMLS::Similarity and UMLS::Relatedness measures (McInnes *et al.*, 2009, 2013), based on the UMLS resource. The UMLS::Similarity Web Interface[47]. was used to calculate similarity and relatedness measures in the $0.0 \ldots 1.0$ range from sets of biomedical terms.

UMLS::Similarity was calculated using a path length measure over hierarchical UMLS terminologies; in this case Medical Subject Headings (MeSH). For UMLS::Relatedness, all UMLS sources were searched to match a search term to a definition, which was then extended and converted into a vector of terms for each initial comparison term. Cosine similarity of vectors was used to calculate the relatedness weight between terms.

[47] http://umls-similarity.sourceforge.net, accessed 26-27/7/2016.

Composite measures

It was hypothesised that combining measures from multiple textual resources could improve upon the reliability and flexibility of individual measures. For example, by including measures from different resources to alleviate the problem of gaps where one resource becomes unavailable or does not recognise a particular term.

To test the hypothesis, individual GR, GI, NGD, WLM, and WN measures were calculated for each pair of terms in a test set, with Google measures determined separately for web domains *all* (unfiltered), *.gov*, *.org* and *.edu*. UMLS::Similarity and UMLS::Relatedness weights were also collected for health-specific term sets to test if composite measures could be adapted to a new knowledge domain by encompassing specialist resources and measures.

Unweighted L_2 Frobenius Norm combinations of individual measures were calculated, and these composite estimates were compared statistically against ground truth relatedness values for each pair of terms.

Ground truth

The generation of relatedness measurements, and hence terminological ontologies, requires input terms that are representative of textual content for the resource being processed. A wide variety of term sources can be used, including domain-specific vocabularies and text taken directly from data records. For validation, some pairs of terms within the set need to be weighted according to human judgements of similarity or relatedness.

In all tests within this chapter, sets of paired terms were annotated with human judgements of relatedness strength. Where necessary, judgements were scaled to the 0.0...1.0 (unrelated...related) range so as to be comparable to each other. For example, the Ped30 test set of medical terms collected responses in the range 1-4: "not at all related" to "very closely related" (Pedersen *et al.*, 2007).

MixMed datasets

A ground truth test set of 50 general and health-related terms — *MixMed*-50, listed in Appendix A — was developed for testing. This represents a subset of terms of interest

within a data resource; in this case, a set of medical information webpages targeted at the general public.

Initially, twenty-four health-related terms were selected from online WHO information pages[48], including eight each of health conditions (such as diabetes, asthma or cancer), categories (such as infectious disease or genetic disorder) and general medical terminology (such as disease, vaccination or virus). An additional eight unrelated terms were added, chosen from articles one or two links away from an English Wikipedia article on one of the initial 24 topics. For example, the term 'poodle' was included because the English Wikipedia article on 'diabetes' contained a link (entitled 'Miniature poodles') to the 'poodle' article:

> "In animals, diabetes is most commonly encountered in dogs and cats. [...] some small dog breeds are particularly likely to develop diabetes, such as *Miniature Poodles*."[49]

From these 32 seed terms, 25 were selected that could be matched to article titles in the 2014 Simple English dump, using the suggest service of a Wikipedia Miner (WM) toolkit instance.

This set of 25 seed terms was augmented with an additional 25 terms, chosen by linking one or two steps away from one of the original 25 Wikipedia articles. In this case, outward links returned by the exploreArticle service on the same WM instance were used, where the reported link weight was greater than 0.4 from the original article. For example, 'cattle' and 'plow' were added as 'vaccination' linked to 'cattle' and 'cattle' linked to 'plow', with relatedness weights of 0.459 and 0.53 respectively.

Each of the 50 *MixMed*-50 terms was manually compared to all others and given estimated relatedness scores between 0.0 (unrelated) and 1.0 (strongly related, though not necessarily synonymous). The human annotator was instructed to label the relatedness of each paired term as one of ['not', 'slightly', 'somewhat', 'moderately', 'highly']. The labels were translated directly into numeric weights $[0.0, 0.3, 0.6, 0.8, 1.0]$.

These weights, tabled in Appendix A, defined the ground truth used for calculations and comparisons. A binary relatedness grade based on these scores was assigned to

[48] http://www.who.int/topics/en/, accessed 22/12/2014.
[49] https://en.wikipedia.org/wiki/Diabetes_mellitus#Other_animals, accessed 1/10/2015.

each pair: *related* ($\geq T$) and *unrelated* ($< T$), where T is a threshold ground truth relatedness weight ($T = 0.5$ except where specified otherwise).

An additional 152 terms were added to *MixMed*-50 to create an extended *MixMed*-202 of terms. This set, also listed in Appendix A, includes 52 diseases and nine health care-related terms as well as general terms.

Health datasets

Annotated paired-term sets for biomedical and health terminology were used to evaluate results for the specialist health domain. These typically use medical personnel such as physicians or medical coders to judge relatedness. Test sets used were the Pak101 Medical Coders[50] and Ped30 sets (Pakhomov *et al.*, 2011; Pedersen *et al.*, 2007).

There are 183 unique biomedical terms in Pak101 and 59 in Ped30. There is some overlap, with 29 terms appearing in both lists, giving 213 unique terms in total. This includes n-grams containing two or more words, and 15 misspelled terms such as 'rheumatoid arthr*iits*'. Spelling errors were corrected before extracting results. Most terms appear in only one pair, with no more than three appearances of a single term within one dataset.

Of the 213 unique terms, 180 match to titles of full English Wikipedia articles and 81 match to simple English articles, as of mid-2016. Several of these articles are placeholders to redirect to a main article, such as 'myoca*ridal* infarction' → 'Myocardial_infarction' and 'premature labor' → 'Preterm_birth'.

3.2.2 Evaluating measures

This section describes results from statistical testing of individual and combination measures against ground truth datasets to evaluate their effectiveness in predicting human judgements of relatedness.

To be considered successful, an automated relatedness measure needed to perform better than chance at predicting relatedness weights between term pairs. This relatively low match prediction threshold was tolerated due to the flexibility of natural languages and subjectivity of human judgements used to determine ground truth values.

[50] http://rxinformatics.umn.edu/SemanticRelatednessResources.html, accessed 25/5/2016.

A number of tests were used to measure performance, including direct comparison by subtracting the ground truth value from the calculated measure for the same pair of terms, and statistical classification measures (Section 2.6.2).

In the field of information retrieval (IR), results returned by a search are considered either positive (relevant to the searcher) or negative (not relevant). Statistics on positive and negative results can be used to demonstrate how successful a search method has been. This approach was applied to testing term-relatedness measures.

For this purpose, ground truth values were simplified into binary classes: related (weight ≥ 0.5) or unrelated (weight < 0.5). For the *MixMed*-50 set, this resulted in $R = 860$ related and $U = 1640$ unrelated term pairs.

Calculated measures were classed as positive (weight $\geq T$) or negative (weight $< T$) at relatedness threshold values $T = [0.1, 0.2, \ldots, 0.9]$. Related pairs classified as positive are true positives (TP), and *un*related pairs classified as positive are false positives (FP). Statistical methods described in Section 2.6.2 were used to judge the success of measures, where perfect classifiers would have TPR=P=A=F_1=1 and FPR=0. The target performance measure is above or below 0.5, respectively.

Individual measures

The three measures based on Google hit counts were compared against *MixMed*-50 ground truth values for filtered domains. Results showed that limiting indexed resources to theoretically more authoritative webpages (by filtering for government, organisation or educational domains) did not improve results, and in some cases reduced the performance to less than chance. This effect can be seen by subtracting ground truth from calculated measures, as illustrated in the boxplots in Figure 3.2.

This figure is a detail view as it does not show all outliers for the GR measure, which ranged between -16.9 and 56.9. It highlights an underestimation of relatedness by the Google-based ratio measures GR and GI, in comparison to the 1-NGD measure. The distribution of the inverted NGD measure is closer to the ground truth value (i.e. a difference of zero), particularly for unfiltered hit counts (labelled 'all').

The GR results followed an exponential trend, even after the removal of outlier values more than three standard deviations from the mean, as illustrated in Figure 3.3. Re-

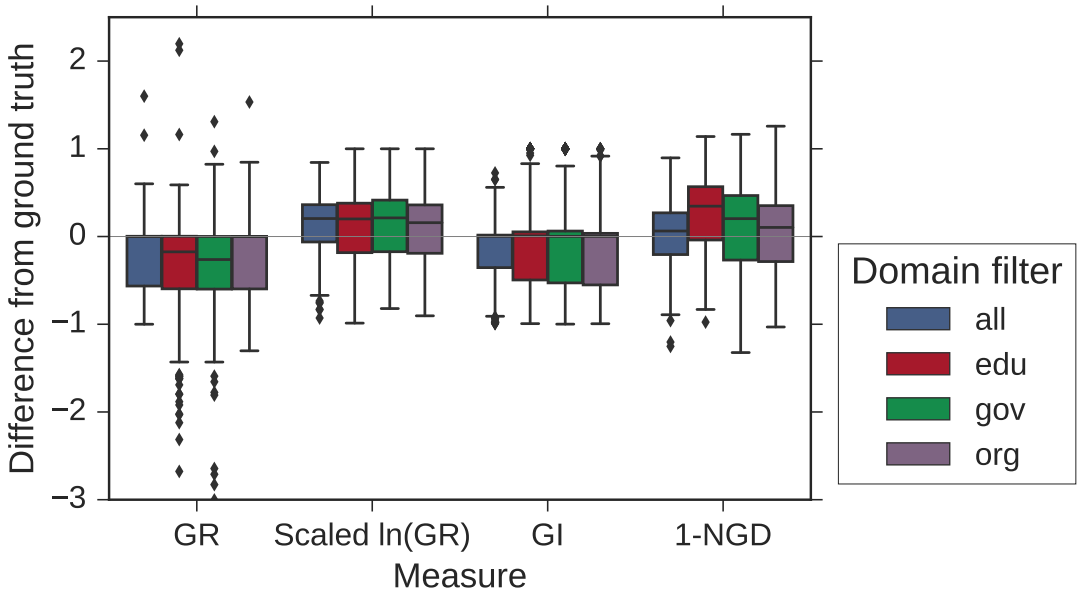
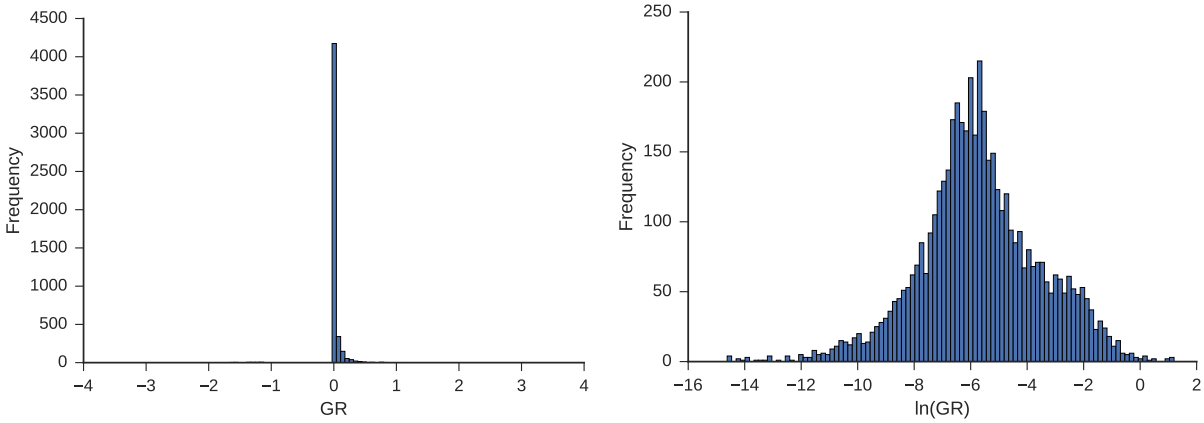


Figure 3.2: Boxplots of Google-based relatedness measures compared against *MixMed*-50 ground truth values, filtered by web domain (detail view: GR had outliers from -17 to 57).

moval of outlier GR weights from *MixMed*-50 links left 4,879 paired relatedness weights out of a possible 4,900. As a result of the exponential trend, the natural logarithm of the GR measure (scaled to the $0.0 \ldots 1.0$ range) was included in tests to discover how, or if, a more normalised distribution of ln(GR) affected results.



(i) GR (ii) ln(GR)

Figure 3.3: 100-bin histograms of Google Ratio measures over *MixMed*-50 with outliers ($> 3\sigma$) removed.

The calculated relatedness values are plotted against ground truth values in Figure 3.4. The vertical bands of points are an artefact of the human judgements over *MixMed*-50, which were selected from values $[0.0, 0.3, 0.6, 0.8, 1.0]$. Each measure is shown with

a linear regression line and accompanying highlighted area for the 95% confidence interval. The unfiltered GR measure, for example, has a large uncertainty element in this linear estimate, due to its exponential nature.

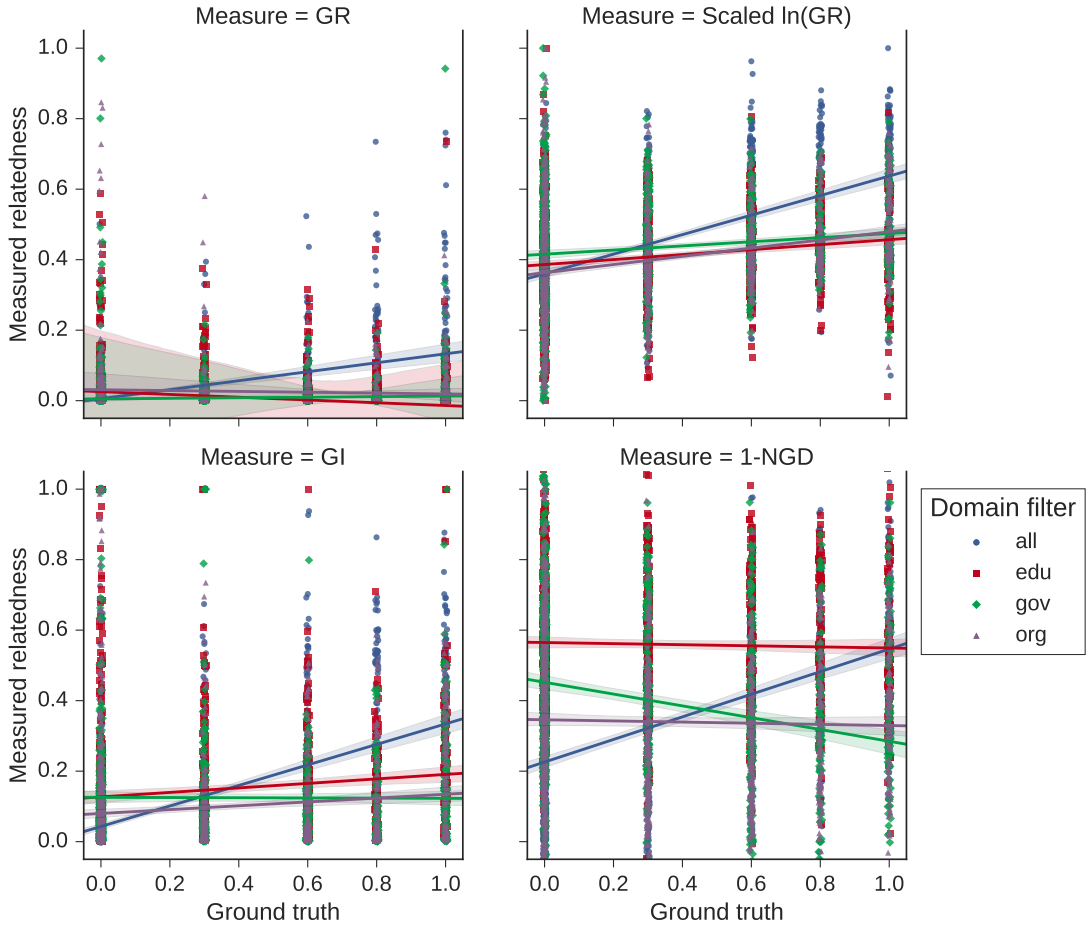


Figure 3.4: Scatterplots with linear regression of Google-based relatedness measures compared against *MixMed*-50 ground truth values ($[0, 0.3, 0.6, 0.8, 1.0]$), filtered by web domain.

Unfiltered calculations in each case show a more positive relationship than edu, gov, or org filtered ones. This is demonstrated most clearly with the 1-NGD measure, with negative relationships for all filtered results. It is theorised that the smaller number of indexed resources in specific domains contributed to this effect.

As a result of these findings, the unfiltered 1-NGD measure was considered the most promising Google-based measure, although other Google measures were also tested within combination measures.

Each of the terminology resources used to calculate relatedness measures — Google, Wikipedia and WordNet — contribute differing strengths and weaknesses to a general

combined relatedness measure.

The receiver operating characteristic (ROC) curve in Figure 3.5 plots the performance for the WLM, 1-NGD, and WN measures as binary classifiers against the *MixMed*-50 dataset, where a ground truth value of 0.5 or above counts as *related*, for threshold values of each measure from $T = [0.1, 0.2, \ldots, 0.9]$.

From this figure, it can be seen that WN did not predict relatedness well, given the low numbers of matches between *MixMed*-50 and WordNet terms: only 25 of the 50 terms matched to a synset. Of these, only three were directly related: ['genetic disorder' $\leftrightarrow$ 'disease'] and ['disease' $\leftrightarrow$ 'illness'], both belonging to the same synset, and ['cholera' $\leftrightarrow$ 'infectious disease'], where 'cholera' is a hyponym of 'infectious disease'.

With these low match values, results for WN were clustered around maximum or minimum TPR and FPR, depending upon the threshold value. This gives a neutral response on the ROC plot. However, although results were sparse, the authoritative nature of WordNet means that any matches discovered were reliable. For this reason, it was included in combination weight tests.

Figure 3.5 shows that the WLM and 1-NGD measures performed better than WN, particularly around a relatively low threshold of 0.3 (WLM) to 0.4 (1-NGD). Given that the definition of the *related* class used a ground truth threshold of 0.5, the threshold of $T = 0.5$ was of particular interest.

More detail about the performance of WLM and 1-NGD measures over *MixMed*-50 at the different threshold values can be seen in Tables 3.1—3.2 where the row for the mid-point threshold of $T = 0.5$ has been highlighted.

It can be seen from Table 3.2 that precision for 1-NGD was low, although it was neutral or better at thresholds of 0.5 and above. Precision of WLM (Table 3.1) was higher, but both measures understandably suffered from a drop-off in TPR at higher thresholds.

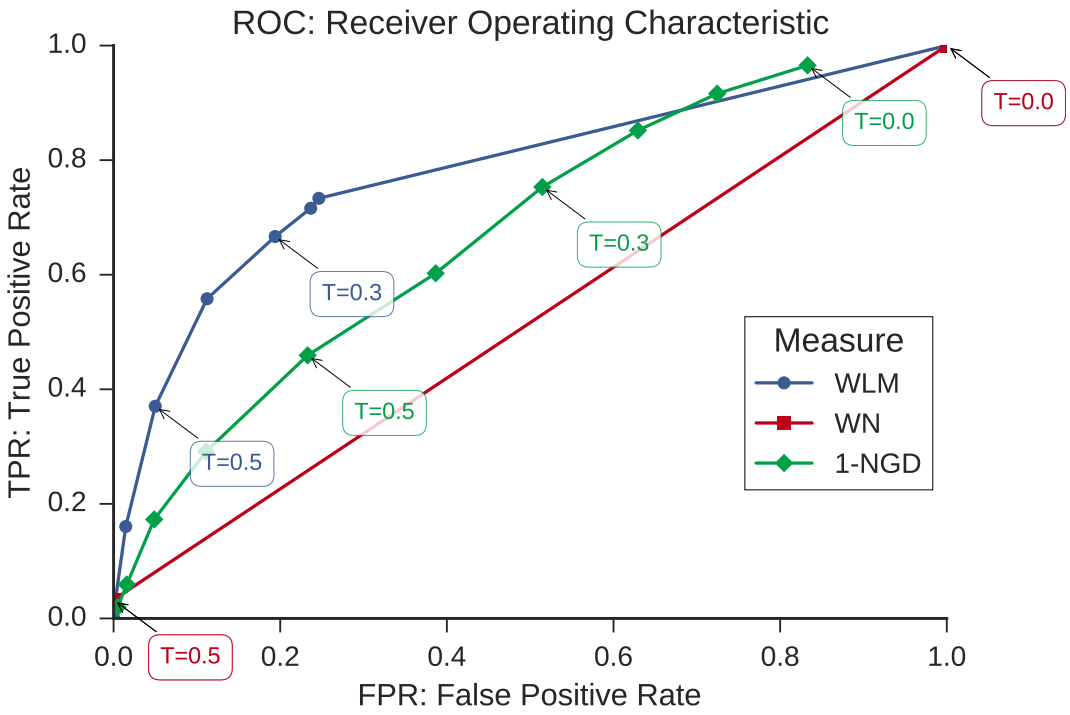


Figure 3.5: ROC for relatedness measures over *MixMed*-50 for threshold values $T = [0.1, 0.2, \ldots, 0.9]$.

Table 3.1: Performance of the WLM measure over *MixMed*-50, where paired terms with calculated relatedness over the threshold T are classed as 'related'.

T	TPR	FPR	P	A	F_1
0.1	0.733	0.246	0.595	0.747	0.657
0.2	0.716	0.237	0.599	0.748	0.652
0.3	0.667	0.194	0.629	0.760	0.647
0.4	0.558	0.112	0.711	0.779	0.625
0.5	0.37	0.05	0.785	0.758	0.503
0.6	0.16	0.015	0.844	0.713	0.270
0.7	0.035	0.002	0.875	0.679	0.067
0.8	0	0	1	0.669	0
0.9	0	0	1	0.669	0

Table 3.2: Performance of the 1-NGD measure over *MixMed*-50, where paired terms with calculated relatedness over the threshold T are classed as 'related'.

T	TPR	FPR	P	A	F_1
0.1	0.921	0.724	0.386	0.489	0.544
0.2	0.857	0.629	0.402	0.531	0.547
0.3	0.758	0.515	0.421	0.576	0.541
0.4	0.607	0.387	0.437	0.611	0.508
0.5	0.464	0.233	0.496	0.667	0.480
0.6	0.296	0.111	0.569	0.693	0.390
0.7	0.178	0.049	0.643	0.696	0.279
0.8	0.064	0.016	0.667	0.680	0.117
0.9	0.025	0.005	0.714	0.674	0.048

Combination measures

Unweighted, i.e. evenly-proportioned mixtures, of individual measures were also tested statistically against ground truth relatedness weights to test if they met the requirement to predict relatedness at better than chance levels (label 2 in Figure 3.1). Specialist measures were also combined with generic resources and measures for testing (label 3 in Figure 3.1).

Measures were combined using an L_2 Frobenius Norm (Euclidean distance), scaled to the $0.0 \ldots 1.0$ range. A combination of WLM, WN and 1-NGD is henceforth called Combined Resources Weight (CRW). The 1-NGD measure was calculated without domain filtering. WordNet was included in spite of the sparsity of its term matches due to the reliability of its manually-collated resource.

Statistical binary classifier results from the CRW measure are listed in Table 3.3, for relatedness threshold values $T = [0.1, 0.2, \ldots, 0.9]$ where true relatedness is classified as ground truth ≥ 0.5. In comparison with Tables 3.1 and 3.2, it yields higher TPR, precision, accuracy and F_1 values and comparably low FPR values at threshold $T = 0.5$. Although the TPR of 0.477 is roughly equivalent to chance (0.5), other performance measures are above the target of 0.5 or greater. As illustrated in Figure 3.6, the area under the ROC curve is greater for the combined measure than any of the individual contributors.

Table 3.3: Performance of the CRW combination measure (L_2 norm of WLM, WN and 1-NGD, scaled to 0.0...1.0) over *MixMed*-50, where paired terms with calculated relatedness over the threshold T are classed as 'related'.

T	TPR	FPR	P	A	F_1
0.1	0.956	0.800	0.371	0.450	0.417
0.2	0.884	0.570	0.434	0.580	0.454
0.3	0.785	0.300	0.564	0.728	0.517
0.4	0.652	0.148	0.686	0.786	0.562
0.5	0.477	0.052	0.818	0.792	0.602
0.6	0.264	0.021	0.863	0.743	0.614
0.7	0.114	0.005	0.920	0.704	0.628
0.8	0.030	0.001	0.923	0.678	0.629
0.9	0.012	0.000	1.000	0.673	0.645

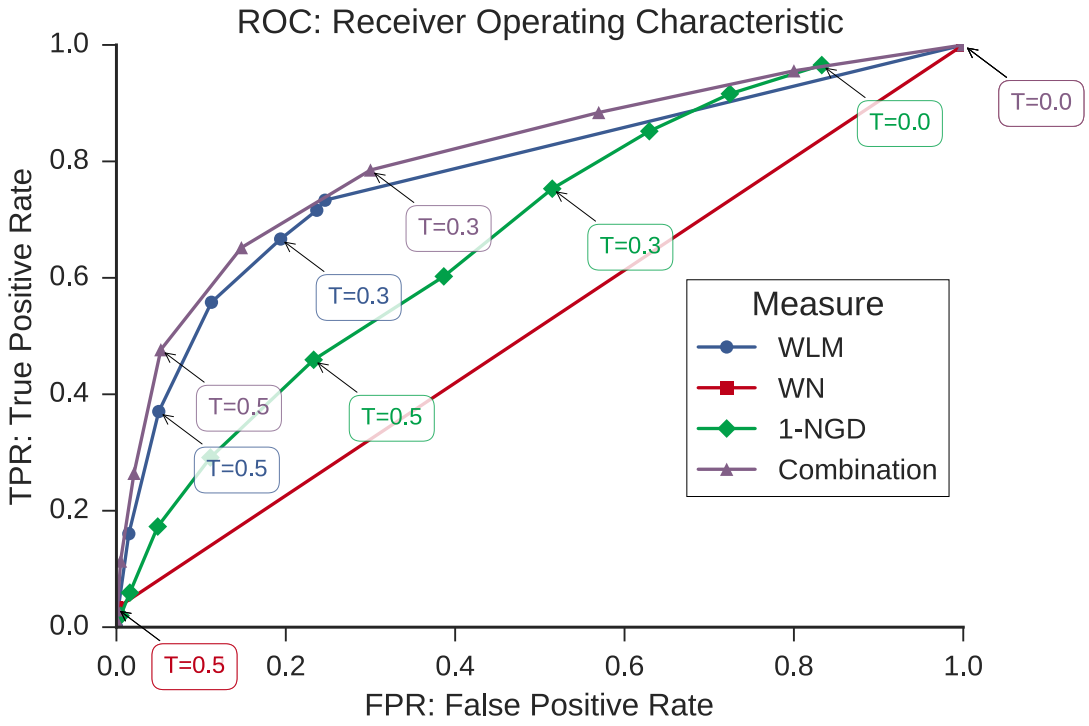


Figure 3.6: ROC for individual and combined relatedness measures over *MixMed*-50 for threshold values $T = [0.1, 0.2, \ldots, 0.9]$. The combination measure is CRW.

Individual results of this combined measure plotted against the *MixMed*-50 ground truth values are illustrated in Figure 3.7. The broad spread of measurements at each ground truth value was expected, given the subjective nature of natural languages. CRW showed a positive trend between calculated and ground truth weights, with fewer false negatives for larger ground truth values than for individual WLM or WN measures, and fewer false positives for smaller ground truth values than the 1-NGD measure.

To test the theory that general measures could predict relatedness between specialist

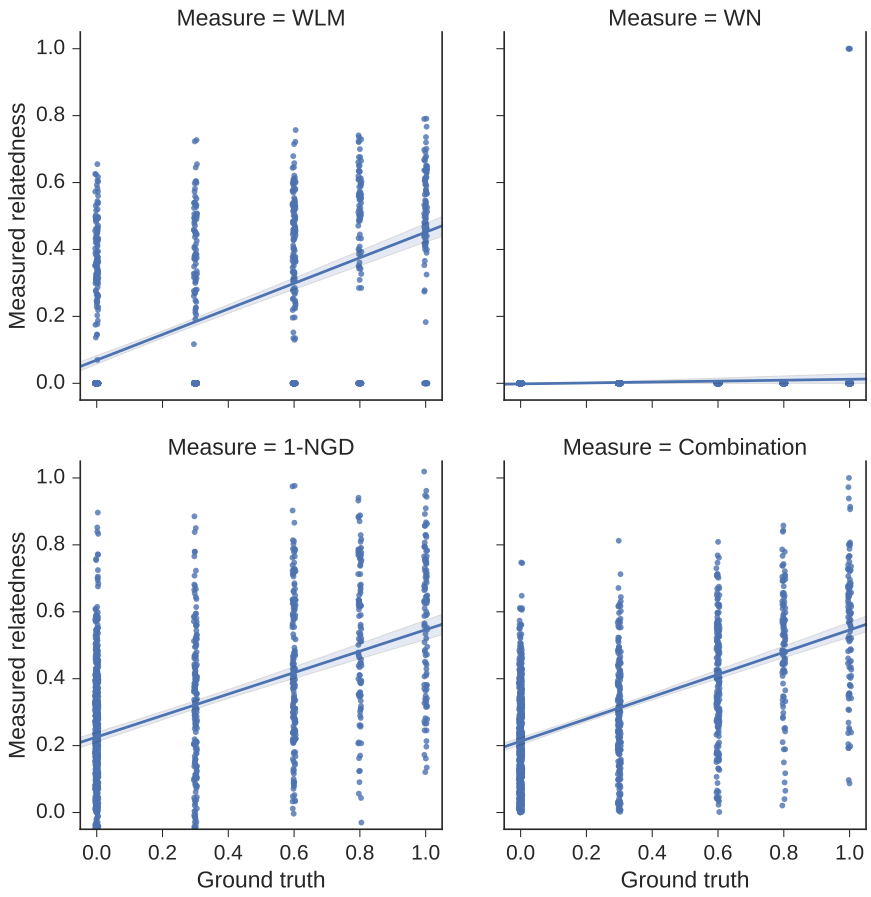


Figure 3.7: Scatterplots with linear regression of relatedness measures compared against *MixMed*-50 ground truth values ($[0, 0.3, 0.6, 0.8, 1.0]$). The combination measure is CRW.

terms, even if less effectively than for general vocabularies, measures were applied to health and biomedical term sets with attached human judgements: Ped30 and Pak101. Results were compared with those from the more general *MixMed*-50 dataset.

The proportion of paired terms yielding valid datapoints for the WLM measure, and consequently for combination measures that use it, were reduced in comparison to 1-NGD. WLM results were returned for 11 out of 101 Pak101 and 9 out of 30 Ped30 term pairs. This is attributable to the lower rate of Simple English Wikipedia article matches to the specialised biomedical terms: 81 article matches out of 213 unique terms, compared to 180 for the full English version. The WN measure has a high proportion of zero values, where no match was found. The impact of missing datapoints is illustrated in scatterplots for the health datasets with missing WLM data points removed (Figure 3.8).

In the same figure, linear regression lines on the 1-NGD measure show a negative

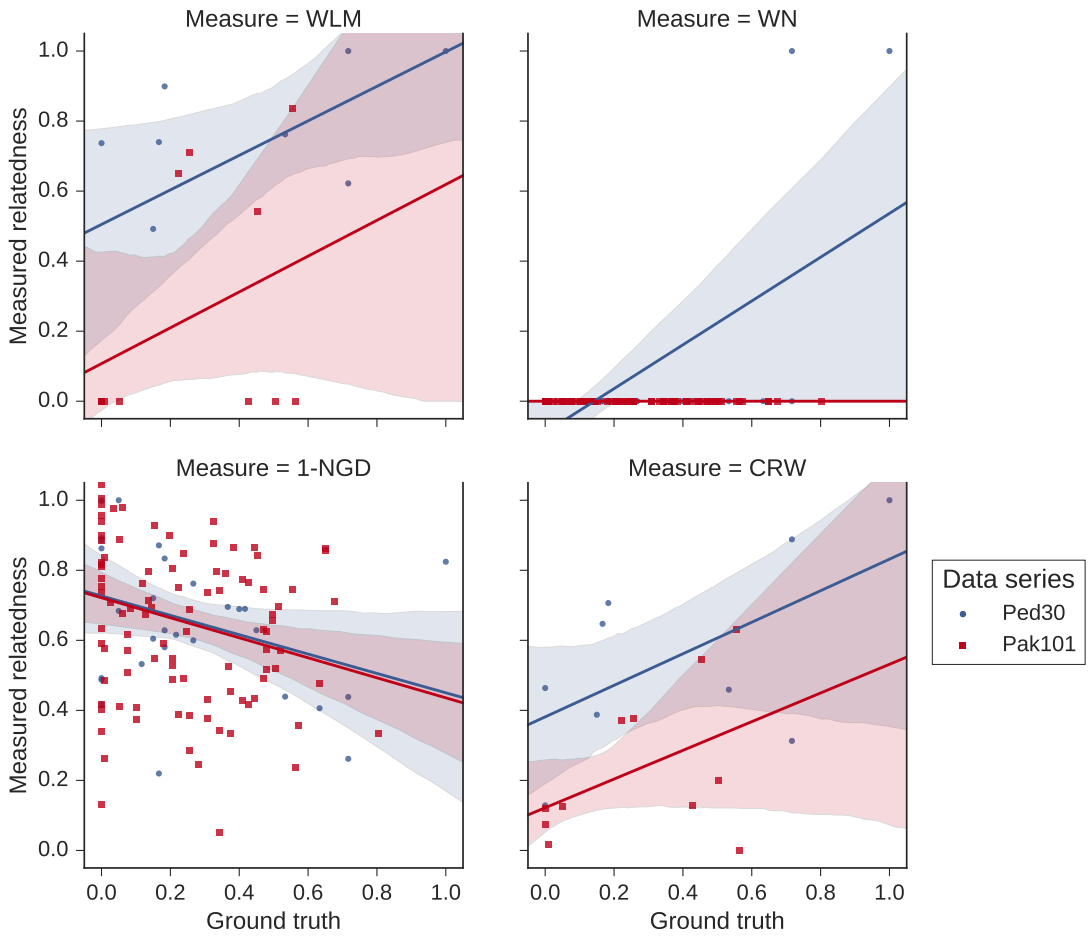


Figure 3.8: Scatterplots with linear regression of relatedness measures compared against health terminology datasets. Datapoints with unknown values have been removed.

relationship between calculated and actual results, opposite to expected results. The combination CRW measure has smoothed out some of the ill effects as it did for the more general terminology in *MixMed*-50. However, some relatedness values could not be calculated due to missing individual measures. There is also a wider margin for error, as shown by the highlighted 95% confidence interval around each regression line.

Replacing missing WLM values with 0.0 and recalculating CRW narrowed the margin of error, as illustrated in Figure 3.9. It has also resulted in a negative correlation between CRW and ground truth for the larger Pak101 set of biomedical term pairs.

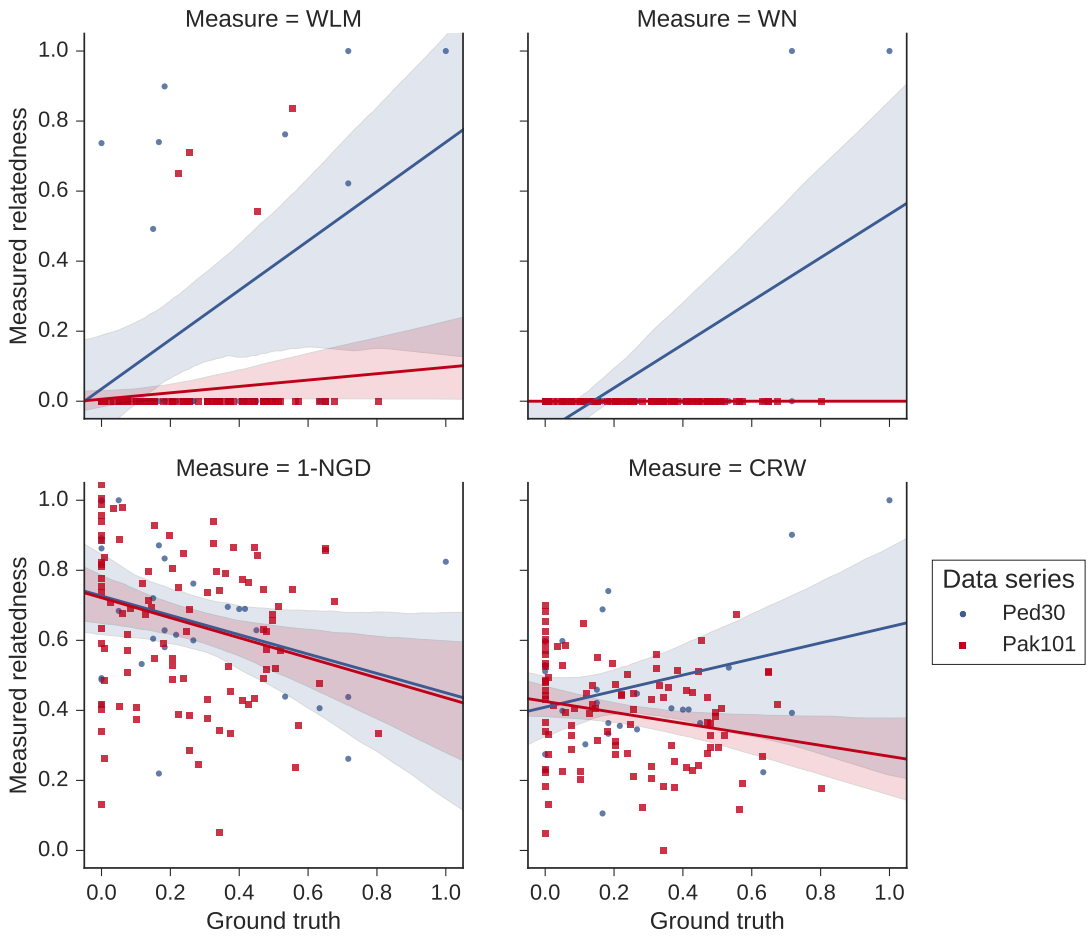


Figure 3.9: Scatterplots of relatedness measures compared against health terminology datasets. Unknown values have been replaced with zeros.

In comparison to the more general *MixMed*-50 set (see Figure 3.7), the relationship between individual and CRW measures and the ground truth is weaker for specialised terminology and, in some cases, the relationship is inverted.

The boxplot at Figure 3.10 compares performance of the mixed and health datasets. Any invalid datapoints, including paired Pak101 terms with incalculable WLM values, have been removed from the plotted data. This plot shows that the combination CRW measure clustered most strongly near the ground truth values in the *MixMed*-50 terminology dataset, and deviated less from ground truth than 1-NGD alone. The combination measure was more successful over the larger Pak101 set than for Ped30, where approximately three quarters of the term pair relatedness was overestimated. In all cases, the combination smoothed the extremes of individual WLM, WN and inverted NGD measures.

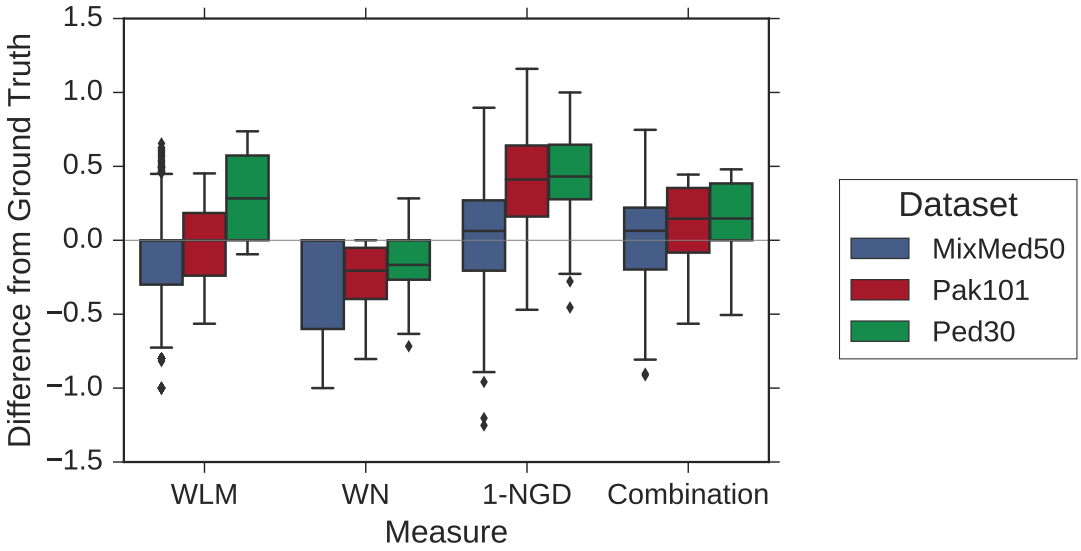


Figure 3.10: Boxplots of relatedness measures compared against *MixMed*-50 ground truth values, filtered by dataset used. The combination measure is CRW.

Relatedness measures tailored to the specialist domain were also tested against ground truth values in the Pak101 and Ped30 test sets. They were compared both as originally intended by their designers — as stand-alone measures — and in combination with the CRW measures of 1-NGD, WLM and WN.

The UMLS measures were recorded for spelling-corrected terms, with three Ped30 and twelve Pak101 terms corrected. Of the combined 131 term pairs, 70 had matches in UMLS::Similarity and 123 in UMLS::Relatedness, in comparison with 20 matches in the simple English WLM.

The impact of adding these additional specialist measures to the general CRW measure is illustrated in Figure 3.11. As expected, the UMLS::Relatedness measure was the best predictor of health term-relatedness. Augmenting the CRW measure by including UMLS::Relatedness improved its outcomes, particularly for the Pak101 set. This implies that incorporating specialist measures with general crowdsourced measures can improve the quality of a composite measure for mixed specialist and general terms.

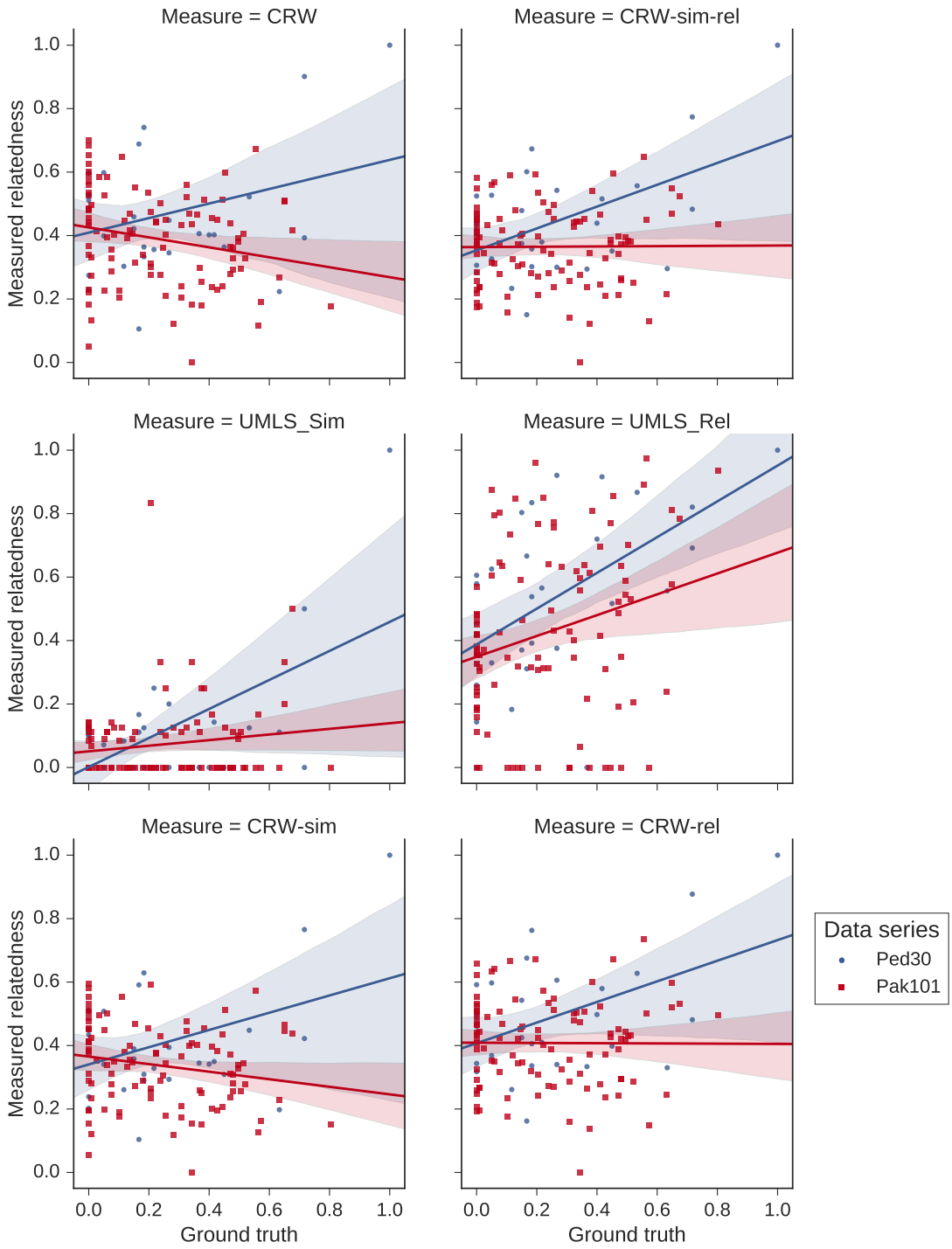


Figure 3.11: Scatterplots of CRW relatedness with the addition of specialist measures (from UMLS) compared against health terminology datasets. Unknown values have been replaced with zeros.

In conclusion, a combination of individual relatedness weights was shown to be at least as effective a predictor of general term link strength as the individual measures, and in general was more robust. Performance was reduced for specialist vocabularies when

using general resources such as Wikipedia and Google, but incorporating domain-specific measures into the general CRW measure provided a reasonable combined relatedness weight for jargon.

The goal for the individual and combined weight measures tested in this section was to assist in automated building of data-source-specific ontologies of terms. This will be explored further in the next section.

3.3 Discovering related terms

This section uses relatedness measures to discover related term pairs that can be added to an ontology of terms. Classifiers using machine learning techniques were used to find term links (label 4 in Figure 3.1). Estimation of link confidence levels and probabilities were also investigated for relevance to the creation of weighted ontologies (label 5 in Figure 3.1).

3.3.1 Binary classifiers

Two supervised machine learning binary classifiers — SVM and Multilayer Perceptron (MLP) — were trained on the ground truth test sets. In both cases, input data tables were created with a row for each unique pair of terms, containing individual relatedness measures and the ground truth weight. Term order was ignored as symmetrical relatedness was assumed.

The classifiers were trained over unique pairs from *MixMed*-50 with ground truth values < 0.5 (*unrelated*) and ≥ 0.5 (*related*). Out of 1225 term pairs, 820 were unrelated and 405 were related. For processing in the Waikato Environment for Knowledge Analysis (Weka) machine learning toolkit (Hall *et al.*, 2009), $n = 1225$ data rows were saved in an ARFF (Attribute-Relation File Format)[51] file.

The classifiers were then re-trained over terms from a specialised domain: health. Human judgements from the Ped30 and Pak101 health datasets were combined in an ARFF file with 131 data rows. Each row included attributes for unfiltered 1-NGD, ln(GR), WLM, and WN. Out of 131 term pairs, 16 were classified as related and 115 as

[51] http://weka.wikispaces.com/ARFF, accessed 4/4/2016.

unrelated at a ground truth threshold ≥ 0.5. An alternative classification was obtained by relaxing the threshold to ground truth ≥ 0.4, giving 34 related and 97 unrelated pairs.

A Weka implementation[52] of SVM from the libSVM library (Chang & Lin, 2011) was used to determine the related/unrelated classification of term pairs. Default input parameter values were used, including a gamma value of 0.0.

After training, results from the models were assessed for validity by comparing against ground truth classes, using 10-fold cross-validation. Classification results for different combinations of measures are shown in Table 3.4.

Table 3.4: Support Vector Machine binary classification evaluation over *MixMed*-50. 1-NGD* = 3× 1-NGD: one measure for each of all, gov and org domains.

	Measure	A	Kappa	MAE	RMSE	RAE	RRSE
	perfect	1.0	1.0	0.0	0.0	0%	0%
i	WLM	0.758	0.459	0.242	0.492	54.8%	104.7%
ii	1-NGD	0.669	0.003	0.331	0.576	74.9%	122.4%
iii	ln(GR),1-NGD*	0.743	0.353	0.257	0.507	58.1%	107.8%
iv	ln(GR),1-NGD*,WLM,WN	0.784	0.483	0.216	0.465	48.9%	98.9%
v	1-NGD,WLM,WN	0.784	0.494	0.216	0.465	48.9%	98.9%
vi	1-NGD,WLM	0.778	0.471	0.222	0.471	50.2%	100.2%
vii	WLM,WN	0.753	0.453	0.247	0.497	55.7%	105.5%

Confusion matrices of correctly and incorrectly classified paired terms from four of these measures are shown in Table 3.5, where r is the number of pairs classified by the model as related, and u is the number classified as unrelated. T and F represent the ground truth related state — true and false, respectively. Note that 1-NGD alone (Measure ii) classified most pairs as unrelated.

[52] https://weka.wikispaces.com/LibSVM, accessed 25/7/2016.

Table 3.5: SVM confusion matrices with related (r) and unrelated (u) counts over four measures (as listed in Table 3.4) on *MixMed*-50. Rows show true counts of related (T) and unrelated (F) pairs. Columns show calculated counts.

	Measure i		Measure ii		Measure iv		Measure v		
	r	u	r	u	r	u	r	u	$\sum$
T	266	139	3	402	229	176	245	160	405
F	158	662	4	816	89	731	105	715	820

The same individual and combined measures, excluding filtered Google counts, were tested over Ped30 and Pak101 paired health terms. The individual WLM weight (measure i) returned a false positive, but other measures classified all pairs as unrelated. This gives a potentially misleading correct classification rate of 87.8% because of the low related:unrelated ratio in ground truth values. Similar results were obtained when the class imbalance was addressed by lowering the ground truth threshold to weight ≥ 0.4.

The WLM confusion matrices (Table 3.6) illustrate the issues created by training SVM models over CRW component measures to classify highly specialised terms. At thresholds of $T = 0.4$ and $T = 0.5$, the only pairs classified as related were false positives.

Table 3.6: SVM confusion matrices with related (r) and unrelated (u) counts over the WLM measure on Ped30 and Pak101 term pairs. Rows show true counts of related (T) and unrelated (F) pairs. Columns show calculated counts.

	i WLM ($T = 0.4$)		i WLM ($T = 0.5$)		$T = [0.4/0.5]$
	r	u	r	u	$\sum$
T	0	34	0	16	[34/16]
F	2	95	1	114	[97/115]

When SVM models were retrained with general measures in combination with specialist health measures UMLS::Similarity and/or UMLS::Relatedness, all term pairs were unrelated for all such composite measures. Training purely on the UMLS::Similarity and UMLS::Relatedness measures, separately or together, gave the same result.

An implementation of MLP from the Weka toolkit[53] (Hall *et al.*, 2009) was used to determine classes for each relationship. Default input parameter values were used, including 500 training epochs, momentum of 0.2, and a learning rate of 0.3. Two hidden layer arrangements were tested: the default ((number of attributes + number of classes) ÷ 2) and two layers with three and five nodes respectively.

After training, results from the models were assessed for validity by comparing against ground truth classes, using 10-fold cross-validation.

Evaluation results for different combinations of measures are shown in Table 3.7, with confusion matrices for a selection of measures in Table 3.8.

Table 3.7: Multilayer Perceptron binary classification over *MixMed*-50.
1-NGD* = 3× 1-NGD: one measure for each of all, gov and org domains.

	Measure	A	Kappa	MAE	RMSE	RAE	RRSE
	perfect	1.0	1.0	0.0	0.0	0%	0%
i	WLM	0.776	0.455	0.315	0.407	71.2%	86.5%
ii	1-NGD	0.687	0.128	0.391	0.454	88.4%	96.5%
iii	ln(GR),1-NGD*	0.726	0.290	0.341	0.425	76.9%	90.3%
iv	ln(GR),1-NGD*,WLM,WN	0.794	0.493	0.278	0.391	62.9%	83.1%
v	1-NGD,WLM,WN	0.773	0.442	0.302	0.400	68.2%	85.1%
vi	1-NGD,WLM	0.773	0.444	0.302	0.400	68.1%	85.0%
vii	WLM,WN	0.774	0.453	0.315	0.407	71.1%	86.5%

Table 3.8: MLP confusion matrices with related (r) and unrelated (u) counts over four measures (as listed in Table 3.7) on *MixMed*-50. Rows show true counts of related (T) and unrelated (F) pairs. Columns show calculated counts.

	Measure i		Measure ii		Measure iv		Measure v		
	r	u	r	u	r	u	r	u	$\sum$
T	225	180	29	376	243	162	210	195	405
F	92	728	24	796	78	742	75	745	820

Retraining the MLP models over Pak101 and Ped30 paired health terms yielded slightly better results than the equivalent SVM results, although the number of pairs classified

[53] http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html, accessed 27/6/2016.

as related was still underestimated. True and false positive counts ranged from zero to four in all measures tested, including ground truth thresholds $T = 0.4$ and $T = 0.5$ and two different arrangements of hidden internal layers: a (number of attributes + number of classes) $\div$ 2; and b: two layers with three and five nodes respectively.

Google-based measures (ii and iii) classified all terms as unrelated, whilst non-Google-based measures calculated slightly more true positives than false positives. Confusion matrices for a selection of non-Google measures are shown in Table 3.9. Results in [brackets] are for the same measures plus the UMLS::Relatedness measure.

Table 3.9: MLP confusion matrices with related (r) and unrelated (u) counts over three measures on Ped30 and Pak101 term pairs ($T = 0.5$). Results in [brackets] include the UMLS::Relatedness measure.

	Measure i		Measure v		Measure vii		
	r	u	r	u	r	u	$\sum$
T	2 [3]	14 [13]	2 [5]	14 [11]	2 [3]	14 [13]	16
F	1 [3]	114 [113]	2 [2]	113 [111]	1 [4]	114 [111]	115

As with the SVM tests, the threshold for relatedness was lowered to weight ≥ 0.4 and the MLP model retrained. Comparing $T = 0.4$ in Table 3.10 with Table 3.9, it can be seen that including a specialist measure increased the number of pairs classified as related, though this is still underestimated. Best performance was for CRW (measure v) + UMLS::Relatedness, which increased the number of true positives without increasing the number of false positives.

Table 3.10: MLP confusion matrices with related (r) and unrelated (u) counts over three measures on Ped30 and Pak101 term pairs ($T = 0.4$). Results in [brackets] include the UMLS::Relatedness measure.

	Measure i		Measure v		Measure vii		
	r	u	r	u	r	u	$\sum$
T	3 [5]	31 [29]	2 [7]	32 [27]	2 [3]	32 [31]	34
F	1 [5]	96 [92]	3 [2]	94 [95]	1 [3]	96 [94]	97

Classification statistics also showed improvements for CRW plus the specialist measure in comparison with CRW or specialist measures alone. Kappa statistic and accuracy

values were higher and error estimates lower for CRW plus UMLS::Relatedness, as illustrated in Table 3.11.

Table 3.11: Multilayer Perceptron binary classification over Ped30 and Pak101 term pairs.
T is the threshold ground truth weight for related pairs.

	Measure	A	Kappa	MAE	RMSE	RAE	RRSE
	perfect	1.0	1.0	0.0	0.0	0%	0%
v	CRW ($T = 0.5$)	0.878	0.159	0.190	0.321	86.6%	97.8%
	UMLS::R ($T = 0.5$)	0.863	-0.028	0.188	0.315	85.8%	96.1%
v+	CRW + UMLS::R ($T = 0.5$)	0.901	0.389	0.155	0.294	70.4%	89.7%
v	CRW ($T = 0.4$)	0.733	0.013	0.371	0.442	96.0%	100.8%
	UMLS::R ($T = 0.4$)	0.756	0.155	0.352	0.427	90.9%	97.2%
v+	CRW + UMLS::R ($T = 0.4$)	0.779	0.243	0.335	0.421	86.5%	95.9%

For the *MixMed*-50 mixed general and health term set, combinations of Google and Wikipedia measures (iv, v, vi) performed best for both classifiers. This is shown by a higher percentage of correctly classified term pairs and lower error rates and can be visualised by confusion matrices in Tables 3.5 and 3.8: the single WLM measure has a higher rate of false positives, and the single 1-NGD measure overestimates the number of unrelated pairs, particularly in SVM. All three measures predicted classes better than random chance for *MixMed*-50.

The addition of the WN measure did not appear to make a large difference to the result (compare measures v and vi over *MixMed*-50). This is likely due to the low numbers of terms found within WordNet. It is theorised that with a wider set of general vocabulary terms, the impact of this measure would increase.

Performance of individual measures is affected by the match rate of input terms to the textual resource they rely upon. Use of the more restrictive WordNet and simple English Wikipedia resources allowed for resilience testing of a composite measure, demonstrating that it evened out the impact of missing matches in component measures.

Measure iv used the largest number of components, including four Google measures (ln(GR), 1-NGD and 1-NGD filtered for .org and .gov web domains). The quality of its results was slightly higher than the simpler Combined Resources Weight composite

measure (WLM, 1-NGD, and WN). However, as inclusion of Google web domain-filtered results required extraction of more hit counts, measure iv carried a higher processing cost than CRW.

Binary classifiers Support Vector Machine and Multilayer Perceptron both successfully discovered linked terms using CRW over *MixMed*-50, with a small improvement in quality for the MLP classifier. Both returned a high number of false positives, which could reduce the effectiveness of an ontology built upon these weights. However, this problem was lessened by combining individual measures, as seen for measures i and ii versus v Table 3.4. False negatives are less problematic for ontology building, as missing term links can be added using alternative methods.

The binary classifiers were less successful at discovering links between Ped30 and Pak101 health terms using the CRW measure. However, combining CRW with a specialist health measure (UMLS::Relatedness) improved the outcomes in comparison to CRW or the health measure alone. This shows promise for tailoring the relatedness measure for use in specialist domains.

In conclusion, a binary classifier such as a MLP model can discover linked terms for populating a terminological ontology, when trained on a combination of measures from crowdsourced and other public text resources, such as CRW components. They can also be used to estimate a term-relatedness strength between paired terms, as will be discussed in the next section. Additional measures are recommended when processing specialist terminology.

3.3.2 Estimating relatedness

A weighted ontology augments the standard ontology design with link strengths (Section 2.3.2). This strength can be used in search — for example, to select likely related terms for query expansion or to contribute to result ranking.

This section investigates methods for the automated calculation of link weights that could be used in a weighted ontology. All calculated and ground truth weights use a 0.0...1.0 range, where zero is unrelated and one is very strongly related.

A number of machine learning methods were tested for calculation of the relatedness strength, or the probabilistic likelihood of a link, between terms in a resource.

Models were trained on term sets in the same domain as the resource, manually annotated with ground truth relatedness values: the *MixMed*-50 and *MixMed*-100 sets (Appendix A).

Methods and models tested included Principal Component Analysis (PCA), linear and logistic regression, and binary MLP. Linear regression assumes a linear relationship between the ground truth and calculated measures, whilst other methods are less naïve. PCA, for example, calculates a mix of measure components to account for as much variance in the data as possible.

Once trained, models defined the ratio of individual component weights in equations that were validated against ground truth values. Such equations can be applied to other terms within the same knowledge domain. For different contexts, the process would need to be repeated to learn a new, more specific weighted composite measure.

A linear regression model from the Weka machine learning toolkit (Hall *et al.*, 2009) was applied to a test set of terms, with manual judgements of link strength 0.0...1.0 (unrelated...related) used as training data. Similarly, a decision tree regressor from the scikit-learn Python package (Pedregosa *et al.*, 2011) was also constructed. In each case, the model was used to determine relative proportions of the individual components in a composite relatedness weight that best matched the ground truth, judged by relative error rates.

Figure 3.12 plots the CRW measure, trained on the *MixMed*-50 (in blue) and extended *MixMed*-100 (in red) test sets, against ground truth weights. It includes linear regression lines for each. Although the calculated weights are spread over a wide range for each ground truth value, a positive trend can be seen on the plot. This trend is clearer where the model was trained over the larger 100-term set, as illustrated by the narrower blue-shaded 95% confidence range around the regression line.

The CRW measure illustrated in Figure 3.12 is unweighted - that is, each of its components (WLM, WN and 1-NGD) contribute equally to the measure, as described in Section 3.2.2. Training a Weka linear regression model over CRW and *MixMed*-50 ground truth values resulted in the formula:

$$W = (0.64 \times \text{CRW}) + 0.06.$$

Validating weights calculated from this equation against ground truth relatedness with

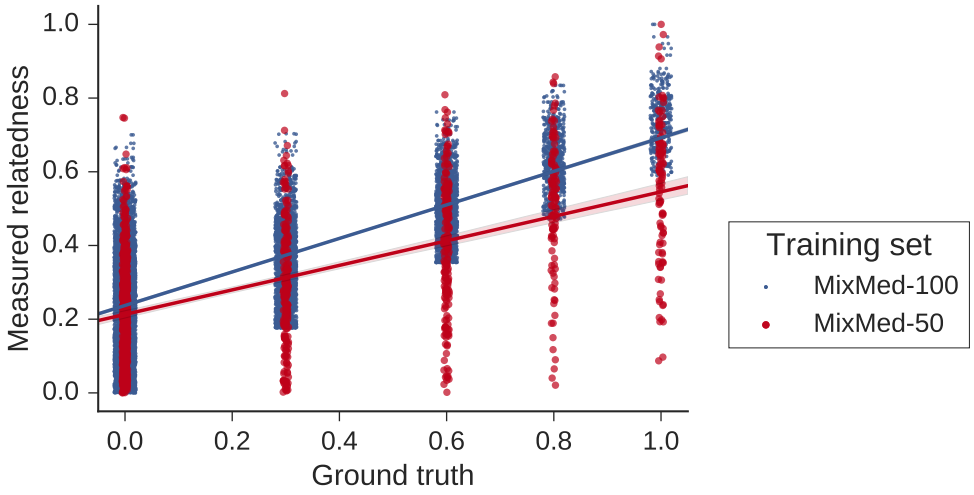


Figure 3.12: Scatterplot of CRW relatedness measure compared against *MixMed*-50 and extended *MixMed*-100 ground truth values ($[0, 0.3, 0.6, 0.8, 1.0]$), including linear regression lines.

10-fold cross-validation gave a correlation coefficient of 0.459 and error rates RAE=86.82% and RRSE=88.8% for *MixMed*-50 terms.

The error rates can be reduced by training the model on CRW components as separate features. Training a Weka linear regression model over the CRW components and *MixMed*-50 resulted in the formula:

$$\text{W} = 0.08 + (0.22 \times \text{1-NGD}) + (0.77 \times \text{WLM}) + (0.36 \times \text{WN})$$

with a correlation coefficient of 0.586 and error rates of RAE=77.05% and RRSE = 80.98%.

The three components of CRW were also used as inputs into a decision tree regressor from scikit-learn, with ground truth weights over *MixMed*-50 used as training data. This built a series of comparisons based on input weight values, as shown in Figure 3.13, to estimate a final relatedness value. For instance, values ($\text{WLM} \leq 0.395$) and ($0.6159 < \text{1-NGD} \leq 0.6159$) resulted in a composite estimate of 0.6833, encompassing 15 out of the 1225 term pairs, with a MSE of 0.0225. The tree produced made no use of the WN input values.

A *logistic* regression model was also trained over the two-class *MixMed*-50 data. This model predicts the probability of class membership, in this case the related class for two terms where the ground truth weight is ≥ 0.5, based on one or more attribute weights. This method provides an alternative relatedness weight between two terms: that is, the probability $(0.0 \ldots 1.0)$ that they are related.

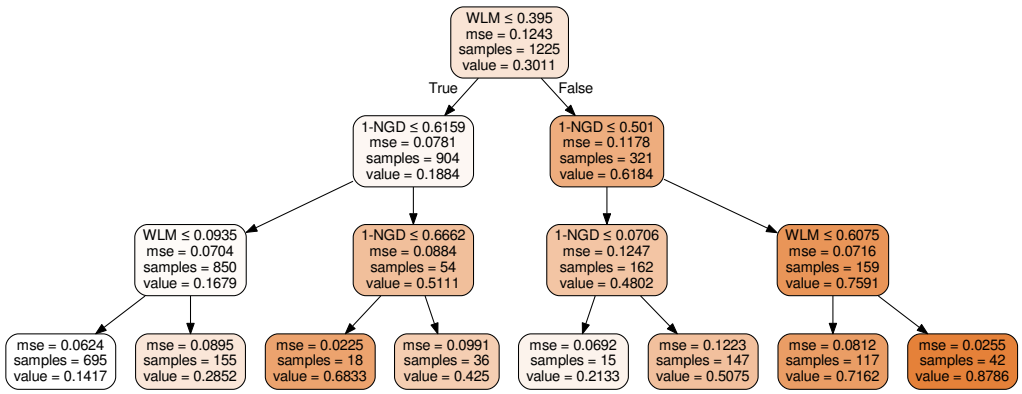


Figure 3.13: Regression decision tree (max depth three) over *MixMed*-50. "mse" is Mean Squared Error.

Coefficients used to calculate the weighted sum of attributes are listed in Table 3.12 for two combination measures (iv and v from Table 3.7). The weighted sum of the measures and an intercept value are used in the logistic regression equation to calculate the probability that a term pair belongs to the related class.

Table 3.12: Logistic regression coefficients of *MixMed*-50 relatedness.

Component	Coefficients	
	Composite *iv*	Composite *v*
ln(GR)	0.498	—
1-NGD	-0.547	1.904
1-NGD-gov	0.847	—
1-NGD-org	-0.323	—
WLM	3.998	4.587
WN	133.400	66.592
Intercept	0.522	-2.405
% correct	78.94%	78.20%
MCC	0.509	0.489

For example, using the component measures of CRW (measure v), the probability p_1 of relatedness between two terms (as determined over *MixMed*-50) was calculated as:

$$W = -2.405 + (1.904 \times \text{1-NGD}) + (4.587 \times \text{WLM}) + (66.592 \times \text{WN})$$

$$\text{Probability that terms are unrelated } p_0 = \frac{1}{1 + e^W}$$

$$\text{Probability that terms are related } p_1 = 1 - p_0$$

A PCA evaluator from the Weka attribute selection library was executed over *MixMed*-50 measures, other than filtered Google measures, using an attribute ranking search method. The algorithm was set to calculate eigenvectors with associated eigenvalues (scalars), until at least 95% of variance in the original data was accounted for. It was trained against the ground truth 0.0...1.0 relatedness weights for each term pair.

Results are shown in Tables 3.13 and 3.14, where *pc* is the principal component number, *var* is the proportion of data variance it accounts for, λ is the eigenvalue and the eigenvector columns show the proportion of a measure within the vector.

Table 3.13: PCA results over *MixMed*-50 with four attributes, trained against ground truth relatedness.

			Eigenvector			
pc	*var*	λ	ln(GR)	1-NGD	WLM	WN
1	0.513	2.051	-0.661	-0.639	-0.384	-0.094
2	0.250	1.000	-0.109	-0.135	0.176	-0.969
3	0.205	0.821	-0.179	-0.322	0.901	-0.228

Using the first component from the four-attribute PCA results outlined in Table 3.13, the weights for each individual measure were used to give an approximate indicator of relatedness W:

$$W = 2.051 - (0.661 \times \ln(\text{GR})) - (0.639 \times \text{1-NGD}) - (0.384 \times \text{WLM}) + (0.094 \times \text{WN})$$

In this case, W accounts for just over half (51.3%) of the variance in the test data. As a result, W can act as an indicative relatedness weight, but should not be used as a representative term-relatedness weight. The index would also need to be scaled to the 0.0...1.0 range, once calculated for all terms within the domain of interest (i.e. beyond the test set used to extract the eigenvalue and eigenvector to use). This presupposes that the test set used is representative of text in the resource to be searched.

Using CRW elements alone, as tabled in Table 3.14, the first eigenvector accounts for less than half (42.6%) of the variability of results.

Table 3.14: PCA results over *MixMed*-50 with three CRW attributes, trained against ground truth relatedness.

			Eigenvector		
pc	*var*	λ	1-NGD	WLM	WN
1	0.426	1.278	-0.666	-0.679	-0.308
2	0.324	0.972	-0.267	-0.169	0.949
3	0.250	0.750	0.697	-0.714	0.069

PCA components can be used as inputs into linear regression. This can be helpful with high-dimensionality data, for example if a large number of specialist term-relatedness measures were included in calculations.

Each method results in different equations for estimating a relatedness weight between terms. For example, when trained over *MixMed*-50 terms using CRW components:

- Linear regression (unweighted CRW).
  $W = (0.64 \times \text{CRW}) + 0.06$
- Linear regression (CRW elements).
  $W = 0.08 + (0.22 \times \text{1-NGD}) + (0.77 \times \text{WLM}) + (0.36 \times \text{WN})$
- Logistic regression.
  $W = -2.405 + (1.904 \times \text{1-NGD}) + (4.587 \times \text{WLM}) + (66.592 \times \text{WN})$
  $P = 1.0 - \left(\frac{1}{1+e^W}\right)$

Results from these three equations are illustrated in Figure 3.14, from which it can be seen that linear regression over the unweighted CRW measure (Lin Reg 1) is less precise than over CRW components (Lin Reg 2). It also tends to underestimate relatedness, whilst the others slightly overestimate it.

P is the probability of relatedness, calculated from logistic regression over CRW components. Its median value is similar to Lin Reg 2, but the interquartile range (green box) is narrower or, in other words, closer to ground truth values.

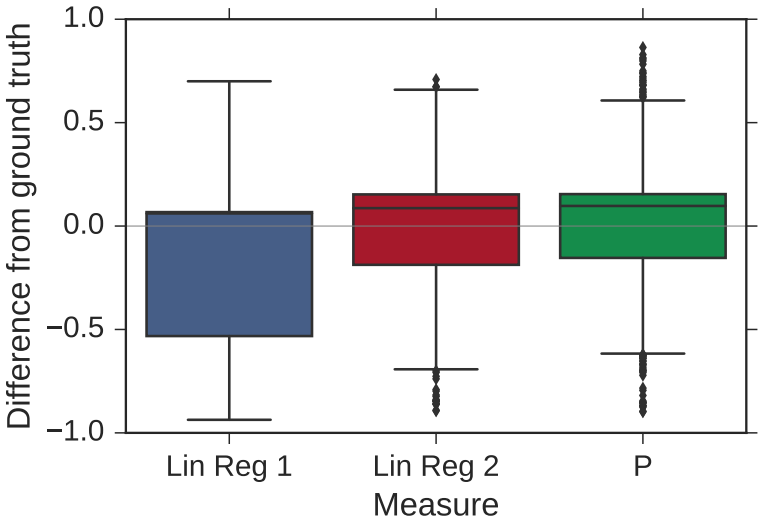


Figure 3.14: Regression results for CRW in comparison with *MixMed*-50. 'Lin Reg 1' is linear regression on unweighted CRW, 'Lin Reg 2' is linear regression on CRW components, and 'P' is probability from logistic regression on CRW components.

Binary classifier models can also be used to estimate term-relatedness probabilities that can be used as link strengths in a weighted ontology. After training a binary MLP classifier from the Python scikit-learn package on *MixMed*-50 values for WLM, 1-NGD and WN (Section 3.3.1), it was used to predict relatedness over *MixMed*-100. The probabilities for the *related* class agreed with the ground truth (on the basis of below 0.5 meaning *unrelated*) in 75% of cases. This result is illustrated in Figure 3.15.

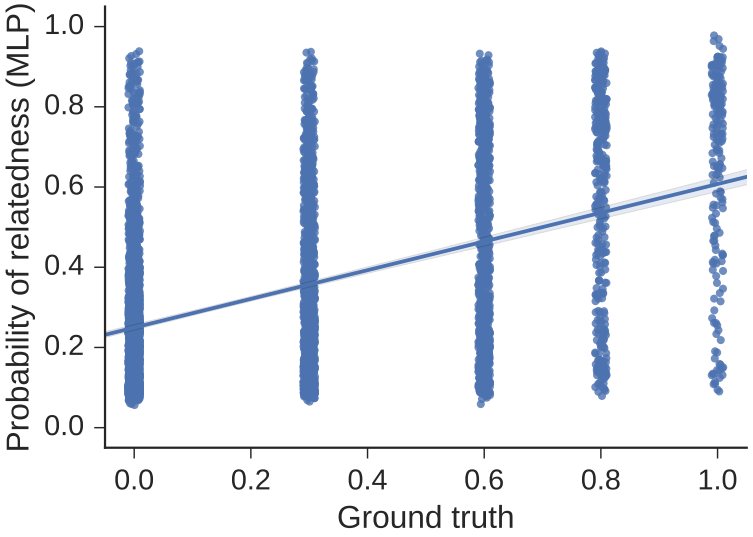


Figure 3.15: Probability of relatedness between *MixMed*-100 terms on a MLP model trained over *MixMed*-50, compared to ground truth weights.

Relatedness probabilities were deemed to be appropriate graph edge weights for a *prob-*

abilistic weighted ontology of terms. A Bayesian approach can then be used to discover new term pairs by multiplying probabilistic weights in a sequence of links. In testing, probabilities were extracted from logistic regression and MLP binary classifier model.

3.4 Categorising and clustering terms

Whilst the stated aim of automatically creating textual ontologies was to detect the existence and strength of links between terms (so that comparative texts could be extended), another useful approach is to group terms into *categories* and *clusters* that share a common concept (labels 6–7 in Figure 3.1)

It was hypothesised that logical groups of terms could be extrapolated from relatedness weights. Machine learning classifiers were trained on predefined categories and validated against manually-assigned ground truth labels. Unsupervised clustering of terms was investigated for detection of novel topics.

Ground truth

Each term in the extended *MixMed*-202 set was manually annotated with a single MeSH descriptor[54] matching its meaning. MeSH descriptors, also called "main headings", include both health-related and generic categories and were designed by expert medical coders to clearly separate out textual descriptions by their main topic:

> "Main Headings should be distinct in meaning (as well as spelling) from other Main Headings in the thesaurus; that is, they should not overlap in meaning." (Nelson *et al.*, 2001)

A subset of MeSH descriptors with associated *MixMed*-50 terms is listed in Table 3.15. MeSH descriptors not allocated within *MixMed*-50 are F: Psychiatry and psychology, I: Anthropology, education, sociology and social phenomena, L: Information science, M: Persons, V: Publication characteristics, and Z: Geographic locations.

To test multiclass classification of paired terms, a number of category labels were allocated to each ground truth term pair. Categories were of two broad types: ordinal

[54] https://www.nlm.nih.gov/mesh/, accessed 19/6/2016.

Table 3.15: *MixMed*-50 terms, classified by MeSH descriptors.

MeSH Descriptor		*MixMed*-50 terms	
A	Anatomy	3	anus, cytokine, lymphocyte
B	Organisms	11	animal, bacteria, bee, *C. elegans*, cattle, dragonfly, hummingbird, mistletoe, poodle, virus, yeast
C	Diseases	18	arthritis, asthma, Black Death, cancer, cholera, diabetes, disease, fever, flu, genetic disorder, haemophilia, HIV, illness, infection, infectious disease, malaria, obesity, STD
D	Chemicals and drugs	1	caffeine
E	Analytical, Diagnostic & Therapeutic Techniques and Equipment	2	hand washing, vaccination
G	Biological sciences	2	enzyme, immunity
H	Physical sciences	4	evaporation, mineral, stream, water cycle
J	Food, drink and technology	4	cheese, chopsticks, plow, waffle
K	Humanities	1	Tutankhamun
N	Health care	2	health, pediatrics
—	(noise)	2	billion, masturbation

grades of relatedness strength and logical descriptors of topics.

- two ordinal grades of relatedness (binary), as used in Section 3.3: *related* ($\geq T$) and *unrelated* ($< T$), where T is a ground truth relatedness weight.
  $T = 0.5$ except where specified otherwise.

- five ordinal grades of relatedness: *not* (<0.3), *slightly* [0.3...0.5), *somewhat* [0.5...0.7), *moderately* [0.7...0.9), and *highly* ($W \geq$0.9) related.

- four ordinal grades of relatedness: *not* (<0.3), *slightly* [0.3...0.5), *somewhat* [0.5...0.7), and *related* ($\geq$0.7).

- sixteen nominal term definition categories: top level MeSH descriptors ['A', 'B', ..., 'N', 'V', 'Z'].

Multiclass Classifiers

Classifiers were trained on ground truth category labels and term-relatedness weights. The classifiers tested were decision trees and the multiclass MLP neural network classifier.

Firstly, decision trees were constructed to investigate the relative importance of specific measures in classifying the strength of relationship between two terms. The purpose of building these decision trees was to gain an initial indication of the relative importance of different measures, rather than as a final measure to use in practice. This is because decision trees can overfit to the input measures, which results in a classifier unsuitable for generalising to a larger vocabulary.

Two decision tree types were trained on paired terms from the *MixMed*-50 set of terms:

- the DecisionTreeClassifier class from scikit-learn[55], and
- the J48 decision tree classifier from Weka[56].

All records in a leaf node belong to the same class. Resulting trees were investigated for how often individual measures were tested, and how early in the classification process they were used. The J48 tree was also validated against the ground truth relatedness categories using 10-fold cross validation.

Decision trees created over *MixMed*-50 data using scikit-learn without a maximum depth limit followed between three and 31 tests before reaching a leaf node of a single link category (either binary or five grades of relatedness). The number of comparisons, broken up into the component measures used to build each tree, is listed in Table 3.16.

Table 3.16: Number of tests in scikit-learn decision trees over *MixMed*-50 relatedness.

Classes	Total tests	(WLM)	(WN)	(1-NGD)	(GR)	(GI)
2	323	78	0	245	—	—
5	642	123	1	518	—	—
5	664	272	1	119	139	133

Constructing a decision tree with the three CRW components for five-graded categories (the highlighted row in Table 3.16) used the WN measure only once. More than half the comparisons in all *MixMed*-50 cases tabled were based upon Google as a resource.

Different combinations of measures were also used to construct trees with a Weka J48

[55] http://scikit-learn.org/stable/modules/tree.html, accessed 5/7/2016.

[56] http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html, accessed 5/7/2016.

decision tree classifier. Validation results for a tree trained with six input measures are tabled in Table 3.17. The trees produced made little or no use of the WN measure — of the trees that did use WN, only 5-9% of internal tree nodes tested it and these were at a tree depth of eight or more

Table 3.17: Weka J48 decision tree validation results over *MixMed*-50 relatedness. 1-NGD* = 3× 1-NGD: one for each of all, gov and org domains.

Component measures	classes	A	Kappa	MAE	RMSE	RAE	RRSE
perfect	—	1.0	1.0	0.0	0.0	0%	0%
ln(GR),1-NGD*, WLM,WN	5	0.721	0.583	0.106	0.289	38.8%	78.3%

From these two sources, it was concluded that WN contributed little to categorisation into ordinal grades of relatedness. This reinforces results from Section 3.2.2, where it was concluded that WordNet had too few matches over *MixMed*-50 to have a large impact but could have a greater effect on a larger set of general-purpose terms.

Weka multiclass MLP models were trained over ARFF files of paired terms from *MixMed*-50. The input ARFF file was modified to include labels for the five grades of relatedness, four grades (merging the top two 'moderately' and 'highly' categories to approximate the size of the 'slightly' and 'somewhat' categories), and MeSH categories.

For MeSH categories, each row was labelled to match the MeSH categories of both terms in the pair. MeSH categories could therefore either be single, e.g. 'A', or a pair e.g. 'A_C' or '-_H' where a dash represents an uncategorised term. This resulted in 64 unique categories for *MixMed*-50, with this larger number of categories reducing accuracy and increasing model training and testing times.

The Weka implementation of the multiclass MLP model was trained over *MixMed*-50 data using 500 training epochs, momentum of 0.2, and a learning rate of 0.3. The model used two hidden layers with three and five hidden nodes, respectively. Results were validated using 10-fold cross validation, as recorded in Table 3.18 for a variety of measure combinations.

Table 3.18: Multiclass Multilayer Perceptron results over *MixMed*-50 relatedness. 1-NGD* = 3× 1-NGD: one for each of all, gov and org domains.

Component measures	classes	A	Kappa	MAE	RMSE	RAE	RRSE
perfect	—	1.0	1.0	0.0	0.0	0%	0%
GR,NGD,WLM,WN	5	0.548	0.233	0.237	0.348	87.2%	94.4%
GR,NGD,WLM	5	0.549	0.235	0.238	0.348	87.3%	94.5%
WLM	5	0.529	0.204	0.240	0.349	88.1%	94.7%
ln(GR),1-NGD*,WLM,WN	5	0.558	0.269	0.229	0.334	84.2%	93.4%
ln(GR),1-NGD*	5	0.520	0.177	0.243	0.354	89.4%	95.9%
1-NGD*	5	0.513	0.095	0.254	0.361	93.4%	97.8%
1-NGD,WLM,WN	5	0.547	0.231	0.238	0.348	87.3%	94.4%
ln(GR),1-NGD*,WLM,WN	4	0.589	0.307	0.268	0.373	80.5%	91.5%
ln(GR),WLM,WN	4	0.593	0.311	0.276	0.375	83.0%	92.1%
1-NGD*	4	0.592	0.313	0.268	0.374	80.6%	91.6%
1-NGD,WLM,WN	4	0.589	0.287	0.279	0.378	83.9%	92.7%
ln(GR),1-NGD*,WLM,WN	64	0.222	0.087	0.029	0.120	96.9%	98.8%
1-NGD,WLM,WN	64	0.225	0.088	0.029	0.120	97.2%	98.7%

MLP results for ordinal categories of relatedness (four or five) were above 50% accuracy ($A > 0.5$), showing that relatedness multi-classification is feasible. It could be useful for indicating low-relatedness pairs that could be ignored, for outlier detection. However, given the uncertain nature of boundaries between relatedness grades and the flexibility of natural language, the binary MLP classifier was judged to be a more appropriate machine learning model.

Classification into MeSH categories resulted in a low percentage of correctly classified pairs, as seen in Table 3.18 (64 classes). The large number of category combinations, some containing very few items, rendered this type of classification impractical for the 1225 term pairs tested. Low numbers of unique terms in the training set meant that the potential for outlier detection (such as paired terms assigned to a 'noise' category) or reinforcing links (such as paired terms assigned to the same MeSH descriptor) were unverified by the classifier models tested.

Probabilities of class membership were also extracted by training a MLP model from the

Python scikitlearn.neural_network library on *MixMed*-50 data with CRW elements. For five ordinal grades of relatedness, the trained scikit MLP model to predict classes over the training dataset yielded 684 out of 1225 correct classifications (accuracy=0.558), and for four ordinal grades (merging the top two layers), 735 (accuracy=0.600), demonstrating small improvements over the Weka model (see grey rows in Table 3.18).

Each term pair is assigned a probability of membership for each possible category. As these always sum to one for each pair, this is a useful indication of the confidence in categorisation of term pairs. For example, the mean maximum probability per row over four ordinal grades of relatedness was 0.598, as illustrated in Figure 3.16.

Individual paired term probabilities from ordinal grades can be used to reinforce weights in a weighted ontology, similarly to binary MLP link probabilities (Section 3.3.2), although the binary probabilities are a closer match in this regard.

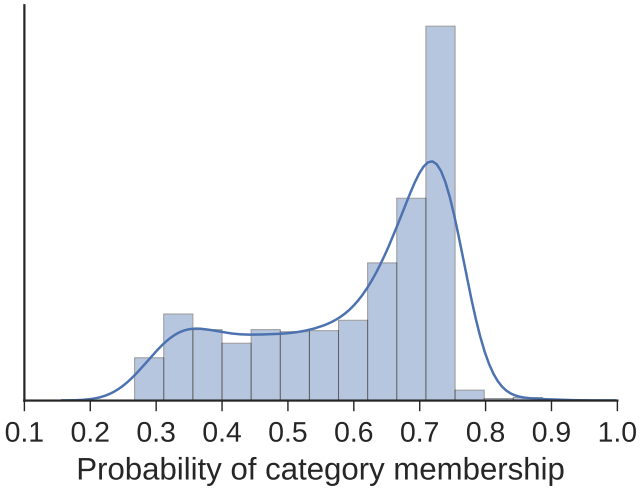


Figure 3.16: Maximum category probability per term pair from a MLP model trained over *MixMed*-50 with four ordinal grades of relatedness.

Clustering terms

A disadvantage of categorisation for finding topics from term-relatedness weights is that terms are considered in pairs rather than individually. Also, the use of supervised classification methods means that the available categories are predefined. Clustering addresses both of these issues: it uses edge weights to cluster individual terms so that they can be considered en masse rather than as separate pairs, and it is unsupervised and hence has the potential to find novel topics.

The Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm was applied to a graph of terms with relatedness measure edge weights. Testing was undertaken with minimal cluster sizes of 3, 5, 10, and 15, and ϵ values of 0.15, 0.35, 0.4 or as determined by the algorithm implementation.

Each relatedness measure was defined as a weighted graph describing a single dimension (feature) for clustering. Combinations of measures were tested, keeping in mind that clustering algorithms are most useful with dimensions higher than two (Jain, 2010).

Graphs were implemented as square matrices of size n x n, where n is the number of unique terms. A cell at row A and column B holds a relatedness value between terms A and B. Although the algorithm can handle sparse matrices, there were no gaps in the term data tested. Weights are symmetric (i.e. for any two terms A and B, weights $W_{A,B} = W_{B,A}$), with all diagonals equal to one.

A small extract for the 1-NGD measure is illustrated as a weighted graph (Figure 3.17) and associated matrix (Table 3.19). The curved, dotted line in Figure 3.17 shows low-weighted graph edges that could be cut, giving two clusters: [‘diabetes’, ‘vaccination’, and ‘flu’] and [‘poodle’].

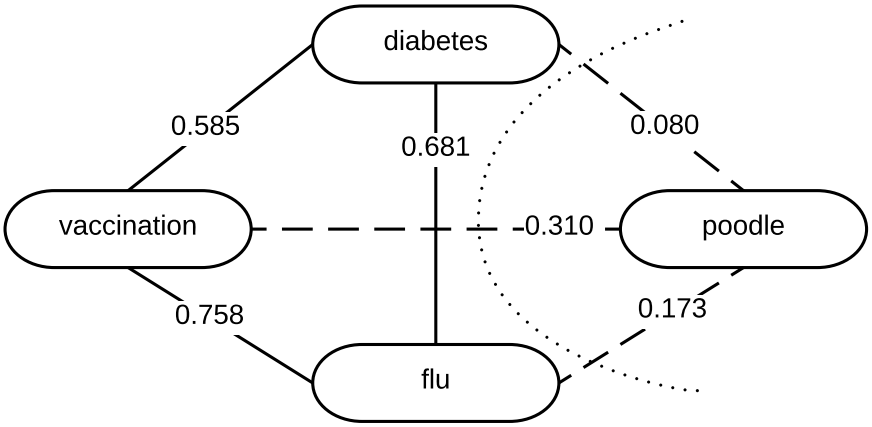


Figure 3.17: Extract of 1-NGD weighted graph over *MixMed*-50. The dotted line shows a potential graph cut, across edges with lower weights (shown by dashed lines).

Table 3.19: Extract of 1-NGD weights matrix over *MixMed*-50.

	diabetes	vaccination	flu	poodle
diabetes	1.000	0.585	0.681	0.080
vaccination	0.585	1.000	0.758	0.310
flu	0.681	0.758	1.000	0.173
poodle	0.080	0.310	0.173	1.000

An implementation of DBSCAN from the Python scikit-learn.cluster package (Pedregosa *et al.*, 2011) was applied to the *MixMed*-50 and extended *MixMed*-202 datasets. A MATLAB implementation of a DBSCAN variant that can calculate an optimal epsilon parameter from the data (Daszykowski *et al.*, 2001) was also tested.

For each test, a distance matrix D was precomputed from the input weight matrices of each individual relatedness measure. For single measures, distance was calculated from the weights matrix W as $D = 1.0 - W$. For composite measures, multiple weight matrices were combined into a single distance matrix using 1.0 - L_2 Frobenius Norm, scaled to $0.0 \ldots 1.0$. All diagonals were set to 0.0.

A result summary was produced after each test, listing:

- numbers of estimated clusters and outliers,
- an extract of terms from each cluster,
- an extract from the list of outliers (terms not allocated to a cluster), and
- metrics for evaluating clustering performance.

After the clustering model was fitted to relatedness measures between terms, contents of the calculated clusters were considered for logical similarity of definitions and compared to manually assigned categories: specifically, MeSH descriptors (Table 3.15) over the *MixMed* set of terms.

Results for the scikit-learn DBSCAN algorithm over *MixMed*-50 and extended *MixMed*-202 terms, using measures including precalculated CRW, are summarised in Table 3.20 where n is the number of terms being clustered, m is the minimum number of terms per cluster, ϵ is the epsilon parameter value, C is the number of clusters and O the number

of outliers. Clustering statistical measures homogeneity (Ho), completeness (Co), V-measure (Vm), adjusted Rand Index (RI), AMI, and Silhouette Coefficient score (SC) are also listed (each described in Section 2.6.2).

Table 3.20: DBSCAN clustering results over *MixMed*-50 and *MixMed*-202 relatedness. n = number of terms, m = minimal cluster size, C = number of clusters, O = number of outliers.

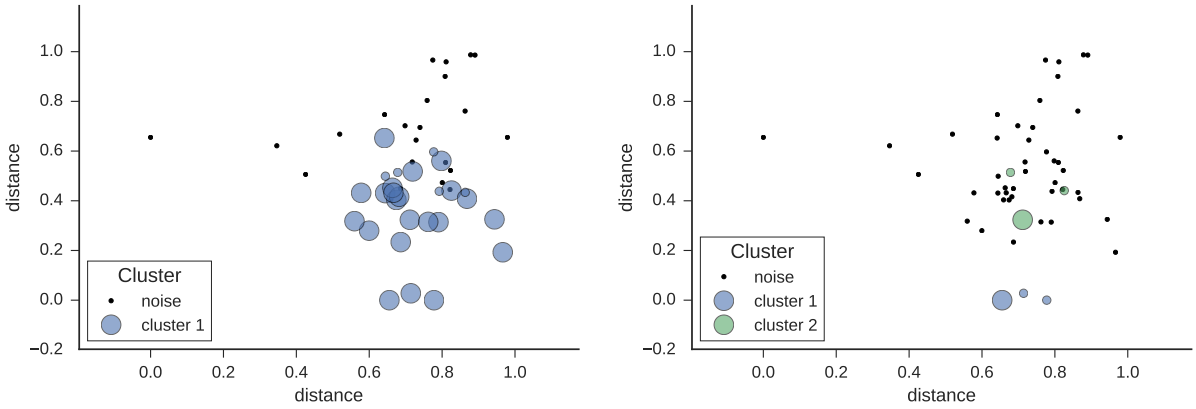
Measure	n	m	ϵ	C	O	Ho	Co	Vm	RI	AMI	SC
perfect:	—	—	—	—	—	1	1	1	1	1	1
i. WLM	202	3	0.35	3	39	0.097	0.29	0.145	0.049	0.061	-0.082
ii. 1-NGD	202	3	0.35	1	1	0.007	0.502	0.014	0.002	0.002	0.169
v. CRW	202	3	0.35	10	26	0.065	0.206	0.099	-0.001	0.001	-0.115
v. CRW	202	3	0.15	2	192	0.028	0.268	0.05	-0.021	0.006	-0.032
v. CRW	50	3	0.35	1	23	0.181	0.511	0.267	0.185	0.122	0.232
v. CRW	50	3	0.15	2	44	0.073	0.316	0.119	-0.052	-0.002	-0.017
v. CRW	50	5	0.35	1	25	0.177	0.498	0.262	0.185	0.117	0.249
v. CRW	50	5	0.4	1	19	0.239	0.699	0.356	0.252	0.184	0.211
v. CRW	50	10	0.4	1	21	0.269	0.770	0.399	0.286	0.216	0.244

Clusters calculated with a small minimum cluster size were generally independent of the MeSH descriptor categories, as shown by near-zero values for Ho, Vm, RI, AMI and, to a lesser degree, Co evaluation metrics in Table 3.20. The SC score, which is calculated separately to known categories, was also close to zero. This shows a lack of logical coherence within the clusters that goes beyond mismatches to MeSH descriptor categories.

The clusters produced tended to either contain many, mostly medically-related terms (across different MeSH medical themes), or contain very few terms. Examples of small clusters include single-term clusters and logically-consistent clusters such as: [‘yeast’, ‘waffle’, ‘pizza’, ‘dough’, ‘pita’] (WLM only, $\epsilon = 0.35$, $m = 3$, over 202 terms) or [‘virology’, ‘pathogen’, ‘carcinogen’] (multi-domain 1-NGD, $\epsilon = 0.35$, $m = 3$, over 202 terms).

Different epsilon values had a large impact on results as seen in Figure 3.18, comparing $\epsilon = 0.35$ and $\epsilon = 0.15$ for clustering *MixMed*-50 terms with the CRW measure where the minimum terms per cluster $m = 3$. At $\epsilon = 0.35$, a single cluster of 27 terms and 23 outliers were discovered. The 27 clustered terms fell into MeSH categories A, C and N (anatomy, diseases, health care), including [‘asthma’, ‘cancer’, ‘disease’, ‘enzyme’, ‘pediatrics’]. This cluster also contained the terms [‘bacteria’, ‘virus’, ‘yeast’], all of which can be considered vectors of ill health. At $\epsilon = 0.15$, two clusters of three terms each were

described: [‘disease’, ‘illness’, ‘genetic disorder’] and [‘infectious disease’, ‘vaccination’, ‘Cholera’], with all other terms labelled as outliers.



(i) Epsilon $\epsilon = 0.35$.

(ii) Epsilon $\epsilon = 0.15$.

Figure 3.18: *MixMed*-50 set clustering with CRW, using DBSCAN (scikit-learn implementation). Minimum cluster size = 3.

An implementation (in MATLAB) of an extended DBSCAN algorithm that automatically estimates an optimal value of epsilon was also executed over *MixMed*-50, using the CRW measure. Each run discovered a single cluster with outliers (Table 3.21). Although some non-medical terms were included in the cluster (such as ‘chopsticks’ and ‘Tutankhamun’ where $m = 15$), most of the cluster consisted of health-related terms. This demonstrates an application of clustering techniques for outlier detection amongst a set of terms.

Table 3.21: DBSCAN results over *MixMed*-50 using automatically estimated epsilon ϵ. n = number of terms, m = minimal cluster size, C = number of clusters, O = number of outliers.

Measure	n	m	ϵ	C	O	Outlier terms
v. CRW	50	5	1.6939	1	12	bee, caffeine, cheese, dragonfly, hummingbird, masturbation, mistletoe, plow, poodle, stream, waffle, yeast
v. CRW	50	10	1.7175	1	10	bee, caffeine, cheese, dragonfly, hummingbird, masturbation, mistletoe, poodle, stream, waffle
v. CRW	50	15	1.7315	1	9	bee, caffeine, dragonfly, hummingbird, masturbation, mistletoe, poodle, stream, waffle

A common outcome over *MixMed*-50, particularly with larger values of the minimum cluster size m, was detection of a large cluster consisting mostly of health-related terms plus a number of outliers. The silhouette coefficient in these cases was positive and above zero so that, although the values were low, it was concluded that the clusters

were sufficiently coherent, given the imprecise nature of natural language.

The other clustering measures compared the estimated clusters to MeSH descriptor categories. These also showed signs of some alignment between estimated and MeSH groupings when m had a higher value. This could be a reflection of estimated clusters that encapsulate a number of MeSH categories, such as A, C and N (anatomy, diseases, health care).

Examination of the detected clusters showed that, as expected, most terms within a cluster alluded to a single logical theme such as health or baked goods. As clusters were detected based upon relatedness weights from general terminology resources, this helps to validate the use of relatedness weights to detect ontological links between terms.

It was concluded that relatedness weights calculated over general, crowdsourced resources can be used to predict logical groupings and outliers in a set of terms, albeit with a level of error where some terms may not be allocated to the most appropriate cluster. Based on previous research into clustering and natural language, such as Liu & Croft (2004), it is presumed that more clusters would be found as the set of terms being processed become larger and more diverse. This would facilitate incorporation of detected clusters into a hierarchical taxonomy of linked terms and/or to reinforce existing weights calculated between terms in the same cluster.

The detection of outliers was itself a useful outcome. For example, it served to separate out '*poodle*' from '*diabetes*', where the former was linked to from the *Wikipedia* page on diabetes. Once detected, outliers can be used to remove spurious term links or reduce existing weights in an ontology to improve its reliability.

Clustering algorithms, once trained, can be applied to new data. For example, a DBSCAN model trained on the current contents of a resource can be applied to new data records, articles, and so on as they are added. However, a model trained on one knowledge domain is likely to be less reliable in another. For best results, a new clustering model would need to be trained for a new resource. This could be included within the preprocessing stage of adding new data to be searched.

3.5 Summary and discussion

In this chapter, automated methods for estimating term-relatedness weights were tested against human judgements, and examined for relevance in the production of regular and weighted ontologies of terms.

The unweighted Combined Resources Weight (CRW) measure, derived from multiple crowd-sourced and other text resources, yielded linked terms that were deemed suitable for inclusion in an ontology of terms, particularly for general terminology. CRW values tested *MixMed*-50 terms were closer to ground truth weights than for the individual measures.

In a specialist domain (health), CRW was less reliable for discovering term links but its performance was improved by incorporating a specialist measure. This reinforced the findings by Garla & Brandt (2012) who showed that semantic similarity measures in the biomedical domain were more reliable when calculated from a combination of textual sources.

A composite measure can be adapted to a data resource with the use of machine learning models. Binary classifiers were useful for detecting related term pairs that can be used to build an ontology of terms, with MLP performance showing a small improvement over SVM. Probabilities of relatedness can also be extracted with additional processing over these models. Clustering and multiclass categorisers algorithms did not yield useful term links, but showed promise for refining existing links.

All calculated weights were in the 0.0...1.0 range. However, probabilistic weights in particular were useful for defining weighted ontology link weights, as a Bayesian approach allows for multiplication of weights in consecutive links, through which new term relationships can be discovered. Probabilistic weights were successfully extracted from logistic regression and binary MLP models.

Retaining link weights in a weighted ontology of terms allows for non-deterministic, ranked responses to a query. As each ontology is tailored to terms in a data source, a single query can be augmented to better match contexts of multiple resources.

Models such as MLP, once trained, can be applied to new terms in a resource as it expands. However, these classifiers were trained on manually-labelled ground truth

terms, imposing an extra cost in resources. Also, models would become less accurate over time, unless the terminology used within a data resource was fairly static or the classifiers retrained as the resource was updated.

A potential drawback of combined measures was the cost of extracting individual weights. For CRW, this included the cost of updating Wikipedia dump files and delays caused by multiple Google accesses. The first cost can be mitigated by replacing WLM with alternative Wikipedia-based resources and measures such as DBpedia (Lehmann *et al.*, 2015) or reusing older Wikipedia versions where terminology used in new data records remains stable.

In this chapter, existing specialist term-relatedness measures for biomedical terminology were tested. However, as not all specialist vocabularies are as well-defined, a useful additional resource would be methods to construct specialist measures from known textual sources, for example by adapting the current Google and Wikipedia measures.

Recommendations are, where ground truth term-relatedness weights can be sourced, to train a MLP binary classifier and extract relationship probabilities for resource terms to add to a probabilistic weighted ontology. Where it is not practical to collect ground truth, the CRW measure can be used as an alternative source of weights. For resources containing a high proportion of jargon, specialist measures should be incorporated into the MLP model or combined with CRW elements. Finally, the resulting weighted ontology can be refined by checking for outliers via clustering or multiclass categorisation. Further, semantic relationships such as `"is-related-to"` can be built into an unweighted ontology of terms by extracting links above a threshold weight.

In conclusion, a weighted terminological ontology can be automatically built for a data resource, using crowdsourced and other textual resources. This can be used to augment queries to adapt context from the user to that of the resource being searched. Conversely, it can be used to expand metadata available for a resource.

The next step is to consider the search process itself: finding results within the resource that are a match to the initial question. In the next chapter, the semantic content of data records at a macro, rather than micro, level will be investigated. In other words, searching a data resource by considering entire documents rather than individual terms, where a document can be any type of content that contains text, such as an article, database record, or webpage.

4 Matching texts

Figure 4.1 illustrates the layout of this chapter. Section 4.1 introduces the problem under consideration and the approaches taken. Section 4.2 outlines Natural language processing (NLP) models used for document searches (label 1). Section 4.3 focusses on the production and use of *virtual queries* from an input request (label 2) and Section 4.4 discusses topic detection (label 3). Results are summarised and discussed in Section 4.5.

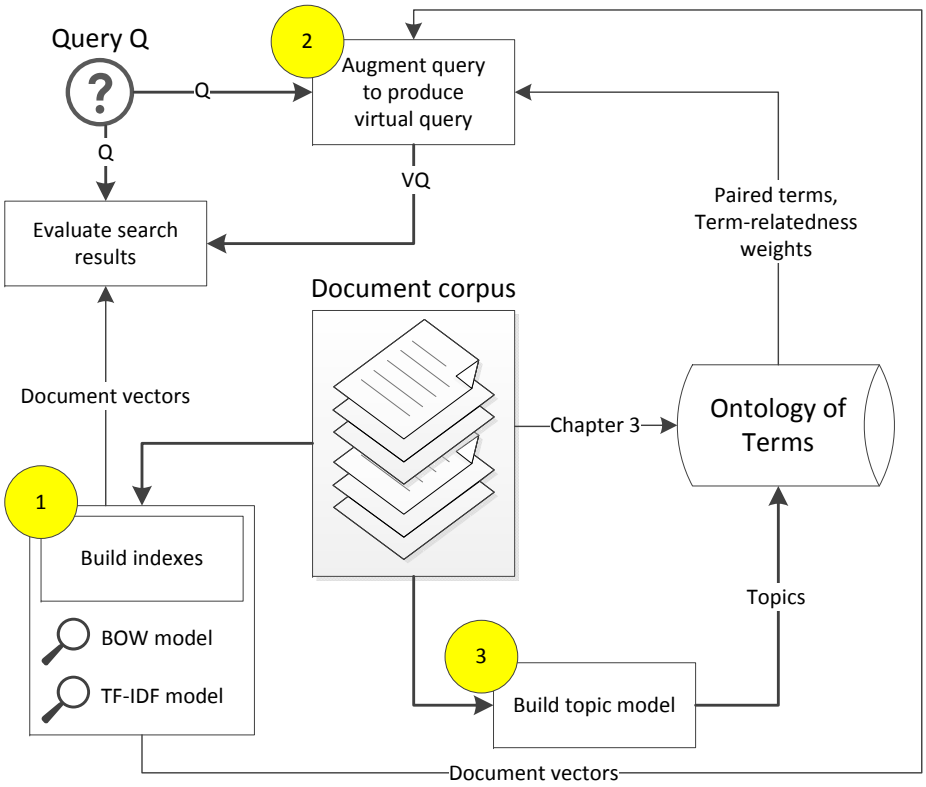


Figure 4.1: Summary of Chapter 4. The yellow circular labels are explained in text.

4.1 Introduction

Keyword searches are useful for matching terms between two documents, such as a query or target document and data record text. However, the flexibility of natural languages means that keyword searches can miss useful matches, particularly between documents written from different perspectives.

This chapter incorporates the textual context of a data resource into a search on that

data. It uses NLP techniques to preprocess resource documents and augments queries with reference to data-specific terminologies.

In this thesis, a document is any item containing text, such as an article, web page, descriptive metadata, database record, spatial feature, or text query. A data resource can be considered a collection — a *corpus* — of documents. Text contained in data records contribute to the context of the data source, including its vocabulary, relevant toponyms, and themes.

Virtual queries were built by augmenting queries with related terms, using local term resources (Chapter 3). Similarity indexes based on different NLP models were then tested for both unaltered and virtual queries.

Topic modelling was also explored for its potential as a term extractor, to assist in building or enhancing a resource-specific weighted ontology of terms.

4.2 Document matching

Given that a contextual search needs to consider both user and data, the first step in information retrieval (IR) is to translate between these contexts.

One component of textual context for a data resource is a similarity index of its documents, which can be produced by preprocessing its contents using NLP techniques. Such an index can improve the efficiency of comparisons between an external text (such as a query or exemplar document) and a resource.

NLP bag-of-words (BOW) and term frequency-inverse document frequency (TF-IDF) models (Section 2.2.2) were used to produce indexes for matching short queries to corpus documents. Methods were tested on corpora of crowdsourced Wikipedia articles.

4.2.1 Indexing

Test corpora were created using the text content of English-language Wikipedia articles as documents. All formatting tags and URL addresses were removed, as was the reference list, where present. Although images were removed, any captions were retained. Navigation labels were also included, as plain text without forward link information.

Textual information from Wikipedia constructs such as category list, infobox, or other template was also kept.

Two separate corpora and associated dictionaries were built from Wikipedia articles:

1. **MixMed-50**: 50 articles (48 unique) with titles matching the *MixMed*-50 (Section 3.2.1) set, and

2. **WHO-105**: 105 health-related articles from terms in World Health Organisation (WHO)[57] health topics web pages.

The *MixMed*-50 corpus was created from English Wikipedia (1 October 2015) articles whose titles matched a term in the *MixMed*-50 set. This resulted in 48 unique documents because two of the original 50 terms redirected to another term in the set: 'infection' ↔ 'infectious disease' and 'illness' ↔ 'disease'. In essence, 'illness' and 'disease' were treated as synonymous in spite of subtle differences between the terms (disease normally describing a condition of the body, and illness how a person feels). Because these terms are often used interchangeably, a record that is deemed relevant for one term in the pair is likely to be relevant for the other.

The WHO-105 corpus contains articles from the English Wikipedia whose titles match health-related terms derived from WHO information pages. All articles were collected on 10 April 2015, except for two that were collected on 15 April 2015: ['cleft lip and palate', 'club foot']. The 105 titles are listed in Appendix B. They can be loosely grouped into logical categories including anatomical terms ['blood', 'ear', 'eye', ...]), medical care ['symptom', 'therapy', 'vaccine', ...], and types of medical conditions such as infectious diseases, chronic conditions and genetic disorders.

Each test corpus was processed to create a similarity index between its documents, based on the tokens contained in each. First, all document text was converted to lower case and punctuation, digits, and single-character words were removed. Any accent marks such as acutes (´) or umlauts (¨) were also removed. Stop words as defined by the Python nltk library were dropped, as were tokens that only appeared once in the corpus. Months of the year and personal names, as reported by the USA Social Security Administration for the years 1880, 1940, and 2013[58] were also removed. This resulted

[57] http://www.who.int/topics/en/, accessed 11/8/2016
[58] http://www.ssa.gov/oact/babynames/limits.html, accessed 8/6/2015

in the loss of some general terms that are also names, for example ‘Rose’, ‘Grace’, ‘Sky’, ‘Sunny’, ‘Red’, and ‘Wolf’.

For each corpus, a dictionary of unigram tokens and BOW vectors was created over the source texts, using the Python gensim library (Řehůřek & Sojka, 2010). Gensim TfidfModel objects were initialised to convert each corpus into TF-IDF-model term vectors. A similarity index across articles was created for the BOW and TF-IDF models of each corpus, using instances of the gensim.similarities.Similarity class.

Efficacy of the similarity indexes was confirmed by comparing document similarity in the *MixMed*-50 corpus to the ground truth relatedness values between titles (Section 3.2.1). As expected, document similarity within the *MixMed*-50 corpus followed the same trend as calculated term-relatedness. However, the narrower definition of similarity compared to relatedness resulted in lower weights: compare Figure 4.2, which plots cosine similarity of *MixMed*-50 articles against the term-relatedness of their titles, to the *MixMed*-50 Combined Resources Weight (CRW) measures in Figure 3.12.

Note that the BOW model resulted in a greater overlap of similarity weights at each of the ground truth values [0, 0.3, 0.6, 0.8, 0.9]. The range of similarity values is also greater in all cases except for the outlier similarity of one in both results. From these results, it was concluded that the BOW model resulted in a lower precision of results relative to the TF-IDF model. This was expected, given that BOW models do not adjust for the frequency of terms within the corpus.

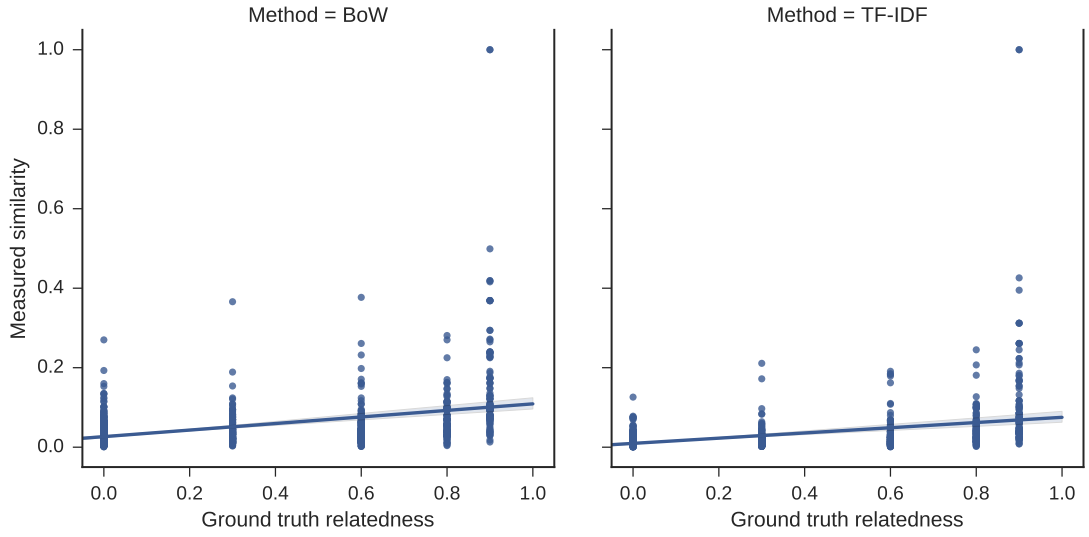


Figure 4.2: Cosine similarity of paired *MixMed*-50 corpus articles versus ground truth relatedness of their titles.

4.3 Virtual queries

When searching for similar results to an example document, the exemplar itself contains context. This provides scope for NLP methods to match it against the document similarity index of a data source. However, the volume of textual context is limited in short queries such as `"clinics near Perth"`. Some resources also contain short documents: consider a data record description of `"Northbridge Medical Centre"` that has no keywords in common with the previous query but is a logical match in Western Australia.

This section describes the automatic production of *virtual queries* from short queries. Virtual queries are augmented with related terms from an ontology derived from the context of the data resource. Together, these serve to contextualise a short query relative to one or more searchable resources.

For each article in the WHO-105 corpus, different types of query were produced to target each article. The tests consider only the primary target, ignoring articles that are somewhat related such as 'injury'-'accident' or 'psychiatry'-'psychology'. A successful IR operation will rank the target article highly, ideally first, in the set of results.

In the initial tests, similarity was based on a BOW model constructed as described in the previous section, and any additional terms were selected based on unigram tokens. The test queries produced were:

- the title of the target article,
- the most common terms within the target article (terms with the three highest TF values), excluding stop words like 'was' and 'the',
- title terms plus their immediate synonyms, taken from WordNet, and
- virtual queries: the title augmented with weighted ontology tokens that link to title term with weights above a threshold $T = [0.3, 0.4, 0.5, 0.6]$.

Virtual queries were built using a weighted ontology, constructed from extended *MixMed*-202 terms with CRW term-relatedness weights (Chapter 3.3). First, the initial text (the article title) was split into unigram tokens that were added to a new virtual query Q_V. Then, for each original query term t_Q, a set of tokens $[t_V]$ in the weighted ontology were

added to Q_V, where term-relatedness weight ($t_q \leftrightarrow t_v$) were at or above a threshold T. The $[t_V]$ set could equally be taken from unweighted ontology triples, where a link exists from t_Q to t_V.

Each virtual query Q_V was constructed as a BOW vector of all t_q and related t_v terms. A term can be repeated within a single Q_V, giving it a greater TF and potentially more influence on the outcome. For example, in a query of `"flu vaccine"`, the term 'vaccination' has a CRW weight above $T = 0.5$ to both 'flu' and 'vaccine' and hence it would be added twice to an augmented BOW vector.

For each query, a result set R was extracted, containing $|R|$ articles. The size $|Q|$ of the query is the number of tokens it contains. The set of potentially useful, relevant articles is U, with size $|U|$=1 in this case as there is a single target article. The maximum search depth (number of items to return) is k.

A summary of the results by query type is shown in Table 4.1 where n is the number of queries, $\overline{|Q|}$ is the mean number of tokens per query, and $\overline{|R|}$, $\overline{r}$, $\overline{RR}$, and $\overline{P@10}$ are mean result set size, mean rank of the target, reciprocal rank, and precision at k=10, respectively (Section 2.6.2). In IR, ranking reflects the potential relevance of results to the user, with higher-ranked results considered more likely to be used (Section 2.6.2).

Table 4.1: Results by query type over WHO-105 articles, targeting a single article. n is the number of queries and $|Q|$ the mean number of tokens per query. $\overline{|R|}$, $\overline{r}$, $\overline{RR}$, and $\overline{P@10}$ are mean result set size, mean rank of the target, reciprocal rank, and precision at k=10. Augmented titles include terms from the *MixMed*-202 ontology, where link strength to a title term is >T.

Query type	n	$\overline{\lvert Q\rvert}$	$\overline{\lvert R\rvert}$	$\overline{r}$	$\overline{RR}$	$\overline{P@10}$
Title terms	105	1.5	21.4	10.0	0.177	0.023
Top doc terms	105	7.6	50.2	20.4	0.201	0.023
Top non-title terms	65	7.6	43.0	22.3	0.151	0.020
Title synonyms	104	4.3	29.6	13.0	0.189	0.025
Augmented (T=0.3)	24	179.0	34.2	21.8	0.029	0.013
Augmented (T=0.4)	24	119.3	40.8	27.4	0.095	0.008
Augmented (T=0.5)	24	61.3	39.0	21.8	0.089	0.013
Augmented (T=0.6)	24	26.7	32.5	22.2	0.086	0.026

The virtual queries at threshold T=0.5 found the target article more often than the unaugmented title at high values of k, though with a lower rank, as shown in Table 4.2 and Figure 4.3. The size $|Q|$ of virtual queries was larger in comparison to the title query, although this does not necessarily result in more or better rated results: consider the

results in Table 4.1 for augmented T=0.3 that had larger queries on average but a smaller result set and lower reciprocal rank than at larger thresholds.

Table 4.2: Results by query type over WHO-105 articles, targeting a single article. n and k are the number of queries and maximum result set size. Augmented titles include terms from the *MixMed*-202 ontology, where link strength to a title term is >T.

		Queries ranking target article in top k results						
Query type	n	$k=1$	3	5	10	20	50	100
Title terms	105	16.2%	17.1%	17.1%	21.9%	26.7%	33.3%	33.3%
Top doc terms	105	18.1%	18.1%	19.1%	22.9%	30.5%	50.5%	55.2%
Top non-title terms	65	12.3%	13.9%	15.4%	20.0%	26.2%	43.1%	47.7%
Title synonyms	104	16.4%	19.2%	19.2%	24.0%	28.9%	38.5%	39.4%
Augmented (T=0.3)	24	0.0%	4.2%	4.2%	8.3%	16.7%	33.3%	33.3%
Augmented (T=0.4)	24	8.3%	8.3%	8.3%	8.3%	16.7%	33.3%	37.5%
Augmented (T=0.5)	24	4.2%	8.3%	12.5%	12.5%	29.2%	45.8%	45.8%
Augmented (T=0.6)	24	4.2%	4.2%	12.5%	25.0%	29.2%	50.0%	54.2%

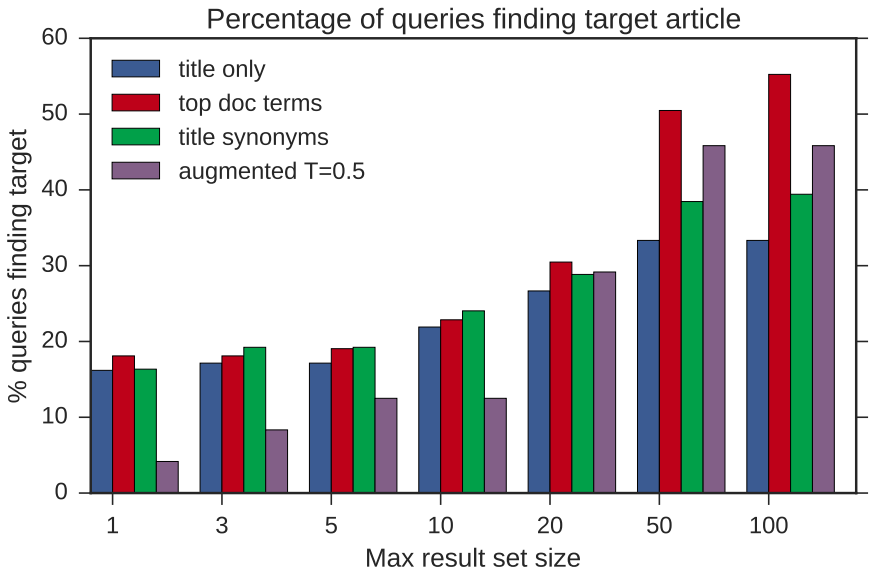


Figure 4.3: BOW results by query type over WHO-105 articles, targeting a single article.

The larger result set size and lower ranking of virtual queries sacrifices precision in favour of greater recall. This could be problematic in a search strategy focussing on finding a known target, where a user typically views only a few higher-ranked results. However, it is more advantageous for data exploration tasks, particularly for uncommon

topics, or as a component weight within more complex searches.

Queries using top terms from the target article had the best outcomes. Intuitively, this can be explained by the circular nature of these queries, which are subsamples from the target article. These queries are useful for comparison but impractical in application as they rely upon prior knowledge of the target document.

Term frequency-inverse document frequency model

The principle of augmented queries can be extended by incorporating term-relatedness weights into the selection of additional terms, and using more sophisticated NLP models such as TF-IDF.

A two-step process was used. First, the original query text was converted into a TF-IDF vector for the corpus to be searched. Then, for each t_Q token in the vector, related queries t_V were temporarily extracted from the weighted ontology where the relatedness (r_{QV} = relatedness(t_Q, t_V)) was above a threshold T. Secondly, the TF-IDF vector weight of the original term w_Q was multiplied by the relatedness of the two terms: $w_V = w_Q \times r_{QV}$. If the new combined weight $w_V \geq T$, the term t_V is added to the virtual query with w_V acting as a proxy for its TF-IDF vector weight. If a term t_V matched more than one query term, the highest calculated W_V weight was used.

Given the subjective nature of natural language interpretation, similarity weights (each an inverse of the $0.0 \ldots 1.0$ distance between two documents) are not precise but can be used to rank the contents of a resource by likely relevance to the user:

> "... while the curse of dimensionality tells us not to rely on the absolute values of distances, it is still viable to use distance values to derive a ranking of data objects." (Houle *et al.*, 2010)

For example, consider the single-term query 'flu', with a corpus TF-IDF weight of 0.664. Given a term-relatedness weight from 'flu' → 'vaccination' of 0.758 (taken from 1-NGD weights in Figure 3.17), t_Q = 'flu', t_V = 'vaccination', and $w_V = 0.664 \times 0.758 = 0.503$. This would result in a new virtual query Q_V =[(flu, 0.664), (vaccination, 0.758)].

This multiplication of weights uses a Bayesian approach to combine term-relatedness weights for a virtual TF-IDF query. This was chosen to reduce the impact of secondary terms not directly selected by the user.

Consider the earlier, multi-term query `"flu vaccine"` over the *MixMed*-50 corpus, where the most relevant results (i.e. the target documents) were expected to be the 'vaccination' and 'flu' articles. Note that the corpus dictionary tested contained only unigrams, meaning that n-grams, such as 'infectious disease' are split up or missed.

Using the TF-IDF model on the *MixMed*-50 corpus, the query `"flu vaccine"` was represented by the vector [(flu, 0.6635), (vaccine, 0.7482)]. Following the process outlined previously, a virtual query was built by adding related terms and weights from the weighted ontology to create a new document vector, containing 12 terms:

> [(vaccine, 0.7482), (flu, 0.6635), (immunity, 0.7482), (vaccination, 0.6879)[59],
> (infectious, 0.6053), (asthma, 0.5691), (virus, 0.5532), (bacteria, 0.5489),
> (sexually, 0.512), (transmitted, 0.512), (enzyme, 0.5094), (animal, 0.5094)]

A comparison of the top ranked results with document similarity weights is shown in Table 4.3 for the different approaches. Expected results are in bold. For the original query, all other results are shown in grey. The two augmented queries show only the subset of results with a calculated weight greater than 0.1.

Table 4.3: *MixMed*-50 document similarity (TF-IDF model) to original and augmented query. Augmented query results with similarities below 0.1 are not listed. Matching documents are listed by article title and cosine similarity weight. * infection/infectious_disease are identical documents, as are illness/disease.

Query `"flu vaccine"`	Augmented BOW	Augmented TF-IDF
vaccination (0.3538)	arthritis (0.3555)	**vaccination** (0.3787)
flu (0.1101)	asthma (0.3193)	immunity (0.2819)
virus (0.0299)	**vaccination** (0.3083)	asthma (0.2449)
infection* (0.0282)	cattle (0.2323)	infection* (0.1986)
cholera (0.0224)	HIV (0.1929)	virus (0.1708)
malaria (0.0196)	cholera (0.1903)	bacteria (0.1527)
health (0.0113)	immunity (0.1822)	STD (0.1518)
disease/illness* (0.0074)	obesity (0.1497)	**flu** (0.1477)
STD (0.0063)	diabetes (0.1391)	enzyme (0.1315)
cancer (0.0058)	virus (0.1291)	HIV (0.1213)
fever (0.0057)	**flu** (0.1194)	
cytokine (0.0050)	STD (0.1111)	
Black Death (0.0037)	infection* (0.1094)	
masturbation (0.0016)	pediatrics (0.1032)	
	malaria (0.1029)	

[59] The vaccination weight differs here from the earlier, single-term Q_V virtual query example due to the use of 1-NGD rather than CRW term-relatedness weights from Figure 3.17.

The TF-IDF-augmented virtual query resulted in 41 matches out of the 50 documents. The 'vaccination' article was ranked first and 'flu' was ranked eighth. Compare this to the BOW-augmented virtual query over the same corpus that ranked the targets third and eleventh, respectively. The original query returned the targets as the two highest results, although within a smaller result set (fifteen).

These results demonstrate that the original query has greater precision and recall for smaller results sets (i.e. $P@k$ and $R@k$ for smaller values of k) when the relevant documents are defined as the two target articles. Virtual queries, particularly the BOW-augmented query, resulted in lower ranks for these two targets. This lower precision is less useful for navigational IR tasks, where the user seeks a single target such as a specific web homepage. However, their larger result sets have greater potential for data exploration.

Relatedness weights were higher for the TF-IDF-augmented query and, in comparison to the original query, for both virtual queries after the first ranked result. Although the weight values are not significant in themselves other than as indicators of rank order, the larger weights lend themselves to multiplicative or other combinations with alternative ranking relatedness measures, such as spatial proximity.

4.4 Topic modelling

Topic modelling estimates approximate themes in a corpus that can be used when matching related documents. Each topic consists of a weighted set of terms, where each term can appear in more than one topic. Topics are not static, as the corpus and modelling parameters used affect their generation: each time a Latent Dirichlet Allocation (LDA) model (Section 2.2.2) is trained on the same corpus, it defines different topics. However, a LDA model, once trained, can produce comparable document vectors of topic weights based on its corpus and dictionary.

A 100-topic LDA model from gensim was trained over the WHO-105 Wikipedia articles. LDA results for virtual queries (Figure 4.4) showed a greater improvement in recall for larger result sets, in comparison to the BOW model (Figure 4.3). In particular, the improvement in recall at higher ranks was more striking in comparison to other query types, including the top document terms.

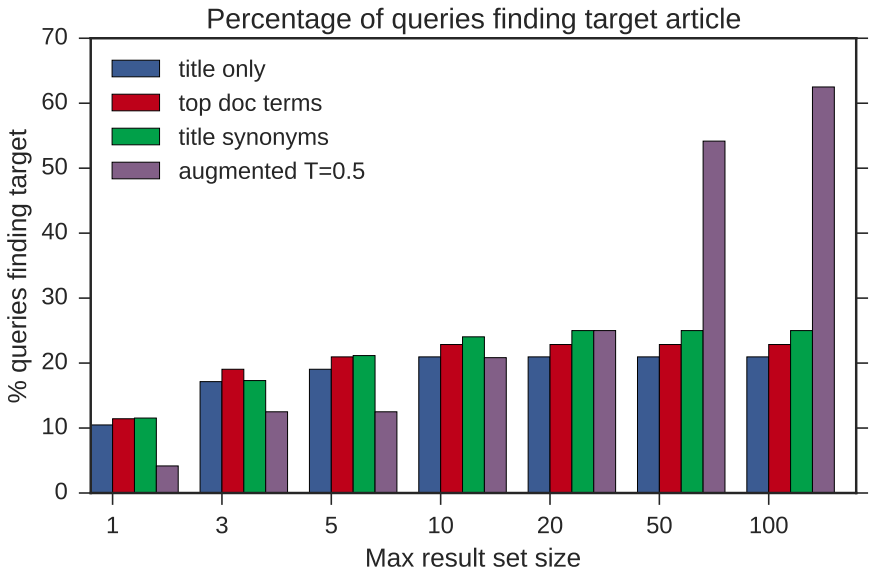


Figure 4.4: LDA results by query type over WHO-105 articles, targeting a single article.

Updating ontologies

Topic modelling has potential beyond similarity indexing of documents. The LDA model produces lists of terms that can differentiate between corpus documents, and includes probabilities of their importance within topics. This introduces a couple of possible applications:

- A source of terms to generate a new weighted ontology for a data resource, or to enhance an existing ontology (Section 3.3).
- Semi-automatic production of themes, as a kernel to build a hierarchical ordering of terms or documents upon.

LDA topic term weights can be used as a source of term-relatedness weights, for example to augment CRW-based weights (Section 3.3.2) via topic and term-in-topic probabilities. Because LDA topic weights are probabilistic, all term weights in a topic sum to one. Therefore, a relatedness weight can be calculated by summing term probabilities for two terms within a topic where, if both terms appear in more than one topic, the strongest weight would be used. The strongest weight can be calculated as the largest sum or the sum from the topic with the largest mean probability across all corpus documents.

Other potential sources of foundation terms include manual lists, domain-specific vocabularies (such as Medical Subject Headings (MeSH), in the health domain), and NLP

resources such as a corpus dictionary, as used in BOW and TF-IDF models. Disadvantages of the dictionary are that it is unweighted and is likely to contain a very large list of terms, many of which would be of low importance in the corpus.

LDA and other NLP models could also be used to update a weighted ontology with user feedback, by extracting text queries and selected documents from search logs. Bundling query text together with the selected document creates expanded documents that are suitable as inputs for training a new LDA model. An existing LDA model can be updated with new documents, or a new model can be trained. This is recommended for any dynamic corpus with a vocabulary that is likely to change over time.

One advantage of using this type of feedback is the incorporation of common query terms overlooked by data providers, such as common misspellings that would not be found in the original descriptive content or metadata. For example, if many users searched for `"diabeties"` and selected a document about 'diabetes', this misspelling would have a high TF within the expanded corpus.

Using LDA topics as themes in a terminological resource would require a cautious approach as topics are not stable and do not necessarily match human judgements of semantic themes. However, the term and topic probabilities could be used as graph edge weights in clustering algorithms, as discussed in Section 3.4. The LDA model is a more reliable source of topics than clustering terms by their estimated relationship strengths, particularly given the LDA calculation of topic and topic-term probabilities.

4.5 Summary and discussion

In this chapter, natural language processing (NLP) methods to measure document similarity were tested in combination with tailored terminologies (Chapter 3) for their effectiveness in information retrieval (IR). Topic modelling with LDA was proposed as a method to extract seed terms from a resource for building or expanding an ontology of terms.

An IR request is more likely to return satisfactory results if the terminology used in the input document is similar to that used in the resource being searched. However, it cannot be assumed that every user and data provider will always have a similar perspective and use a similar vocabulary. Terminologies, including public or specialised

ontologies, can be used to augment input text to improve the likelihood of a relevant match being returned.

Searches using the LDA model resulted in high recall rates for large result sets, indicating relevance of the extracted topic terms to the data source. Therefore, it was concluded that LDA topics are a potential source of terms applicable to the data source as a whole. These extracted terms can potentially be used as seeds to add to — or create — a tailored ontology. As LDA groups terms roughly into themed topics and calculates term-in-topic probabilities, it also has potential as an additional source of term-relatedness weights.

It is important to note that LDA is a generative model and so results will vary when run over the same documents; however, it can be relied upon to return terms useful to a terminology.

It was shown that short input texts (for example, a brief query or descriptive text attributes of a data record, could be augmented with related terms extracted from the a local ontology before comparing them in a search. When applied to a user query, this is a way of merging contexts of data and user by augmenting the terminology used in an IR request.

The purpose of contextualising input text was to retrieve a greater number of results of potential use to a user. Each record in the data resource was treated as a document within a corpus, and NLP methods were used to index the documents by their similarity to each other. An input text document was then assigned a similarity weight in comparison to each resource document.

Results from virtual queries had a lower precision and greater recall than unaugmented queries. Ranking documents based on this larger set of results was considered more suitable for exploration than navigational IR tasks.

Calculated weights estimating the likelihood of document matches were higher for virtual queries than unaugmented queries, providing greater scope for their combination with other weights that focus on different aspects of a search, such as spatial proximity. The next chapter investigates methods for including data resource, query and user context in spatial relatedness estimates that can be used to rank geographic information retrieval (GIR) results.

5 Matching locations

The layout of this chapter is as follows: after introducing the semantic spatial search problem in Section 5.1, Section 5.2 addresses aspects of spatial context for data and user, including development of a multi-dataset spatial index based on region overlap ratios (label 1 in Figure 5.1). A non-deterministic proximity measure based on distances is described and tested in Section 5.3 (labels 2–3 in Figure 5.1). Results are discussed in Section 5.4.

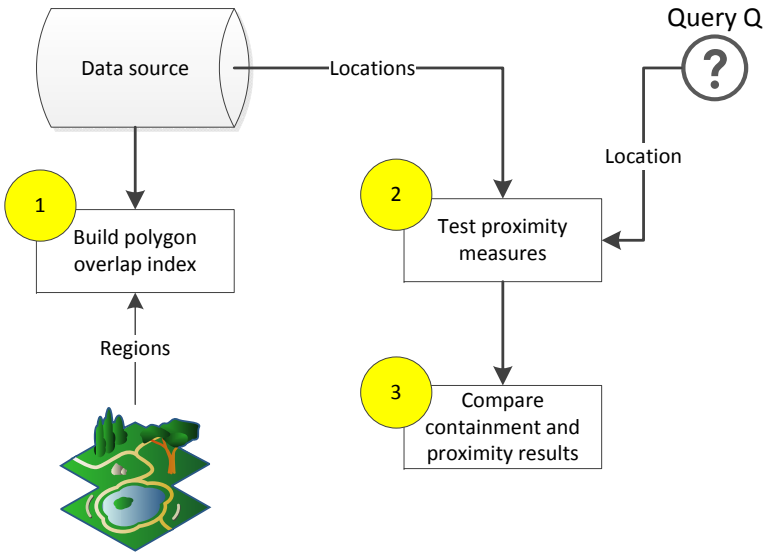


Figure 5.1: Summary of Chapter 5. The yellow circular labels are explained in text.

5.1 Introduction

Spatial datasets include location context within their metadata and individual data records. Similarly, users have their own spatial context — both in general, such as their current location, and relating to a specific request for information. Location can be the semantic link between data and user where other contexts differ widely.

Whilst text-based methods lend themselves to creation of semantic indexes (such as a trained Latent Dirichlet Allocation (LDA) topic model, Section 4.4), spatial data is not so easily indexed for semantic searches. Spatial indexes such as k-D, quad- or R-trees improve the efficiency of searches over geometries (Hjaltason & Samet, 1995, 1999) but do not account for location *meaning*. For example, whether a larger feature (such as a shop) or sparser features are worth a greater travel distance.

Online searches for spatial locations use topological operators (Section 2.4.2) like within, overlaps, and within-a-distance-of, where results are either true or false. For example, a feature is either contained within a region of interest (ROI) or it is not. This leads to spatial search results that do not allow for uncertainty, such as an imprecise current user location, a fuzzy ROI boundary, or features found adjacent to a named target region.

This chapter investigates non-deterministic methods for ranking locations by spatial proximity, including by overlap proportion and distance. Firstly, a preprocessed index of polygon overlap ratios was developed that can be used to find intersecting regions or for probabilistic spatial ranking of regions (Larson & Frontiera, 2004).

Secondly, a proximity measure based on distance measurements was developed to produce confidence levels in the spatial relevance of features relative to a target geometry. It uses an exponential decay function, with a decay constant adjusted according to data or query context. The process was tested on point features within a ROI, and can be extended to other geometry types.

5.2 Retrieving spatial context

A range of user and data spatial context can be extracted for a geographic information retrieval (GIR) request, including user and query locations, data records, and metadata, as described in Section 2.1.1 and Section 2.4.2.

Although in some cases a target geometry can be retrieved directly from a request, for example as a search region manually entered into an online map interface, they can also be referred to indirectly, such as by a place name (toponym) in query text.

Text queries can define a target location implicitly (`"bus stops [near me]"`) or explicitly (`"bus stops in Perth"`). Interpreting spatial queries depends upon contextual information including current user location — searching for `"Perth"` from Australia or Scotland, for example. Consider a user at Perth Airport searching for bus stops (Figure 2.8). Spatial context in this case includes user location, search area (if specified), the extent of the suburb the user is currently within, and the extent and contents of a dataset of bus stops.

Where available, the current location of a user can be extracted from the originating request, for example as latitude and longitude coordinates retrieved via the HTML5 geolocation Application Programming Interface (API) from the originating device.

Place names (toponyms) are an important aspect of spatial context. Automated techniques have been developed to extract toponyms from texts, and public resources are available to translate them into one or more geometries (Section 2.1.1). It should be noted that public place name repositories are general in scope rather than specific to a dataset, and in some cases have usage restrictions. They also do not include identifiers specific to an organisation, such as asset numbers or district codes.

This chapter is concerned with geometry context from both a user query and the data to be searched. The proximity ranking method proposed in Section 5.3 assumes that a ROI or other target feature of interest (FOI) geometry has been retrieved from a user query. It also requires access to dataset geometry — ideally as individual records, but a single extent can also be used. The next section discusses creation of a resource to assist in discovering a query ROI.

5.2.1 Spatial overlap index

One approach for finding spatially related features is by overlap or intersection. As described in Section 2.4.2, a number of measures exist, ranging from simple overlap (true if features overlap by any amount, otherwise false) to probabilistic spatial ranking (Larson & Frontiera, 2004). For all but the simple topological operator, calculation requires measurements of areas, including the intersection region.

Geometry operations such as calculating intersections are expensive, hence a cross-referenced, pre-computed spatial index can improve the efficiency of semantic spatial searches in comparison to on-the-fly calculations. The aim for a spatial overlap index is to relate polygons in multiple datasets hosted together (for example by a data provider or in a geoportal), where it would be updated as each new or edited dataset was published. A large organisation with multiple departments and data policies can also maintain a comparative index between its spatial datasets. Also, an index can be maintained for a search application or service and updated upon discovery of new datasets.

Such an index enables quick comparisons of regions without expensive spatial operations. Once constructed, it can be used by applications and services with limited spatial capabilities. Potential applications of this method include ranking features by overlap ratios relative to a query region, and finding alternative search regions from a named boundary.

One use case is to link between hierarchically aggregated boundaries, such as Australian Bureau of Statistics (ABS) Statistical Areas[60]. These are the main geographic regions used by the ABS to collect and report census and other statistical data, and describe four levels of detail across Australia from the largest SA4 regions (107 as at the July 2016 census) down to the smallest, SA1 (more than 55,000). Each level completely covers Australia and the levels are hierarchical, so that each SA4 is aggregated from a group of SA3s, with no overlaps. Similarly, SA3s contain SA2s, which each contain SA1s. SA1s are aggregated from mesh blocks that can also be related to non-ABS regions including suburbs, Local Government Areas (LGAs), and postal areas[61].

A polygon overlap index compares regions irrespective of feature type or theme. For example, consider the Western Australian wheatbelt, incorporating a number of LGAs grouped into five sub-regions. A link could be found from the Yilgarn LGA (south-east of Figure 5.2ii) to the Mukinbudin SA2 in Figure 5.2i. Similarly, a query `"average income in wheatbelt"` could be answered by linking from the combined wheatbelt extent to statistical areas that overlap it[62].

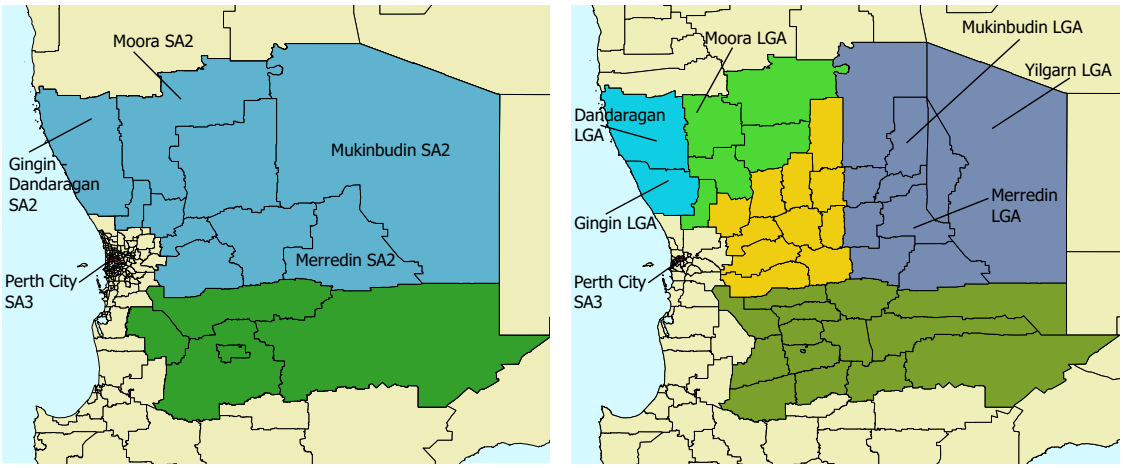
An index of regions, based on overlapping area ratios, is proposed to augment traditional spatial indexes for location and distance. The index links overlapping regions alongside precalculated 0.0...1.0 relationship strengths, calculated from area ratios. In essence, it is a spatial equivalent to a terminological weighted ontology, as described in Chapters 3 and 4.

The process for updating the index with a new region dataset is outlined in Listing 5.1. As each new dataset is loaded, pre-loaded features are filtered to extract any intersecting regions (line 4). Filtered feature areas are extracted (line 6) and the area of their

[60] http://www.abs.gov.au/websitedbs/d3310114.nsf/home/autralian+statistical+geoegraphy+standard+(asgs), accessed 29/1/2017

[61] http://www.abs.gov.au/ausstats/abs@.nsf/Lookup/bySubject/1270.0.55.001~July2016~MainFeatures~NonABSstructures~10008, accessed 8/7/2017

[62] Note that care must be taken when working with spatial statistics. This use case assumes that rates or counts would be associated with matching statistical regions rather than the initial query region.



(i) SA2 Statistical Areas, within two SA3 areas (north and south)

(ii) LGA Local Government Areas, within five sub-regions

Figure 5.2: SA2 and LGA areas within the wheatbelt region of WA[a]. Note that the Mukinbudin SA2 (south-east) contains several LGA regions, including one also named Mukinbudin.

[a] Data sources: ABS LGA and SA2, 2011: http://www.abs.gov.au, accessed 31/10/2014. Wheatbelt Development Commission, WA Government. http://www.wheatbelt.wa.gov.au/our-region/maps/, accessed 29/1/2017.

intersection with input features calculated (line 7). Combined areas, i.e. unions, are calculated by adding the separate areas and subtracting one of the repeated intersection areas (line 8). Three overlap ratios are calculated (lines 9-11) and then recorded with references to the features they link. Features that are adjacent but not overlapping have zero values for all three ratios, hence are not added to the index.

Listing 5.1. Loading a new dataset into a polygon overlap index.

```
1  input: feature dataset F
2  begin
3      foreach feature f in F:
4          FR ← set of features that overlap f
5          area ← area of f
6          A ← areas of FR features
7          I ← intersection areas f ∩ FR features
8          U ← (area + A) − I
9          intersect ← I ÷ U
10         overlap1 ← I ÷ area
11         overlap2 ← I ÷ A
12         record f, FR, intersect, overlap1, overlap2 in index
13     end
14 end
```

The three calculated ratios relate the intersection area to combined and individual areas: for regions $r1$ and $r2$, *intersect* is relative to the combined area, *overlap1* is relative to $r1$ and *overlap2* is relative to $r2$. The calculations are described by Equations 5.1 and 5.2,

where the two overlap values are $O(r1, r2)$ and $O(r2, r1)$. The symmetric intersect ratio I is a useful general measure of spatial relatedness whereas the asymmetric overlap ratios favour one of the two polygons, which is useful for measuring their relative importance in the relationship.

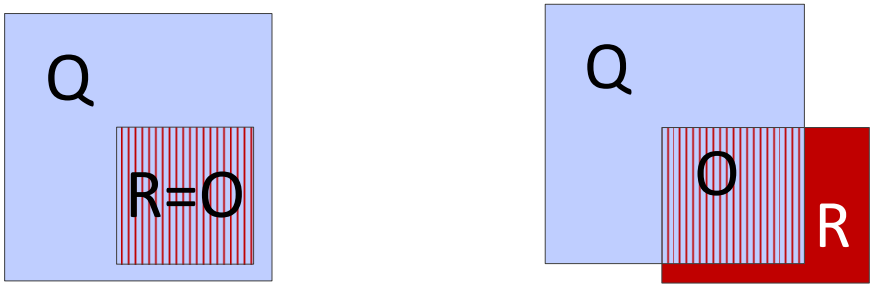
$$\text{Intersect } I(r1, r2) = (r1 \cap r2) \div (r1 \cup r2) \tag{5.1}$$

$$\text{Overlap } O(r1, r2) = (r1 \cap r2) \div r1 \tag{5.2}$$

for regions $r1$ and $r2$

Consider query region $r1 = Q$ and comparison region $r2 = R$ in Figure 5.3. The two overlap ratios $O_R = O(R, Q)$ and $O_Q = O(Q, R)$ favour regions R and Q respectively: in both examples, $O_Q < O_R$ because the intersection overlaps a smaller proportion of Q than of R.

The asymmetric measure O_R describes the relevance of region R to the query area. In Figure 5.3i, the O_R weight is 1.0 as this region is completely contained within the query region. By comparison, O_R in Figure 5.3ii is smaller as R is partly outside the area of interest. This was used to search for alternative regions to a test polygon R by selecting those with an O_R ratio above a threshold value.



(i) Q contains R, with overlap O (ii) Q overlaps R, with overlap O

Figure 5.3: Overlapping query region (ROI) Q and comparison region R.
Area of Intersection $I = Q \cap R$.
Area of Union $U = Q \cup R = Q + R - O$.

An example index was implemented using Geospatial Data Abstraction Library (GDAL)[63] and Geometry Engine - Open Source (GEOS)[64] Python libraries via the GeoDjango Web framework, with results saved in a PostGIS/PostgreSQL database. Features within the same dataset were not tested against each other as tiled regions were assumed.

[63] http://www.gdal.org, accessed 20/1/2017
[64] https://trac.osgeo.org/geos, accessed 20/1/2017

The index was constructed for three input datasets, each in either GeoJSON or shapefile format, although the implementation can be modified to accommodate other formats. Datasets loaded were: Suburbs in Perth, WA[65], Australian states[66], and Australian Public Health Network regions[67].

Each polygon or multipolygon dataset was loaded into a PostGIS spatial database via the object-relational mapper (ORM) of the GeoDjango web framework. During loading, features already in the index were filtered to find those that intersected with each dataset feature and intersection areas calculated for each, as shown in the Structured Query Language (SQL) statement in Listing 5.2. Intersection areas were then combined with region areas to calculate and record the three overlap ratios for each pair of overlapping regions, following Equations 5.1 and 5.2, with links to the regions.

Listing 5.2. Spatial SQL intersection query (PostGIS). The test region 'Belmont' is a multipolygon feature containing a single region with 125 vertices, represented here by '...'. The dataset containing Belmont has ID=31.

```
SELECT r.*, (ST_INTERSECTION (
    r.boundary,
    ST_GEOMFromEWKB('...'::bytea))) AS "intersection"
FROM Region r
WHERE (NOT r.dataset_id = 31)
    AND ST_Intersects (
        r.boundary,
        ST_GEOMFromEWKB('...'::bytea))
    )
```

Once constructed, the index was tested by searching for any regions intersecting the suburb `"Belmont"` where the recorded overlap ratio was 0.5 or above. This was compared with on-the-fly filtering, using the same SQL query and calculation as for index construction. On-the-fly filtering by overlap area took more than 750 times as long as the equivalent index look-up, as reflected in timing results:

```
#seconds taken for 1000 iterations of each method:
timeit.timeit(testIndex, number=1000)
    2.527310848236084
timeit.timeit(testFiltering, number=1000)
    1898.2958090305328
```

Belmont was selected as one of many examples of Australian suburb names that are

[65] ABS suburbs, 2011: http://www.abs.gov.au, accessed 31/10/2014
[66] ABS states, 2011: http://www.abs.gov.au, accessed 31/10/2014
[67] PHN regions: http://www.health.gov.au, accessed 23/4/2015

reused in different states - examples of 'Belmont' suburbs can be found in New South Wales, Queensland, Tasmania and Victoria as well as Western Australia. This demonstrates use of the polygon overlap index for disambiguation of regions — a deep search of record descriptions can find the name 'Belmont', with the index verifying its state.

As the index is only of use for known region datasets, unindexed regions would still require live calculations. Overlap ratios are also not appropriate for all geometry types. The next section describes a proposed proximity method for ranking spatial features based on distance from a target ROI or other FOI, in response to a GIR request.

5.3 Measuring proximity

The ranking method described in this section uses distances between features and a target, as well as spatial context such as extent size, to calculate 0.0...1.0 proximity weights for features in relation to the target. It includes user context from a GIR request and data context so that feature or dataset weights from different sources can be compared.

The perception of nearness is affected by user and query context, including search area size, but also by the context of the data to be searched. Five kilometres (approximately a one hour walk) may be considered a long distance for concentrated spatial features, such as small, densely-distributed building footprints or common points of interest (POIs) in an urban area. The same cannot be said for sparse data, e.g. airstrips used by the Royal Flying Doctor Service (RFDS) to serve remote and rural regions.

Online searches typically employ a simple spatial search, for example to find spatial features contained within a bounding box. This does not handle uncertainty as results are not ranked: instead, features are classed as either relevant or irrelevant. A more flexible arrangement is a weighted proximity measure that can rank features by likely relevance to a target location. An exponential decay over distance values is an example where the perceived nearness of a feature diminishes as its distance from the target increases.

Figure 5.4 compares proximity weight calculations based on distance from a target region. The first, `P1`, is simple containment — one if a feature is within the target, or zero if it is not. Adding a distance buffer around the ROI gives a similar cut off but

at a distance larger than zero. `P2` and `P3` use exponential decay functions to calculate 0.0...1.0 weights based on distance from the target, where smaller distances imply greater proximity. The steeper rate in `P2` means that a point using it has a weaker relationship with the target region than a point at the same distance using the `P3` rate. This represents a smaller perceived 'nearby' distance for P2 compared to P3.

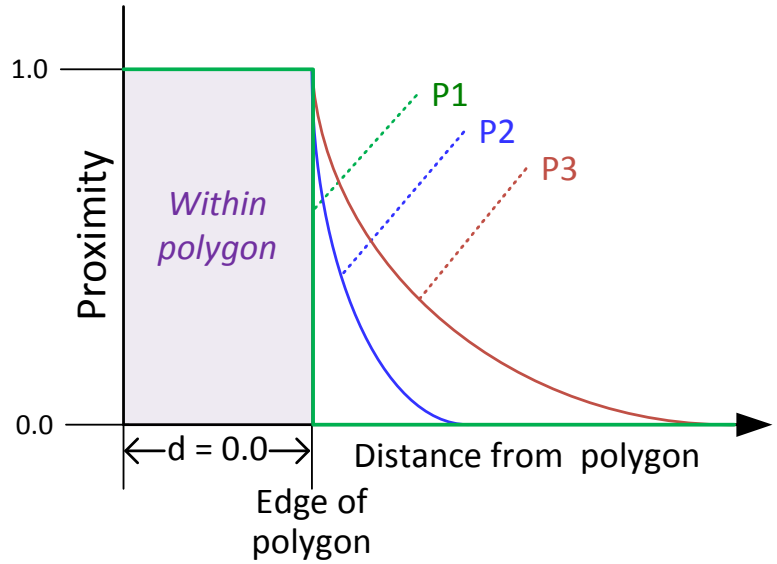


Figure 5.4: Proximity versus distance. P1: simple containment. P2 and P3: weights calculated from distance values using exponential decay functions.

Exponential decay functions were calculated from distances with the rate of decay, and therefore perception of what is considered 'near', controlled by a constant (λ). Equation 5.3 gives the general formula for a 0.0...1.0 proximity measure using distance d from the target FOI and a decay constant λ, based on region size. For features contained within a target polygon, $d = 0$.

$$\text{Proximity } P(d) = \begin{cases} 1, & \text{if } d = 0 \\ e^{-d/\lambda}, & \text{if } d > 0 \end{cases} \tag{5.3}$$

for distance d and decay constant λ

A smaller query region size is assigned a narrower surrounding zone of uncertainty, i.e. a steeper rate of decay, to reflect its precision. The λ constant is calculated from region size and was inverted to $1/\lambda$ in order to ensure a steeper rate of decay for smaller regions by producing an exponent $-d/\lambda$ in Equation 5.3 with greater magnitude than for a smaller region. Decay rates as indicators of perceived nearness were tested by

deriving λ from ROI or dataset extent size.

In order to scale relative to region size, decay rate values were calculated based on a polygon *crossing distance*, i.e. the maximum straight-line distance across the polygon, which can be calculated accurately or substituted with a metric. An example metric treats a polygon as if it were circular, using its area A to calculate a radius r (Equation 5.4). For a spatial dataset, this value can be calculated from its extent or, where it contains polygons, as the average feature radius.

$$\text{radius } r = \sqrt{\frac{A}{\pi}} \quad \text{for area } A \tag{5.4}$$

Three test ROI polygons were selected based on toponymic similarities such as could occur from ambiguities in a text query. The polygons are the suburb `"North Perth"` and the Public Health Network regions `"Perth North"` and `"Tasmania"`, illustrated in Figure 5.5. The Tasmanian PHN was chosen because this region contains a suburb named Perth. Their areas are approximately 3km^2, 3,027km^2, and 67,720 km^2 respectively. North Perth is completely contained within the PHN Perth North region.

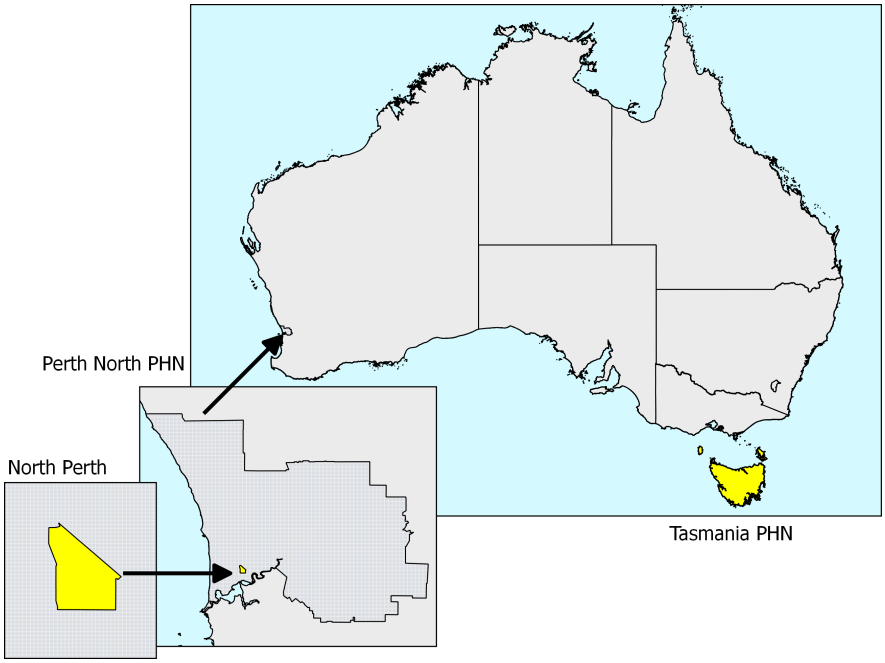


Figure 5.5: Location of three North Perth regions: North Perth suburb, Perth North PHN, and Tasmania PHN (containing a suburb named Perth).

Proximity measures were tested on two POI datasets: WA public transport stops from

Transperth[68] (15,550 points), and public toilets[69] (WA extract, 908 points). Distances were calculated between each point and the test polygons using the geos Python library. Any points within a ROI have a distance of zero, and hence had a proximity of one. The proposed method can be adapted by substituting other distance measures, such as described in Section 2.4.

The general method can be adapted to suit likely searches by altering distance measures and/or the decay rate. Where no query region is available, other contexts can be used to determine the rate of decay, for example by the density of test features; the resulting decay rate can be applied to feature distances relative to a target point location.

Decay rate constants were calculated based on three reference values: the query ROI polygon, the extent of the test dataset, and the average area per test feature, i.e. the dataset extent divided by the number of features. The latter is an approximation of feature density where the true values are not easily retrieved from records, assuming a fairly even dispersion of similarly-sized features. Proximity measures used for different decay rates over these areas are listed in Table 5.1.

Table 5.1: Proximity measures tested, where decay constants $\lambda_1 = \frac{\sqrt{A}}{2.0}$ and $\lambda_2 = \sqrt{\frac{A}{\pi}}$.

Proximity measure	Decay constant	Region
P_1	λ_1	A = area of ROI
P_2	λ_2	A = area of ROI
P_3	λ_1	A = area of test data extent
P_4	λ_2	A = area of test data extent
P_5	λ_1	A = average test feature area
P_6	λ_2	A = average test feature area

Two distance measures were tested: polygon edge-to-point and point-to-point — both direct distances rather than route-based. It was assumed that geometries were described using valid spatial projections for distance calculations. Other distance measures such as great circle distance for widely separated global features or network distances as outlined in Section 2.1.1 could be substituted into the general formula as required.

[68] WA Transperth bus stops http://www.transperth.wa.gov.au/About/Spatial-Data-Access, accessed 6/3/2015

[69] WA public toilets, 2016: http://data.gov.au/dataset/national-public-toilet-map, accessed 18/3/2016.

In the absence of an accessible query region area, alternative methods are required for calculation of a decay rate value. Two tested approaches derived decay rates from dataset context — its extent (P_3 and P_4 in Table 5.1), and a metric for feature density (P_5 and P_6: extent area divided by the number of features). These were tested against edge weights for comparison with the ROI-based proximity weights.

Figure 5.6 is a detail view of a simple containment search for bus stop points in North Perth. Bus stops either side of the same boundary road are contained by different suburbs, hence a search by suburb may find only one of a pair of bus stops used for inward and outward bound versions of the same bus route. Using a proximity weight would alleviate this problem by assigning non-zero weights to nearby bus stops.

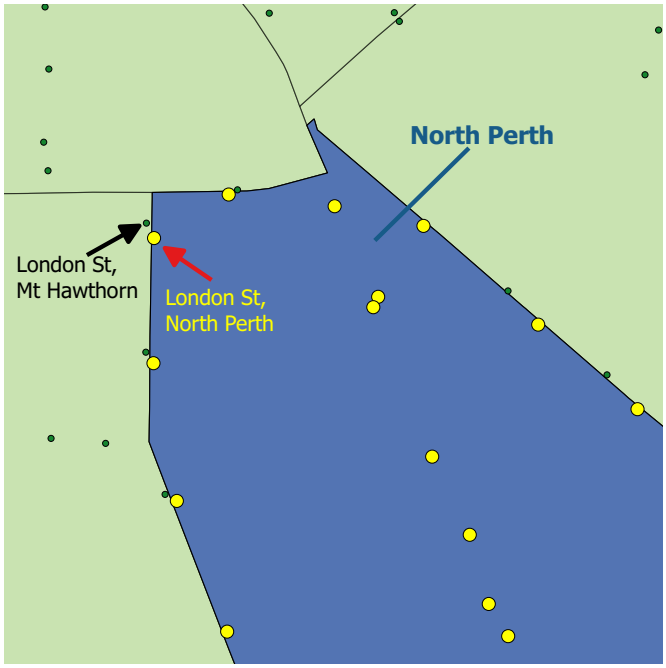


Figure 5.6: Proximity of bus stop points to North Perth (detail), where green circles are outside and yellow circles are inside the suburb.

Calculating proximity weights from the shortest distance to the ROI, i.e. the edge distance, assigns the same weight to all points contained within the target region: approximately 1.0. To rank *all* features, a distance from the target anchor point (such as its centroid, as described in Section 2.1.1) can be used instead of edge distance. This variation can also be used for queries against a target point, such as user location.

Query-based decay constants

The number of features with proximity weights at or above a given threshold are listed in Table 5.2 for the public toilet point set. The P_1 and P_2 proximity measures were calculated from the area A of the query region, using decay constants $\lambda_1 = \frac{\sqrt{A}}{2.0}$ and $\lambda_2 = \sqrt{\frac{A}{\pi}}$.

Table 5.2: Counts of public toilet point features per North Perth region at different proximity thresholds (T), using decay constants based on query region area. The POI dataset contains 908 points in total.

T	Proximity measure	Counts (Suburb)	Counts (PHN)
1.0	contained	8	444
0.99	P_1	8	449
0.9	P_1	8	525
0.75	P_1	11	655
0.5	P_1	16	729
0.99	P_2	8	451
0.9	P_2	8	534
0.75	P_2	11	673
0.5	P_2	17	741

These results demonstrate the impact of target region size upon the number of search results. The PHN area is approximately ten times the size of the suburb and subsequently resulted in many highly-weighted, non-contained points. For the smaller suburb region, there is negligible difference between the number of results from λ_1 (half the square root of the area) and λ_2 (radius metric).

The larger size of the PHN illustrates the shallower decay from the latter, with higher feature counts returned. Both constants returned viable estimates, however the slightly more conservative P_1 measure is recommended for general use. Tuning of the decay constant can be used to tailor proximity for specialised data sources, for example by using expert knowledge.

For both query regions, a point-to-point distance relative to a precise user location would be a more appropriate search. However, where such a reference point is unavailable, query region size is a reasonable alternative, as long as the query region is not overly large. It is recommended that this method be applied where the query region is smaller than the extent of the dataset being tested.

As all proximity measures are based on distance measurements, ranking order of test

features within the same dataset was as expected. Results from testing the P_1 proximity measure for two test polygons (North Perth and PHN Perth North) are illustrated in Figure 5.7. Note the steeper curve for the smaller suburb region, restricting its effects to smaller distances. This means that points close to North Perth are rated highly, but points deep within neighbouring suburbs have much lower weights.

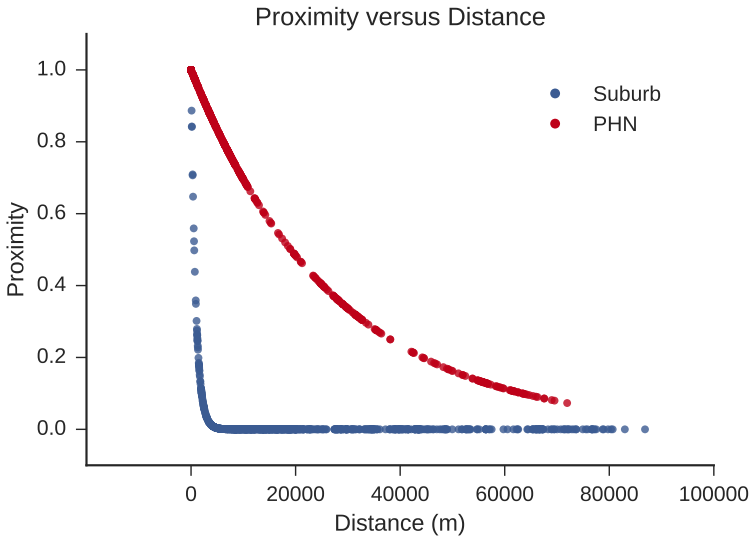


Figure 5.7: Proximity versus distance of public toilets relative to the North Perth suburb and Perth North PHN region, using proximity measure P_1.

Map visualisations of spatial ranking results using proximity measure P_1 are given in Figures 5.8 to 5.9. Figure 5.8 highlights the 15 top-ranked public toilet point features relative to the North Perth suburb of interest, with points symbolised by stars — yellow for the eight contained points and red for points outside North Perth. Figure 5.9 demonstrates the same P_1 measure over these features for the PHN Perth North region, with features contained in the PHN highlighted in blue.

Compare non-contained points in each case: ID 44759, north of North Perth (ranked 10) in Figure 5.8, and ID 15889, north-east of PHN Perth North (ranked 472) in Figure 5.9. Respectively, they have proximity weights 0.843 and 0.970 and are 150.5m and 829.7m from the ROI boundary. In other words, the PHN point has a much higher proximity weight but at a greater distance from the target region.

This is not a problem when searching a dataset with a single query region, as results can be ranked against each other in order of proximity weight. However, comparing results (for example, using alternative ROI) could give misleading results. In such a case, it is recommended that feature weights be combined with a measure of confidence

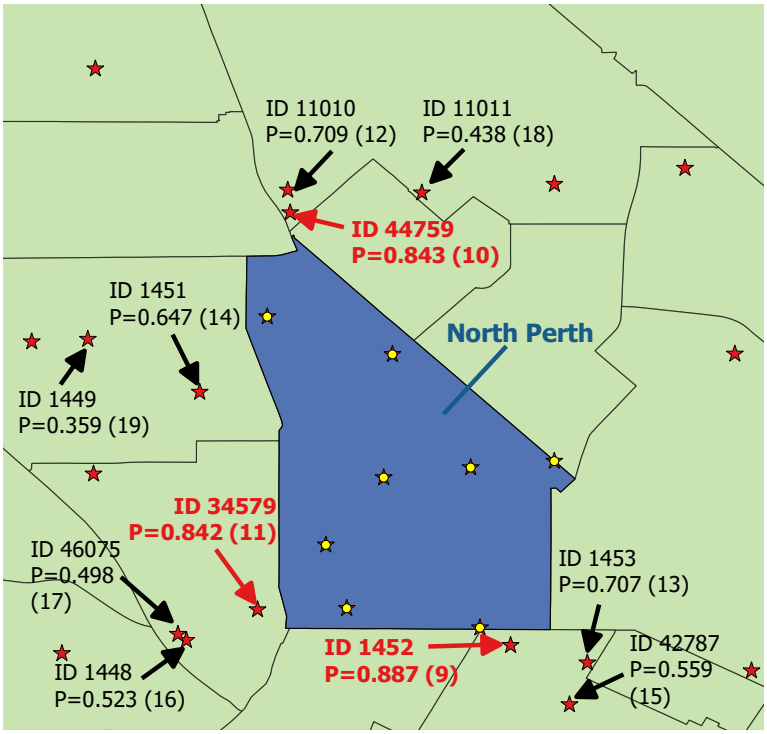


Figure 5.8: Proximity of public toilets to the suburb 'North Perth', using proximity measure P_1. Yellow stars are points contained within the suburb, which have a distance of zero and proximity of one. Red stars are points outside the suburb, with a calculated 0.0. . . 1.0 proximity. In total, there are 8 points contained in the suburb.

in the test region itself.

By contrast, simple containment is deterministic, in that the same number of features are returned for a query with all feature weights either one (within) or zero (not within). This would always return eight bus stops for North Perth (Figure 5.8) and 444 for the Perth North PHN (Figure 5.9). It is equivalent to using the proximity measure (regardless of decay constant) and ignoring any weights lower than 1.0; the sharp cut-off means that there is no flexibility to consider nearby features as possibly relevant.

Dataset-based decay constants

Proximity measures based on dataset context were compared with query region size-based measures, as shown in Table 5.3. Measures were based on dataset extent and a metric representing feature density (P_3–P_6). Results for the six proximity measurements are shown in Table 5.3 for nearest-distance (point to query polygon edge) and point-to-point distance at a threshold of 0.9.

Weights from both extent-based metrics were particularly high: as an indicator, 160

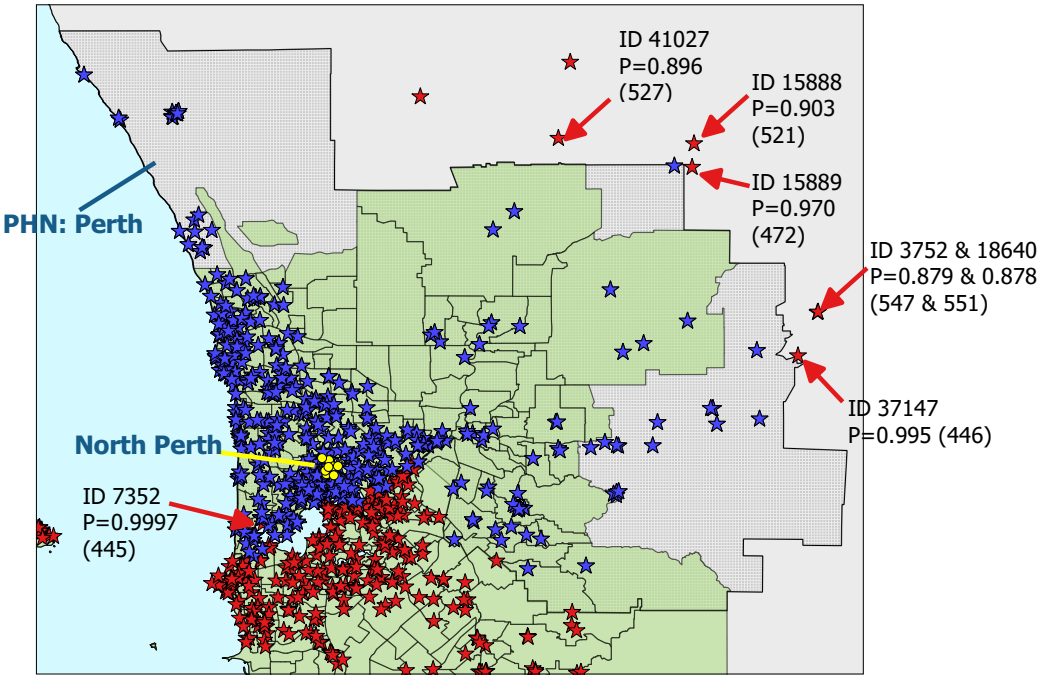


Figure 5.9: Proximity of public toilets to the Public Health Network region 'Perth North', using proximity measure P_1. Yellow points are within North Perth and the PHN, blue points are within the PHN, and red points are outside both. In total, there are 444 points contained in the PHN.

Table 5.3: Counts of public toilet point features per North Perth suburb and Perth North PHN regions at proximity threshold 0.9 for different proximity measures. The POI dataset contains 908 points in total.

	Edge-to-point		Centroid-to-point	
Proximity measure	Suburb	PHN	Suburb	PHN
Contained	8	444	8	444
P_1 (ROI)	8	525	1	4
P_2 (ROI)	8	534	1	4
P_3 (data extent)	160	588	118	8
P_4 (data extent)	180	602	138	8
P_5 (data density)	11	446	1	0
P_6 (data density)	11	446	1	0

or 180 North Perth public toilet points were found with proximity weight ≥ 0.9, in comparison to the eight it contained. Although ranking order is valid to compare results within a single dataset, the greater concentration of higher values could overwhelm weights from datasets using other metrics, and thus skew ranking results in its favour. These predominantly high-weight proximity measures could also be used internally to detect outliers with low ranking weights.

The feature-density metric had results closer to the ROI measure when tested on edge distances, as a result of the smaller area used. It performed better than query region

size for the large Perth North PHN, though this would depend upon query region size. This measurement is recommended for comparative purposes, for example when testing a dataset against multiple query regions of differing sizes.

This proximity weight was retested using centroid-to-point distances, which changed the ranking order of features slightly. This was expected, given the irregular shape of the test regions and the assumption of circular area built into the crossing distance-based metrics. The data-extent-based metrics overestimated counts for the suburb but underestimated for the PHN at a threshold of 0.9, and the other metrics returned fewer features than expected.

Although proximity weights were lower, the highest-ranked ten records for the North Perth suburb contained all eight suburb-contained points, with weights ranging from 0.242 to 0.926 for the P_1 measure. This shows promise for the metrics as a means for ranking the closest dataset features to a precise query point location.

5.4 Summary and discussion

This chapter has investigated methods for estimating spatial proximity between a data resource and query, to respond to a user query with an indicator of spatial relevance. The challenge is to estimate the spatial suitability of any given dataset to a query, as accurately as possible relative to its content and capabilities.

Comparing a region with others in the same location was one approach taken to finding and relating polygon features from user and data context, for example to find spatially related regions after initial matches by text label. A method for indexing polygons by overlap ratios was shown to find spatially related regions.

Once a target query feature was extracted from a GIR request, it was demonstrated that proximity weights relative to it could be derived from feature distances, using an exponential decay function with a decay rate λ calculated from user or data spatial context. Additional information such as uncertainty in spatial coordinates from the original data collection (Cheung *et al.*, 2004) was not required to implement this method.

Deriving the decay rate from an inverse of the target region size resulted in steeper decay

curves for smaller ROIs. This reflects the perception of what is considered nearby for a query.

Decay rates can also be tailored to unique requirements of a dataset, for example to account for thematic disparity within a spatial dataset. Consider a POI that contains many different feature types. In this case, the spatial distribution of a particular subset of themed points, such as 'schools', could be used to adjust the sensitivity of the proximity method for a spatial search for that theme. Expert knowledge is another potential source of information for selecting a decay rate, for example by considering territory size in a dataset of animal tracking data. Methods for calculating specialised proximity parameters would need to be encapsulated with the data content.

Distance-based proximity ranking methods rely upon capabilities over the underlying data. Two potential limitations are: a) restricted availability of spatial operators, depending upon feature geometry type and underlying technologies for data storage and access, and b) restricted access to individual records, including Deep Web access and privacy or security issues.

Given the range of resources available for converting spatial formats and applying spatial operations, the former problem was seen as an issue relating to processing efficiency more than availability. Performance issues in online spatial searches are an ongoing field of research, for example as described in Zhang *et al.* (2015).

Record-level access is another potential limiting factor. In some cases, metadata about a set of records, such as minimum bounding rectangle (MBR) extent, can be used instead of individual features to give an overview weight. An example is the use of polygon overlap ratios to estimate spatial relevance. Datasets with security or privacy restrictions on data record access could implement proximity tests internally and return an overview confidence level to a user without violating those restrictions. This requires encapsulation of the search actions — including proximity calculations – with the data itself.

Where individual records can be released, their proximity weights can be used to return ranked lists of features and/or to estimate the relevance of the dataset as a whole. Possible summary approaches for the latter include the proportion of features with proximity larger than a threshold weight, or the average proximity weight of all features.

One benefit of using 0.0...1.0 proximity weights rather than binary containment or overlap tests is the potential for combining ranking measures. For example, incorporating weights from ROI selection, document similarity, spatial proximity, and toponym matches into a single confidence weight. This facility can be embedded with data or managed by a search coordinator with access to individual weights.

If information about the capabilities, requirements, location and other context of a data source were encapsulated with its content and search behaviours, a user could initiate a query without needing to constantly adjust to different local knowledge, systems, syntax, and expertise levels. The next chapter will demonstrate this type of encapsulation by using online search agents, tested in a case study Web application.

6 Managing contextual searches

The layout of this chapter is as illustrated in Figure 6.1. Three stages of agent coordination are explored in Section 6.2 (label 1). Encapsulation of data resource context, content and search tasks is discussed in Section 6.3 (label 2), followed by development of the search agent framework in Section 6.4 (label 3). Search coordination and agent implementations are demonstrated in a case study Web application in Section 6.5 (label 4). Results are summarised and discussed in Section 6.6.

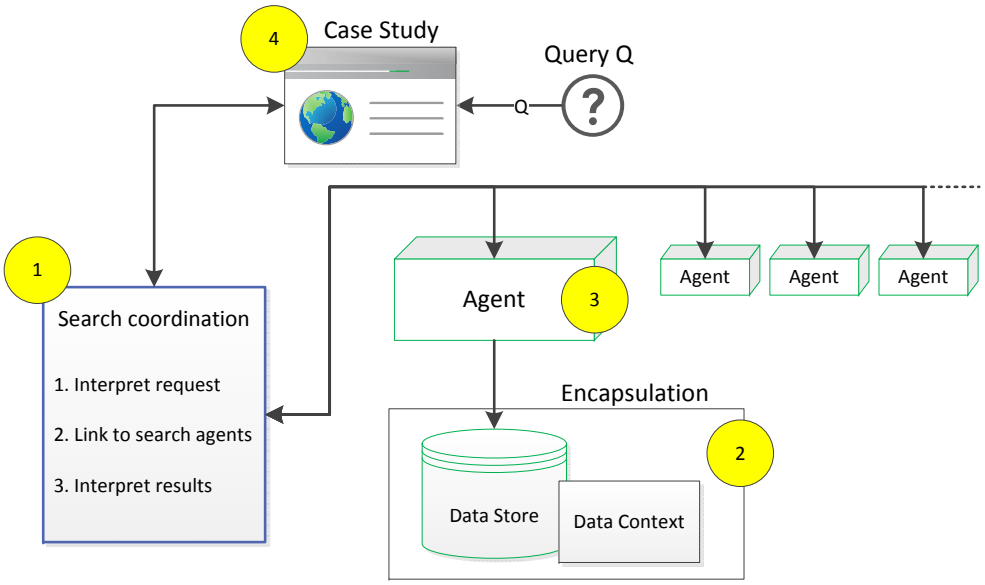


Figure 6.1: Summary of Chapter 6. The yellow circular labels are explained in text.

6.1 Introduction

Searching for data to meet an information need requires matching of user and data contexts. Existing online search applications focus on either a global approach, using broad knowledge or historical queries to infer the information need from a query, or a specialised one, tailored to a particular problem. There is a need for contextualised data search methods that can be used by a wide variety of applications and services.

The primary aim of this chapter is to investigate and evaluate techniques enabling data resources to estimate their own relevance to an information retrieval (IR) request, given the context of the query and its originating user. A secondary aim is to use deep search to facilitate the return of data records, and record-level relevance estimates, within the

same system. The focus is on retrieval of datasets, including spatial ones, rather than texts or webpages.

An 'agent' (Section 2.5) in this thesis is an autonomous software solution with access to one or more specific data resources and their contexts, responding to information requests with a relevance estimate and (optionally) a list of individual records and relevance weights. Online accessibility is provided by Web services that communicate with search agents, enabling the flexibility necessary for comparing and ranking a variety of online data sources and their records.

Existing search agents that incorporate data context from public metadata, for example by using ontology matching and other semantic techniques (Huang & Webster, 2004), overlook the wealth of contextual information locked within data resources.

Encapsulating data context and content with search capabilities enables the drawing together of informational aspects including terminology (language at the micro-level), document text (language at the macro-level), and spatial content.

A software search agent framework is proposed for managing contextual searches. This approach was tested with a case study Web application coordinating search agents.

6.2 Search coordination

Coordinating complex searches across multiple data sources requires management of complexities including data access requirements and merging of results. It was theorised that encapsulating data context and search behaviours within search agents would simplify interactions required for a search across multiple sources, in comparison to a centralised search, fully autonomous agents, or complex middle or interpretation layer (Section 2.5).

In this thesis, a 'search coordinator' is a software application layer that manages IR requests and search results and links to one or more individual search agents. Three main tasks were identified within the search coordination process: 1) interpret the request (including spatial and other user context), 2) pass the request and context to search agents, and 3) interpret results. Examples of task 3) include merging ranked

result lists, refining a search with user feedback, and visualising results and confidence levels.

Figure 6.2 illustrates the main components of the proposed search coordination system that pushes search logic towards specialist search agents. The middle tier illustrates the coordinator with its tasks of interpreting an initial query and user context (label 1: Query Parser plus optional Spatial Contextualiser), linking to search agents (label 2: Linker), and interpreting results returned by those agents (label 3: Interpreter). It should be noted that the coordination layer does not have knowledge of individual data resource contexts that are encapsulated in search agents.

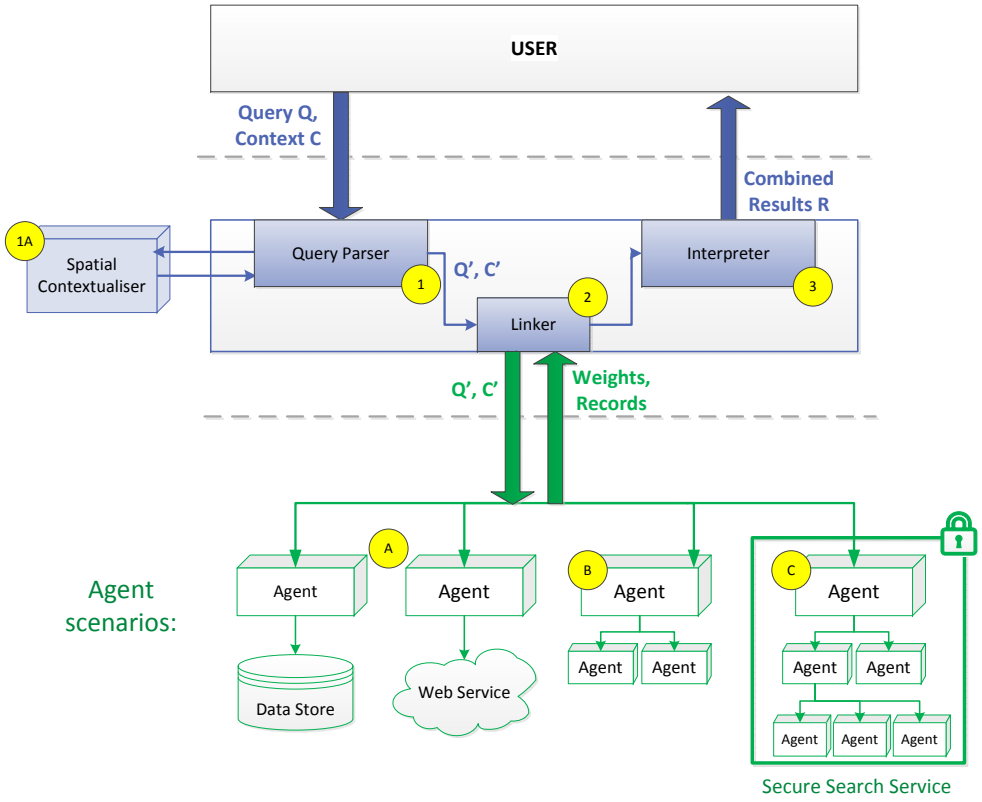


Figure 6.2: Contextual search system using agents. The yellow circular labels are explained in text.

Interpret request

First, the search coordinator processes the input query, which consists of up to four elements: 1) search topic or theme, e.g. `"hotels"`, 2) target location, e.g. geographic coordinates or a toponym like `"Perth"`, 3) spatial relation, e.g. `"near"` or `"within"`, and 4) metadata about the query and user, e.g. spatiotemporal origin or additional search limitations. The interpreter passes the original text query to agents and can also

send any one or more of these elements separately.

In the first stage, a coordinator can adjust inputs based on its global knowledge or intended purpose, for example with reference to query logs (Baeza-Yates & Tiberi, 2007; Crabtree *et al.*, 2007; Xiong *et al.*, 2017). An example of this is query suggestion, where a search interface suggests search terms as the user types. Although not implemented in this thesis, the search framework was designed to accommodate extensions such as this in the future.

The query parser (Figure 6.2, label 1) can access application-specific or global resources to adjust or augment the initial request, for example using a spatial contextualiser (Figure 6.2, label 1A) to extract a precise user location or query region from the initial request.

A target feature of interest (FOI) can be extracted from query and user context, or else from datasets being searched (Section 2.1.1). Examples include a point location, a manually entered bounding box, a set of bus stops, a named road, a nation, or a district boundary.

Potential sources of FOIs, in decreasing order of certainty, include:

1. user-specified geometry,
2. geometries from toponyms in query text (clarified against user context),
3. current user location from mobile services,
4. default search location from a user profile or Web cookie,
5. location derived from the Internet Protocol (IP) address of a Web request,
6. anchor point of the dataset to search, or
7. extent of the dataset (i.e. no spatial filter).

The first two items may differ from the current user location.

Extraction of a place name from a text query (item 2) can be implemented by a search coordinator or agent, either of which may also convert it into an initial target geometry. User location can be used in toponym disambiguation.

Link to search agents

In the second stage, query and user context is passed to one or more search agents via the linker (Figure 6.2, label 2). Each agent then compares context from the query and user to its own, and replies with the likely relevance of its data. Depending upon the capabilities of the agent, it may also return individual records with or without relevance weights.

An application can predefine a set of relevant search agents or seek relevant agents via a data discovery process. In either case, it will judge the relevance of all possible data sources for a specific IR request by relatedness weights returned by agents.

Where data records are required by the application, they can be sought with a single-pass or two-pass approach. In the first case, requests for records are sent to all known agents and responses and empty result sets are ignored. In the second, agents are first ranked by overall relevance, and then records are requested from the most highly ranked. This is similar to the chaining versus composition problem in orchestrating online processing services (Rautenbach *et al.*, 2013), as discussed in Section 2.5.1.

By using Web services as search interfaces, the original query text can be sent as an input parameter along with contextual elements such as user location and search region. The coordinator does not need any knowledge of how the service will access data and whether (or how) the agent it connects to will adapt inputs to its own context.

Interpret results

In the final stage, the interpreter (Figure 6.2, label 3) translates agent results into a response for the user. As the proposed agents can estimate the relevance of their data source(s) to a query — essentially returning a confidence level in their usefulness — each source can be ranked and then listed with the most likely matches first. A user can then choose datasets to purchase or download from the ranked list of sources.

Depending on its intended purpose, a coordinator may display a visualisation or report based on one or more results, particularly where results include record lists. Not all agent types are expected to have the capability for deep search but, when present, the logic used to extract records is encapsulated within the agent.

Other processes that could be implemented at the interpretation stage include manual query refinement, faceted search, and chained processing.

A coordinator can use a ranked list of datasets as a starting point for a new search, allowing an iterative process where the user refines their goals based on search results. Attaching 'more like this' links to each result allows a user to focus on one outcome (Noel *et al.*, 2014), for example where a coordinator suggests new data sources that have historically been accessed alongside the highly-ranked results.

Results can be faceted (i.e. categorised) by theme and/or region. For example, results from a query for `"hotels in Perth"` can either by ranked as a single list based upon an origin location or, if no origin is available, the list can be split by topics into smaller ranked lists e.g. WA, Scotland, Tasmania, and so on.

A ranked list of records can also be used as inputs for further processing. Examples include creation of automated thematic maps (Rautenbach *et al.*, 2013), user-led geovisualisation exploration (Moncrieff *et al.*, 2016), or production of new, combined datasets (Zhang *et al.*, 2010b).

6.2.1 Combining measures

Measures of relatedness between a query and dataset were proposed for both text and spatial content in Chapters 4 and 5.

Depending upon the goals of a coordinating application and the capabilities of each resource, different relevance weights can be combined. The compounding step can be carried out either by the data source itself or by a search coordinator retrieving separate weights. Example scenarios for a composite relevance weight include:

1. two-step search for target FOIs and then features: consider separate confidence levels for targets and then spatial features near them,

2. chained processes: consider the accumulated confidence level when multiple datasets are sought for one process, and

3. spatial and lexical weights: consider both location and natural language contents of a dataset.

Two combinatorial methods were considered: *probabilistic* and *weighted average*. Both equations assume that the input weights are in the 0.0...1.0 range.

By treating relevance weights as independent probabilities, they can be combined with multiplication as shown in Equation 6.1. An example application of this multiplication rule of probability is calculating the probability of relatedness between terms A and C via an intermediate term B: $P_{AC} = P_{AB} \times P_{BC}$ (Section 3.3). Conditional probabilities, such as the likely relevance of a spatial feature given a particular region of interest (item 1), is another example, as illustrated in Equation 6.2.

$$P = P_1 \times P_2 \tag{6.1}$$

$$P = P(\text{region}) \times P(\text{feature} \,|\, \text{region}) \tag{6.2}$$

for probability P

Probabilistic combination leads to lower compound weights as each multiplicative step gives a fraction of the previous values. Although only relative values are important for ranking results, this could disrupt the expected order if results are taken from agents using different numbers of probabilistic elements to calculate weights.

The weighted average method assigns a constant value to each relatedness measure before averaging them, as shown in Equation 6.3 for n weights (w_i) and measures (m_i), assuming that all individual measures are normalised to the 0.0...1.0 range. The constant weights represent the relative importance of each component (Hutchinson & Veenendaal, 2013).

$$\text{Combination measure } M = \frac{\sum_{i=1}^{n} w_i m_i}{\sum_{i=1}^{n} w_i} \tag{6.3}$$

for measure m and weight w

A situation more suited to a weighted average is comparison of data sources with different capabilities. Equation 6.4 illustrates the combination of spatial proximity and natural language text relatedness measures $m(s)$ and $m(t)$. As the context of each data source differs, the same query may be tested against datasets that can estimate

relevance based on only spatial or textual content, or a combination of both.

$$m(s,t) = \frac{w_1 m(s) + w_2 m(t)}{w_1 + w_2} \tag{6.4}$$

for spatial and textual measures $m(s)$ and $m(t)$,

and weights w_1 and w_2

An agent can use its own context to judge the relative importance of different weight elements. Alternatively, a search coordinator could define the relative importance of the lexical and spatial components and then request these as separate values. For example, if the spatial measure was considered more important than the textual in Equation 6.4, the individual weights would be adjusted such that $w_1 > w_2$. Using this method, a result with a very strong text-based match to the IR query but no known spatial context, i.e. $m(s) = 0$, could be ranked higher than a result that is spatially distance from and weakly lexically related to the region of interest (ROI).

It should be noted that a potential consequence of embedding proximity calculations into a data resource is that the calculation method may differ to that expected by a user. However, this is compensated for by adaptation to data context. Also, the IR approach of ranking results gives a range of answers to the user so that lower-ranked results are still accessible.

Spatial proximity measures calculated from data context can be tailored to user context, such as preferred travel time, where this is supplied with a query. For example, the constant in a distance decay function can be scaled as described in Section 5.3 to adjust for user preferences.

6.3 Encapsulating search and context

Relatedness measures explored in Chapters 4 and 5 used context from both users and data. Whilst user context differs for each information request, dataset context is directly related to its content. There are a number of potential advantages to encapsulating data content, lexical and spatial context, and search capabilities together:

1. Pretesting. An initial pass during a search can reduce processing costs, e.g. if a dataset has no records in the target region, there is no need for further testing.

2. Privacy and security. Where data records are held behind a firewall, an initial test can discover if the dataset is of potential interest without needing to grant user access to hidden detail (Moncrieff *et al.*, 2016).

3. Comparability. Different data resources can be ranked by likely relevance to a query region, regardless of how they describe locations or how their data is accessed.

A search system cannot assume that user context will always be available and precise. For example, if no target location could be extracted from the user or their query text, a spatial search would rely solely upon text and dataset context.

Geometry context

The context and capabilities of the data resource dictate how they respond to a request for information, including whether proximity measures can be returned for the dataset as a whole, for individual records, or separately for both.

Proximity measures discussed in Section 5.3 use feature distances whose availability depends upon the spatial capabilities of the data resource being queried. Choices related to proximity calculation are affected by dataset context, including its extent, the geometry types it contains, their feature type (i.e. topic), presence and relative importance of any outlier features, and availability of related resources (such as an appropriate network for calculating network distances).

Potential complicating factors for spatial operators used across different sources include interpreting the geometry format used, matching spatial reference systems, and accounting for large differences in accuracy or precision.

Direct distance and other proximity measures relative to a FOI, ranging from more sophisticated and accurate to simpler (and potentially faster), include:

1. distances from all dataset features to the nearest point on the FOI,

2. distances from all dataset features to the FOI anchor point,

3. overlap ratio between the dataset and FOI extents,

4. binary containment of all dataset features relative to a target region,

5. nearest distance from dataset extent to FOI, and

6. distance between dataset and FOI anchor points.

Network distances are appropriate for specialised queries such as destinations within a maximum travel time from an origin. They have additional complications: an appropriate network is required, as is routing capability to find the shortest path. Different queries over a single dataset may even be suited to different networks — consider hotels linked by roads, footpaths, and public transport routes, each with different paths and associated travel times. In some cases, alternative metrics such as Minkowski distance can be tuned to the data and applied to spatial queries (Shahid *et al.*, 2009).

Ideally, online spatial queries should trigger a deep search of individual records to avoid overestimation of relevance from an extent like a minimum bounding rectangle (MBR) (Section 2.1.1). However, some data sources either lack this capability or implement it inefficiently. Searches restricted to metadata may only have access to a bounding box or centroid point location. Also, record searchability does not imply availability of spatial operations — in the Web Feature Service (WFS) standard, for example, spatial filters such as Within are optional.

Hence, spatial searches should prioritise accurate tests where possible, but utilise fallback operations (such as 'within bounding box' or 'within x metres of the centroid') where necessary.

Toponym context

To allow a data resource to recognise relevant places in a text query, a contextual search needs access to a toponym resource. For example, each of the datasets in Figure 2.1 (federal, state, and suburb) could respond successfully to a request for a 'Perth' boundary. The originating application can use result rankings to decide which one or more best suits its primary purpose.

Automated extraction of place name labels was tested on English-language WFS data. To find the relevant WFS label attribute for use in a search, dataset metadata was searched for the first attribute name that partially matched a synonym of 'label': specifically, the first case-insensitive match to any of `["label", "name", "id"]`. For example, on an Australian Bureau of Statistics (ABS) dataset of census regions, a relevant

attribute "SSC_NAME" was found without manual intervention. Where no match is found for a data source, a default label ("<unknown>") is returned.

Technology context

The technical capabilities and requirements of the resource providing a dataset can also be considered part of its context. Consider Western Australian government data as an example, available through an online data portal[70]. As at 20 January 2017, this portal listed 809 datasets tagged as 'polygons' in a variety of data formats including WMS, WFS, GeoJSON, and Excel. They include data from different organisations and government departments, each of which can contribute several datasets.

The formats and technologies used to store data affect what is accessible and how it can be used. This includes what spatial operations are available (e.g. Within, Distance, and Intersect) and how they can be accessed and calculated. In some cases, privacy and security settings may restrict access or impose user validation constraints.

Another important aspect of technical context is the spatial reference system (SRS) that data is stored in, and whether the resource is capable of re-projecting its data on-the-fly. The resource needs to be able to report its SRS and/or transform it on request to ensure that proximity measurements are accurate.

6.4 Search agent design

Software agents were chosen as the means to encapsulate search behaviour with data content and context because they can be self-contained with respect to their data source or sources. Agents can also be made accessible via public interfaces, including Web service requests, for online searches. Each agent can use different methods internally and make use of available resources, meaning they can be adapted to suit specialised search actions and specific data source types.

Figure 6.3 illustrates a detail view of the template DataAgent, connecting to one or more data sources at the bottom and linking to a search requester (e.g. Figure 6.2, label 2), at the top. Specialist search agents, such as labelled a, b, and c in Figure 6.2,

[70] http://www.data.wa.gov.au, accessed 20/1/2017.

all inherit from the DataAgent framework but can directly or indirectly access different types of data resource and add other actions.

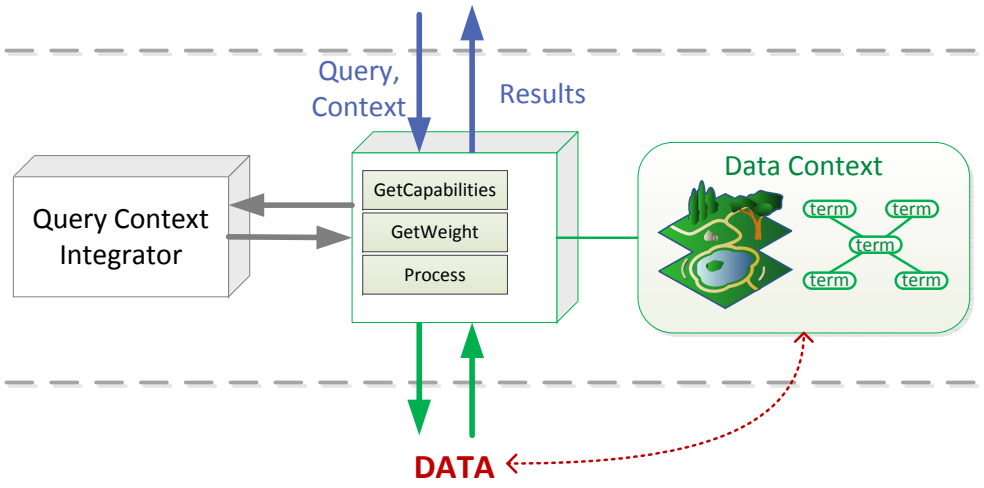


Figure 6.3: Detail of a search agent. The initial query and user context come from, and results are returned to, a search requestor or coordinator as illustrated by the middle tier of Figure 6.2. Each agent accesses one or more data sources that can be of any type, e.g. database, another search agent, Web service, or document collection.

Two components of encapsulated search, besides processing, are illustrated in Figure 6.3: the Query Context Integrator (QCI) at left and data context elements at right.

The QCI is tasked with adapting query and user context inputs to better match local data context, for example through query expansion as discussed in Section 4.3. It can use any of local data context, local content, and public resources such as domain-specific terminologies. Detail is hidden within the agent, and accessed through public interfaces.

Consider a text query `"footy in Freo"`, passed as a request to a search agent accessing sports club information in WA. The QCI would use local context to add related terms more likely to directly match data record descriptions, such as 'AFL', 'football', and 'Fremantle'.

Local data context can include lexical (e.g. terminological ontologies) and spatial (e.g. extent) information. This context is related to data content, and may be preprocessed or extracted on-the-fly by different agent types.

Agent framework

Three essential behaviours are included in every agent: GetCapabilities to describe the agent, GetWeight to estimate the relevance of its data to a query, and Process to search data content. Each agent can be accessed via requests sent to a Web service unique to the agent.

The GetCapabilities behaviour is important for service discovery, as discussed in Section 2.5.1, and does not require any query or contextual input parameters. It returns a description of the agent, including what it is capable of and how to use it. The operation name is taken from the mandatory action of Open Geospatial Consortium (OGC) standards including WFS and WMS.

The GetWeight action uses data context to estimate the 0.0...1.0 relevance of the local data resources as a whole to an information request. Methods used to estimate this weight can vary for different agent types, as described in Sections 5.3 and 6.2.1.

Process triggers the primary action of the agent; in this case, a search operation. The OGC WPS standard uses an equivalent operation name of Execute to trigger an action (OGC, 2007). Where possible, Process will implement a Deep Web search to retrieve relevant records. Where a deep search is not feasible, metadata about agent data resources, including relevance weight, are returned without record detail.

Both the GetWeight and Process tasks adapt an input request via the QCI. Local ontologies and specialist terminologies within a QCI serve a similar purpose to the semantic translators in the architecture proposed by Zhou (2010), although the latter were designed to interact with each other in a single system: a geospatial portal. Encapsulating information within an agent facilitates reuse across multiple systems, with each agent working as a black box.

Resources accessed by a search agent are not restricted by format or technology, as the logic required for data access is embedded within the agent. Examples of data sources include databases, document collections, Web services, or other search agents, i.e. a hierarchical search agent.

A hypothetical use case for a hierarchy of search agents is illustrated as a Secure Search Service in Figure 6.2 (label c). Data with access restrictions — such as confidential medical records or data provided for a fee — cannot be released without confirmed author-

ity. Using a search agent, a request for confidential data without the necessary access credentials could still receive information about the dataset, such as name, relevance estimate, and contact person, whilst private records are withheld by the agent.

Faceted search and recommender systems (Section 2.4) are two examples of IR techniques that can make use of search agents. Although not implemented in the case study application, the agent framework is adaptable to these and other search tasks. For example, agents can optionally specify a type key for each result or record item that can be used to organise combined results by themed facets. See Appendix D for an example of the type key in results from an agent.

6.4.1 Implementation

The design was implemented in Python, using the Django[71] Web framework and associated GeoDjango[72] module. Each type of data agent was implemented as a Python class — more specifically, as a Django model.

A search agent template was designed that could be extended for specific purposes and data resource types. DataAgent is the superclass of all search agents, and includes functionality for minimal requirements including recognition of required parameters and request types.

Handler classes were designed to add functionality for common tasks, such as working with geospatial data. A subclass of DataAgent can include one or more handlers using multiple inheritance, as illustrated in Figure 6.4 for a specialised agent (WFSAgent). Handlers SpatialHandler and TextQueryHandler were implemented in the case study; the latter includes functionality to extract place names from text queries.

A hierarchical agent calls upon one or more search agents as data sources. As all agents respond to the three main request types and return results following a similar format, source agents can be of different types. This manager-agent/source-agent relationship was implemented with a DataAgentSource class, as shown in Figure 6.4, to allow for many-to-many relationships between agent types: an agent can be both a data source and a manager of other agents.

[71] https://www.djangoproject.com, accessed 9/2/2017.
[72] https://docs.djangoproject.com/en/1.10/ref/contrib/gis/, accessed 9/2/2017.

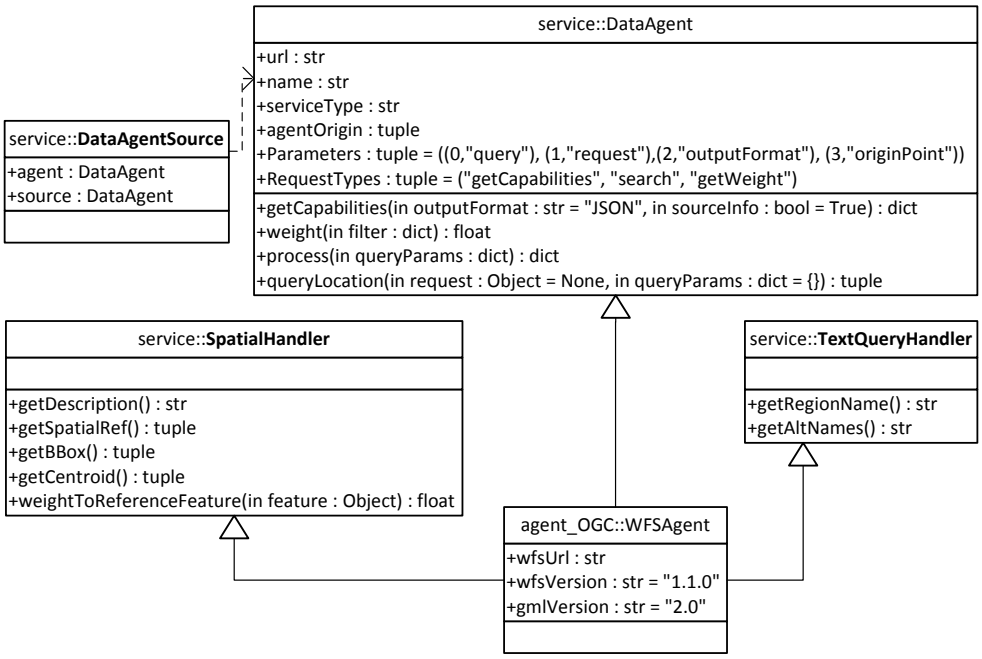


Figure 6.4: Unified Modelling Language diagram of DataAgent, DataAgentSource, SpatialHandler, TextQueryHandler, and WFSAgent classes.

Python code implementing DataAgent models and handlers is listed in Appendix E.

Input parameters for a data agent are defined using a key-value pair (KVP) mechanism, where each agent type recognises a set of known text keys that can be associated with input values. This is illustrated in Figure 6.4 by the queryParams and filter inputs to the DataAgent methods process and weight, respectively.

All search agents include a process method that recognises at least four input parameter keys: request, query, outputFormat, and originPoint. These four keys, where present in a request, are recognised as valid by all agent types, although their values may be ignored.

The request parameter specifies the required task, and hence is closely linked to the list of available actions for an agent. All data agents respond to a minimum of three request types: getCapabilities, getWeight and search.

Of the other input parameters common to all data search agents, query represents a text-based IR question (e.g. `"bus stops near Perth airport"`), outputFormat specifies the format to use for results (e.g. 'JSON' or 'XML'), and originPoint is a point location representing the query origin as a latitude, longitude coordinate pair.

Specialised data agents can add to the list of parameters to cater for specific actions, such as recognising a bbox or geom key for spatial filtering. Further examples of data agent input parameters are listed in Appendix C.

Responses from agents also use the KVP model, pairing value elements with known keys. Every agent response includes at least request (listing all input parameter values from the initial request), service (naming the Web service accessing the agent), date (of the request), and results (a list of results, which can be empty). Responses can also include an error key to report problems encountered when searching a data resource.

The results key is linked to a list of dictionary entries, one per agent data source, each containing at least a dataset name. Each results entry contains sub-elements, such as weight for the 0.0...1.0 likelihood of dataset relevance, as well as the required name.

Where individual records are available, they are returned in a list within a result entry, using a records key. Each record requires an associated label, but otherwise its contents can vary depending on the agent type. Examples include geometry for spatial feature coordinates, weight for a 0.0...1.0 estimate of record relevance in its result set, and type for categorisation of records, e.g. for plotting or mapping.

Specialist data agents can add extra output keys to augment the default information, such as origin to document the location that proximity was judged against. See Appendix D for an example JavaScript Object Notation (JSON) response from a DataAgent source.

Web services

Discovery and use of Web services is simplified if services are consistent in their inputs and result formatting (Section 2.5.1). DataAgent instances can be accessed programmatically via an Application Programming Interface (API) or online via RESTful Web services with an OGC-style GetCapabilities request.

Each agent was assigned a Django view, with RESTful parameters extracted from a GET or POST HyperText Transfer Protocol (HTTP) request. The case study implementation returns results in JSON format, although the system has been designed to

allow for alternative formats such as eXtensible Markup Language (XML). Example service requests and responses for OGC and proposed services are described in Gulland *et al.* (2016).

Once the agent retrieves results (and/or errors) from its data sources, they are passed back to the application or service that sent the original request.

Where the results element in an agent response does not contain an empty list, it documents each of its data sources with, optionally, weight keys estimating relevance. A result entry can also include a records element listing individual data records, each containing a label plus other, non-compulsory elements such as weight and geometry.

Each agent uses the context of its data, including technical requirements, to transform request inputs into a form recognisable by its data source or sources. For instance, assuming a WFS data source, valid request syntax for an OGC-compliant WFS depends upon its specific version and capabilities — the WFS version 2.0 count key is equivalent to the version 1.1 maxFeatures, and filters such as Within are not implemented by all WFS. Listing 6.1 is an example of a WFS request to retrieve bus stop features within a Geographic Markup Language (GML) bounding box geometry, tailored for a specific WFS instance.

Listing 6.1. RESTful GET request for a Web Feature Service.

```
http://www.busexample?service=WFS
 &version=1.0.0
 &request=GetFeature
 &typeName=busexample:busstops
 &maxFeatures=100
 &outputFormat=application/json
 &filter=<Filter xmlns:gml="http://www.opengis.net/gml">
    <Within><PropertyName>the_geom</PropertyName>
        <gml:MultiPolygon srsName="EPSG:28350">
        <gml:polygonMember><gml:Polygon>
        <gml:outerBoundaryIs><gml:LinearRing>
        <gml:coordinates>
            389340.34,6463999.38  388340.34,6463999.38
            388340.34,6464999.38  389340.34,6464999.38
            389340.34,6463999.38
         </gml:coordinates>
         </gml:LinearRing></gml:outerBoundaryIs>
         </gml:Polygon></gml:polygonMember>
         </gml:MultiPolygon>
    </Within></Filter>
```

Specialist search agents

WFSAgent is an example of a specialist agent type that extends the DataAgent template by adding tailored behaviours and extra input parameters. This agent implements the SpatialHandler and TextQueryHandler handlers, as illustrated in Figure 6.4.

A WFSAgent can derive and execute a request such as Listing 6.1 by discovering the capabilities, version, and restrictions of its WFS data source to compile a valid request. The equivalent WFSAgent request is shown in Listing 6.2, requesting the same information as Listing 6.1.

Listing 6.2. RESTful GET request for a WFSAgent.

```
1 http://.../ogc/wfs?service=wfs
2  &request=search
3  &outputFormat=JSON
4  &query=bus+stops+in+West+Perth
5  &bbox=115.818593,-31.954588,115.829058,-31.954490
```

Note that Listing 6.2 includes common input keys request, query, and outputFormat, although some (query, in this case) can be unused. It also includes a specialist input: bbox.

Specialist search agents implemented in the case study were:

- WFSAgent: Connects to a single WFS.
- BoundaryAgent: Searches for named regions from one or more online DataAgent sources.
- NearbyAgent: Creates an expanded search area from an input boundary geometry, then returns features in it from a DataAgent source, specified as a URL input parameter.
- PublicTransportAgent: Finds spatial features of interest within a bounding box or one or more boundary geometries.

BoundaryAgent and PublicTransportAgent agents are hierarchical, in that they can define one or more spatial search agents as data sources. The DataAgent template includes a GetSourceResults function that passes query parameters on to its source agents and

assembles results from them. For non-hierarchical agents, this method always returns an empty list.

Specialist agents can add output keys and/or request types beyond compulsory keys. The complete set of input parameters applicable to a search service can be retrieved with a GetCapabilities request.

Listing 6.3 shows a code extract from the BoundaryAgent model illustrating how new input parameters are added to the standard set of input keys. Note the use of master parameters inherited from DataAgent (lines 4-8).

Listing 6.3. Extract from BoundaryAgent model code highlighting extension of input parameters from the DataAgent template.

```
1  from service.models import DataAgent, TextQueryHandler
2  from service.spatial import SpatialHandler
3  class BoundaryAgent(DataAgent, TextQueryHandler,
       SpatialHandler):
4      masterParams = DataAgent.Parameters
5      n = len(masterParams)
6      (BBOX, REGION_TYPE, NAME, GET_BBOX, RESULT_TYPE) \
7          = range(n, n+5)
8      Parameters = masterParams + (
9          (BBOX, "bbox"),              #as text string
10         (REGION_TYPE, "regionType"), #re.g. suburb, SA3
11         (NAME, "name"),         #name (full/partial) of region
12         #specify how to return shapes:
13         (GET_BBOX, "getBBox"), #T: as bbox, F: as polygons
14         (RESULT_TYPE, "resultType") #combine: list, union,
               ...
15     )
```

Listing 6.4 shows a RESTful POST query request for a BoundaryAgent. The equivalent Python API call is:

```
anAgent.process({'request':'search', 'outputFormat':'JSON',
    'query':'bus stops in West Perth', 'name': 'West Perth',
    'resultType': 'nearest'})
```

Listing 6.4. RESTful GET request for a BoundaryAgent.

```
1  http://.../spatial/boundary?service=boundary
2      &request=search
3      &outputFormat=JSON
4      &query=bus+stops+in+West+Perth
5      &name=West+Perth
6      &resultType=nearest
```

The resultType parameter (line 6 in Listing 6.4) is a specialist key not defined in DataAgent. It specifies the behaviour if more than one valid spatial record is found — in this case, the feature that is judged to be nearest the query or user origin. If neither is available, the default home location of the search agent is used instead. The default behaviour for a BoundaryAgent where this parameter is not specified is to produce a single multipolygon feature from all matching records.

The KVP mechanism of query parameters and results allows the flexibility for further extensions in the future, for example to allow for user logins for confidential data, or to return data in a format suitable for visualisation (Moncrieff & West, 2013). However, KVP is not the most efficient communication medium for large data volumes (Perera *et al.*, 2014). The agent framework has been designed to allow for alternative response formats.

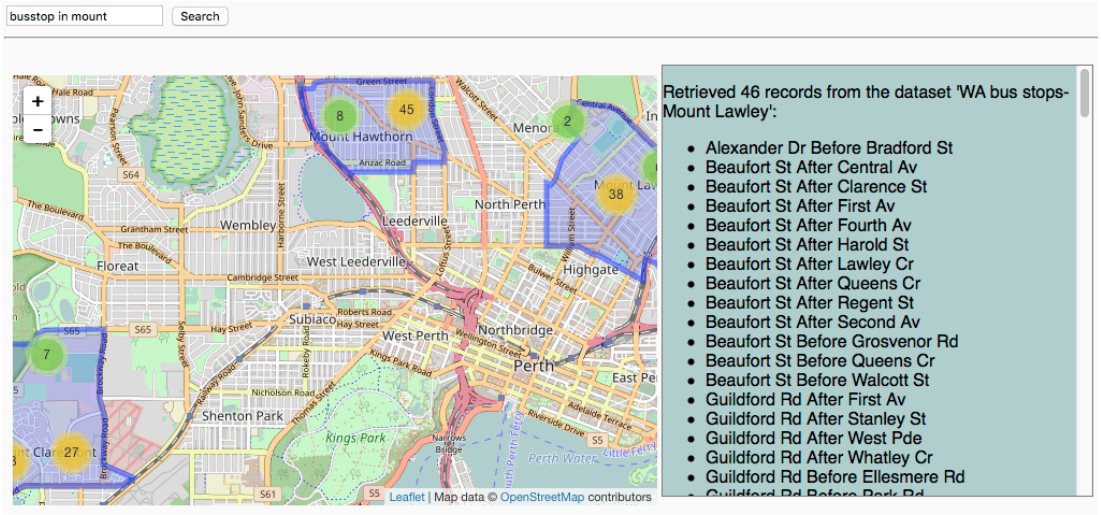
The aim for DataAgent and agents that inherit from it was to encapsulate search behaviours with data context and context, whilst maintaining a consistent, simple interface for online access. To test this, a Web application was developed to transform a query input into a visualisation of results from multiple agents.

6.5 Case study

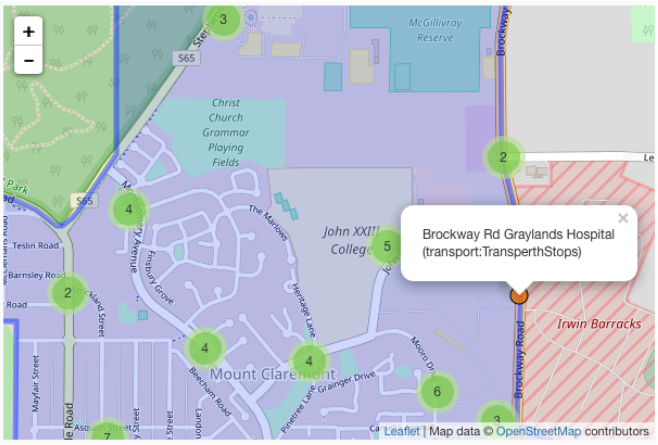
A Web application was designed to find bus stops within or near a named region using two datasets, both in the region of Perth, WA. Each source is a single WFS, hosted as a GeoServer layer: ABS suburb regions (2011)[73], and Transperth bus stop point locations[74].

The graphical user interface (GUI) includes a text box input, start button, and display areas for text results and a map, implemented using the Leaflet[75] JavaScript mapping library with an OpenStreetMap (OSM) background map, as illustrated in Figure 6.5.

[73]Suburb: http://www.abs.gov.au, accessed 31/10/2014.
[74]Bus: http://www.transperth.wa.gov.au/About/Spatial-Data-Access, accessed 6/3/2015.
[75]http://leafletjs.com, accessed 16/2/2017



(i) Mapped suburbs and bus stop counts (as green and orange cluster markers, where orange markers signify a greater number of bus stops). Note that the list of bus stop names at right is partial — scrolling down reveals results from multiple suburbs.



(ii) Zoomed-in view showing the pop-up label on a bus stop point.

Figure 6.5: Screenshot from a demonstration Web application showing mapped results from a search for bus stops in regions (partially) named 'Mount'.

6.5.1 Extracting context from a query

The case study search application uses a text parser to extract three elements from a query: named ROI, feature type, and spatial operation elements. It searches for one of two patterns, as described in Gulland *et al.* (2016):

1. `<feature type> <spatial operator> <region name>`

   e.g. `"hotels near Perth Scotland"`.

2. `<region name> <feature type>` or `<region name>`

   e.g. `"New South Wales hotel"` or `"Perth Scotland"`.

The first pattern type recognises spatial operator names `["near", "nearby", "near to", "close to", "in", "within", "inside"]`, although this can be extended via terminological resources such as ontologies or thesauri.

The second pattern type retrieves multiple toponyms for testing to allow for n-gram names, e.g. `["New", "New South", "New South Wales", "New South Wales hotel"]`. As no spatial operator is specified, a simple containment operation is assumed.

The application determines a user origin from the IP address of the Web request, where available, using MaxMind GeoIP[76]. A blank location is returned if an IP address cannot be extracted.

6.5.2 Search process

As the focus was on testing the efficacy and coordination of multiple search agents, rather than their discovery, Web service URLs were preset in the case study application. Instances of four agent types were used:

- Two WFSAgent instances that can each return spatial features via deep search. Each connects to a single WFS hosted as a GeoServer layer: ABS suburbs and Transperth bus stops, respectively.
- PublicTransportAgent, with the bus stop WFSAgent as a data source.
- BoundaryAgent, with the suburb WFSAgent as a data source.
- NearbyAgent, with no recorded data sources. Instead, a source URL (the bus stop WFS) is specified as an input parameter. From this, it will retrieve or build a search agent to manage searches on this online data source.

The application coordinator uses a sequential process:

1. Parse the query text to extract a target type, spatial operator and place name, as outlined in Section 6.2.
2. Search for one or more ROIs by name, using a BoundaryAgent.

[76] https://www.maxmind.com/en/geoip2-services-and-databases, accessed 14/1/2017.

3. Pass ROI geometries to a PublicTransportAgent and retrieve any spatial features within the region(s).

The region name found in the parsing step was processed to provide upper, lower, and title case alternatives. This is also the point where application-level context can be added, such as alternative feature types from search logs or place names via a spatial contextualiser (Figure 6.2, label 1A).

The query feature type was compared to a set of valid targets (`["station", "bus-stops", ...]`). If no match was found, an error message was added to the output and the search operation ended.

In step 2), a likely label attribute name for labels was extracted from the WFS source. This attribute location process was implemented as a method (matchAttribute) in WFS-Agent. Using the appropriate attribute name, WFS requests were created to extract spatial records where the attribute value matched requested region names, at least in part. Matching labels were also attached to records in the set of results.

Each matching boundary record was added by the WFSAgent to a result set, along with its matching label, and returned to the coordinator. If no region matches were found, the search operation ended with an error message attached to the output.

The coordinator sent the geometries from step 2) as input parameters to the Public-TransportAgent in step 3). This agent used its WFSAgent(s) to retrieve spatially relevant records within a ROI. Each point record found was labelled with the region name plus its own name (if any existed).

When a Near spatial operator was detected by the coordinator in step 1), a different third step was used. In this case, the region geometries and geometry types (e.g. GeoJ-SON or GML) from step 2) were sent by the coordinator as inputs to the NearbyAgent instance, along with the URL of a WFS. The agent retrieves or, if necessary, constructs a WFSAgent source from the input URL. If a distance (plus optional units) input is received by the NearbyAgent, the input region(s) are buffered by that distance. If not, the agent assigns its own buffer distance. The enlarged region(s) are then used to retrieve point records from the WFSAgent as for the Within process.

The final coordinator task is to display the results to the user. In the case study Web application, bus stops and regions found are mapped, with a pop-up label for each

feature, and bus stop names are listed by suburb (Figure 6.5). Error or warning messages returned by agents are displayed on the Web page as illustrated in Figure 6.6.

6.5.3 Results

Screenshots of results from the query `"busstop in mount"` are shown in Figure 6.5. The map displays suburbs with names containing the text 'mount' (regardless of case): Mount Lawley, Mount Pleasant, Mount Claremont, and Mount Hawthorn. It also maps any bus stops within those suburbs and lists them by name, collected under the containing suburb name. The text to the right of Figure 6.5i shows a partial list of bus stop names within a scrollable box, organised by containing suburb.

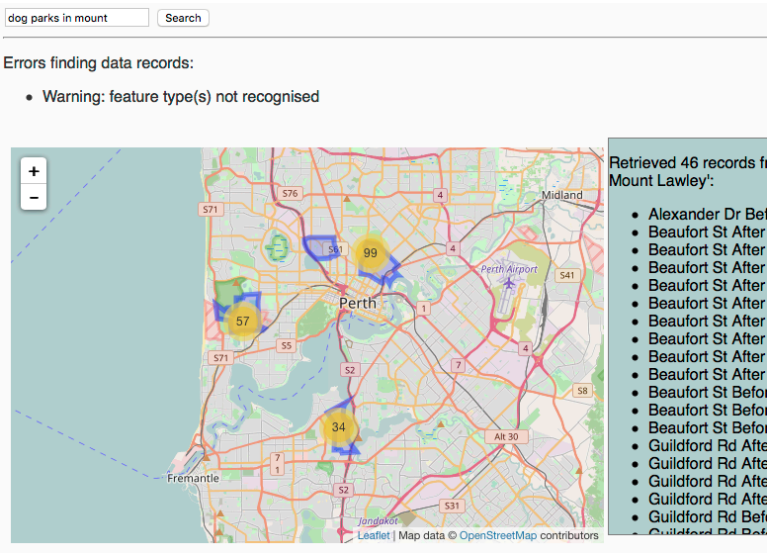
Where query regions are found but the query theme is not recognised, the default target (bus stops) is assumed and displayed beneath a warning message (Figure 6.6i). A map of Australia with no added features is displayed when no query regions are found (Figure 6.6ii).

A second source was added to the PublicTransportAgent agent for further testing, as described in Gulland *et al.* (2016). This source was a WFSAgent linked to a public Geoscience Australia WFS documenting Australian waste management facility locations. This dataset was used as a proxy for bus stop locations as it contained point features in the same approximate area as other test datasets.

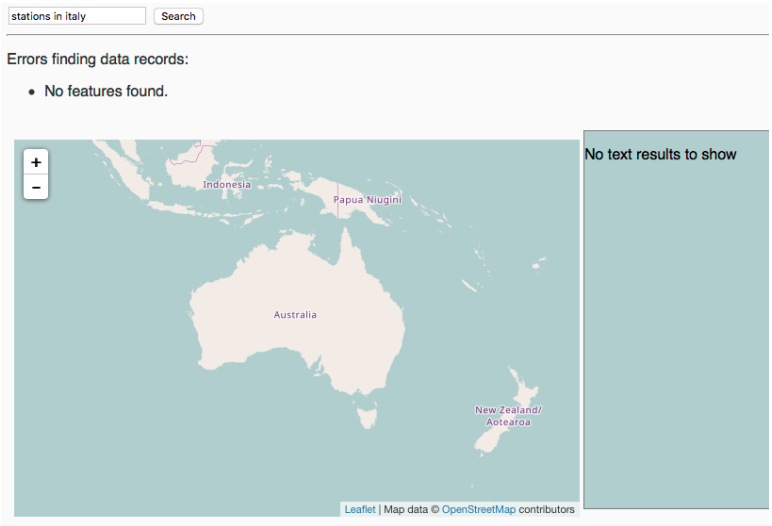
The Geoscience Australia WFS had limited spatial filter capabilities (although a more recent version[77] provides OGC spatial filters including Within, for WFS versions 1.0 and 1.1). It also supported XML only, whilst the prototype WFSAgent expected WFS results in JSON format. Hence the WFSAgent result entry for this source contains an empty record list, and it also adds to the optional error list. Importantly, this did not disrupt the search coordinator, which used results from the other point data source and ignored the blank result list.

The coordinator application called upon agents through general agent framework interfaces and thus needed little or no knowledge of local data context and technologies. This matches the aim for agents to respond to an IR request in a common format.

[77] Geoscience Australia waste management sites: http://services.ga.gov.au/gis/services/Waste_Management_Facilities/MapServer/WFSServer, accessed 16/2/2017.



(i) Result where the target type (dog parks) is unrecognised. The default type (bus stops) is assumed and mapped, but a warning message is displayed at the top of the page.



(ii) Result where the region name (Italy) is unrecognised. A warning message is displayed at the top of the page.

Figure 6.6: Screenshot from a demonstration Web application showing warning messages for unrecognised query components.

Results also demonstrate support of data record retrieval via the Deep Web, as spatial features are extracted from within WFS sources that regular Web browsers do not index.

6.6 Summary and discussion

In this chapter, data search agents were proposed to manage contextual search requests for specific data resources. The DataAgent framework was designed to encapsulate search tasks with local data content and context and allow IR query responses to return a 0.0...1.0 relevance estimate for the dataset and/or individual data records.

By encapsulating search actions into agents, a coordinator can send the same request to multiple data sources without requiring prior knowledge of their context or technical requirements. Communication with data search agents was simplified with consistent formatting of inputs and outputs: in the case study, using key-value pair input parameters and RESTful Web services. Example parameters and output framework (in JSON) are listed in Appendix C and D, respectively.

The KVP mechanism used in the prototype implementation to describe input parameters and define response elements allows for flexibility in agent design and managing different output formats (such as XML and JSON). However, it is not well-suited to large sets of data and could reduce efficiency for complex or large volumes of input parameters (Perera *et al.*, 2014). This warrants further investigation into alternative formats.

The proposed framework introduces other potential disadvantages. Limiting agents to a single task (search, with optional record retrieval) reduces their abilities as part of a larger, autonomous system. This trade-off was judged to be acceptable — given that obtaining data is the first step of complex processing tasks, search agents could be utilised by other, more complex Web services.

Similarly, encapsulating context with search agents can limit comparison across sources. To offset this, individual agents can access public ontologies and other resources with wider knowledge domains. The spatial overlap index in Section 5.2.1 is one such example. Also, hierarchical search agents may define broader contextual resources, such as organisation- or domain-wide terminologies.

Coordination of contextual search across multiple search agents was tested by implementing a case study Web application. The case study demonstrated the three search stages by extracting query elements, linking directly to known search agents in a logical sequence, and extracting geometry and labels from result records for display on an

online GUI. Any error messages returned by agents, such as missing data, triggered the display of feedback for the user.

Agents capable of deep search, including WFSAgent, successfully accessed data records and returned them within a query result. Hierarchical agents that use other search agents as sources of data were also successfully demonstrated with BoundaryAgent and PublicTransportAgent, each using WFSAgent sources.

Advantages of the agent framework include the ability to return an estimated level of confidence in the relevance of dataset (or record) relevance, tailoring of techniques to specialised search tasks and resource idiosyncrasies, and hiding implementation detail to simplify coordination.

Future potential

The DataAgent framework and search coordinator approach was designed for flexibility, to enable agent reuse by different coordinators for different needs. New search agent types can build upon the template described in this chapter to allow for specialised searches and inclusion of other types of context from both user and data, and still respond to standard IR requests.

The existing getCapabilities facility of all search agents, following an OGC service design, allows coordinators to check for availability of specialised inputs. In the future, additional input parameters can be defined, for example to allow for other types of context such as spatiotemporal resolution, time epoch or secure access. A coordinator would need to extract this context from queries, user actions or profiles, or search histories.

Faceted search is an example of an application that could make use of the optional type key in an agent result to group related items together for display to the user. Any items with no allocated type would be added to a single, miscellaneous group.

Data records from agents capable of deep search can also be chained together as part of an orchestrated processing service (Rautenbach *et al.*, 2013). Relevance weights from results and their records would be useful in a visualisation of confidence in contributing data.

7 Conclusions and recommendations

7.1 Conclusions

This thesis presented an approach to automated semantic indexing of data sources by their relevance to a user query, using contextual information from both user and data. This caters for on-the-fly ranking of datasets without requiring prior knowledge about the data.

Research objective 1

> Automate the production of a weighted ontology of terms (that is, a network of interconnected terms) for a data resource, incorporating relatedness strengths between terms. As part of this objective, crowd-sourced text resources will be investigated as sources of term-relatedness weights.

The first research objective was investigated in two stages: firstly term-relatedness measures and secondly incorporating such weights into unweighted and weighted ontologies of terms.

An unweighted, composite term-relatedness measure Combined Resources Weight (CRW) was developed for general vocabularies, with individual weights determined from crowd-sourced (Google and Wikipedia) and other public text resources. CRW was demonstrated to predict ground truth term-relatedness weights more successfully than its individual components (Figure 3.6), although with a low true positive rate at a threshold weight of 0.5 (Table 3.3).

The CRW measure was found to be suitable for building ontologies of related terms by selecting and linking paired terms with term-relatedness weights over a threshold value. Combining measures was shown to smooth out ill effects from individual components, and the combination was successfully extended with specialist measures (such as UMLS::Relatedness for biomedical terminology) to accommodate mixed-purpose vocabularies.

Supervised machine learning models were tested for classification of related term pairs and estimation of link probabilities, using CRW components. Although CRW values

and other weighted combinations were judged to be acceptable measures for use in a weighted ontology, probabilistic weights (such as calculated from a logistic regression model or extracted from a trained Multilayer Perceptron (MLP) model) are recommended. This is due to their potential for discovering new relationships via multiplication of interconnected link probabilities.

Developing terminological resources for new data sources with supervised machine learning approaches requires production of a training subset of paired terms labelled with human judgements of term-relatedness. This imposes a processing cost. Where this is impractical, the unweighted CRW measure can be used without training, although at a possible cost of lower reliability for local context.

False positives were seen as a greater issue within a term ontology than false negatives. An approach to reducing such links is to prune or decrease the weight of such links. Clustering terms by term-relatedness weights, whilst not reliably differentiating by topics, found outlier terms without links, and these have potential for discovering spurious term pairs.

Research objective 2

> Rank dataset records relative to an information request, using lexical context including weighted ontologies of terms.

Lexical ranking of datasets was achieved by creating virtual queries that added terms strongly related to the original text, based on term-relatedness weights in a weighted ontology. Virtual queries were then used in natural language processing models to rank documents — that is, data records — within a dataset.

Target documents were ranked highly in search result sets, though lower on average than unaugmented queries and hence virtual queries resulted in lower precision. However, the larger result sets and stronger query-similarity weights lend themselves to data exploration. It also provides a wider base for combination with other semantic weights, such as spatial proximity.

Research objective 3

> Combine spatial context from user, query, and data source locations, including feature-level details from the Deep Web, to inform a spatial search.

To inform a spatial search from combined user and data contexts, elements of spatial context, including place names, extent, and representative origin or anchor points, were investigated for utility in semantic indexing.

A polygon overlap index was developed to assist with region matching and ranked spatial overlap matches. This was useful to speed up area comparisons although, like all such preprocessed indexes, it is restricted to the datasets loaded into it.

Proximity measures were calculated from a query target to features within a dataset to rank individual records. A proximity measure for the dataset as a whole can be calculated as average feature proximity or by applying other methods such as the proportion of features above a threshold proximity value. This caters for complex dataset footprints not available through metadata such as bounding boxes.

An exponential decay function was applied to distances between query and data features, in order to rank potential results. Calculations of context-specific decay constants was shown to be adaptable to query context, via region of interest (ROI) size, and data context, via record-level access of contents and a metric for feature density.

ROI-based decay constants were judged to be useful general applications of query context, particularly where a precise user location is not available. Dataset-specific decay constants, such as feature density-based measures, are considered part of data context to be encapsulated with data content and search behaviours and can be further tuned to idiosyncrasies of datasets.

Research objective 4

> Develop software agents that, in response to an information request using a common format, link data and user context to return a confidence level in the relevance of the data source. These agents need to be able to use data record level detail rather than relying solely on public metadata (as used in current spatial searches, such as in Web Feature Services (WFS)). The

search agent template must also allow for retrieval of a list of records where this is feasible.

Finally, to meet objective four, a search agent framework was developed that could be tailored to the capabilities, requirements, and context of different data source types. A general agent was designed with a set of expected behaviours and input parameters that were inherited by all agent types. These included the ability to pass a query on to other source agents, catering for a hierarchical structure of agents. The agents also accommodate reporting of confidence levels in the relevance of dataset and data records alongside search results.

Methods for combining two or more semantic measures were proposed. This caters for scenarios including comparison of results from a multi-step process over different datasets, and consideration of multiple aspects of context such as spatial and lexical. The latter case enables comparison of data sources with different capabilities, for example where not all datasets being considered are able to access individual spatial features but all can match text.

The template agent is designed to return a single confidence level for its data, but this can be extended to allow specialist agents to also return component weights separately so that a coordinator can apply its own priorities more precisely when comparing results.

The agent design includes the ability to return a list of individual records, where feasible for the data source. It also incorporates an optional ranking weight per record within formatted output (Appendix D). Methods for estimating $0.0 \ldots 1.0$ confidence levels, including document similarity (Chapter 4) and spatial proximity (Chapter 5) weights, can also be applied directly to data records.

Search coordination was tested with a set of specialised agents (Chapter 6). These demonstrated access through a Web service request, with individual agents combining query and data context, including local technical requirements and capabilities, proximity tests and/or text searches. A subset of agents returned a list of records along with information about the data source.

7.2 Recommendations

Automated semantic indexing of datasets based on data and user context as outlined in this thesis is a framework that can be built upon in future work.

The extensibility of the CRW measure of term-relatedness was tested on specialised medical terminology, using existing relatedness measures (UMLS similarity and relatedness). For less well-researched domains, existing measures are unlikely to be available. A useful further area of research would be development of methods to automatically estimate term link weights within a specialist text collection. One possibility is to adapt measures based on the crowdsourced Google and Wikipedia resources for use in specialist search engines and wikis.

A local toponym resource, built and maintained in a similar manner to the weighted ontology of terms described in Chapter 4, would add useful spatial context for responding to semantic search requests. Dataset place names can be sourced from descriptive metadata and record attributes, with related place names drawn out by natural language processing (NLP) methods such as named entity recognition.

Distance decay constants for spatial proximity measures can be further tailored to local context of specialised datasets, for example by incorporating expert knowledge or a spatial density metric dependent on themed feature types within a dataset.

This project has provided a basis for automatic combinations of two or more semantic measures in a relevance weight for a dataset. This premise could be tested further by incorporating more measures within the combined weight. Examples include temporal measures (e.g. by currency or closeness to a specified time epoch) and cost for paid-access data.

The developed search agent framework was designed to be extensible, allowing for additional search capabilities and access to more data source types. This adaptability can be further tested with additional agent types, for example to implement the currently-theoretical capability of indexing datasets with protected data, such as private records or paid-access data.

This would also permit further testing of coordination of different agent types. For example, comparing rank weights for a temporal information retrieval (IR) request

where a time input parameter was recognised by some, but not all, agents receiving the request.

References

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. 1999. Towards a Better Understanding of Context and Context-Awareness. *Pages 304–307 of: International Symposium on Handheld and Ubiquitous Computing HUC 1999.* Springer-Verlag.

Adams, B., & McKenzie, G. 2013. Inferring Thematic Places from Spatially Referenced Natural Language Descriptions. *Pages 201–221 of:* Sui, D. Z., Elwood, S., & Goodchild, M. F. (eds), *Crowdsourcing Geographic Knowledge: Volunteered Geographic Information (VGI) in Theory and Practice.* Springer.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., & Soroa, A. 2009. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. *In: Proceedings of North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT).*

Arun, R., Suresh, V., Veni Madhavan, C. E., & Narasimha Murty, M. 2010. On finding the natural number of topics with Latent Dirichlet Allocation: some observations. *Pages 391–402 of: 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2010).* Springer Berlin Heidelberg.

Bachi, R. 1962. Standard Distance Measures and Related Methods for Spatial Analysis. *Papers in Regional Science*, **10**(1), 83–132.

Baeza-Yates, R. A., & Tiberi, A. 2007. Extracting Semantic Relations from Query Logs. *Pages 76–85 of: 13th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD).* ACM.

Ballatore, A., Wilson, D. C., & Bertolotto, M. 2013. A Survey of Volunteered Open Geo-Knowledge Bases in the Semantic Web. *Chap. 5, pages 93–120 of:* Pasi, G., Bordogna, G., & Jain, L. C. (eds), *Quality Issues in the Management of Web Information.* Intelligent Systems Reference Library, vol. 50. Springer-Verlag.

Bast, H., Buchold, B., & Haussmann, E. 2015. Relevance Scores for Triples from Type-Like Relations. *Pages 243–252 of: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Beard, K., & Sharma, V. 1997. Access to geographic concepts in online bibliographic files: effectiveness of current practices and the potential of a graphic interface. *International Journal on Digital Libraries*, **1**(2), 153–160.

Bergman, M. K. 2001 (September). *The Deep Web: Surfacing Hidden Value*. Tech. rept. BrightPlanet Corporation.

Berners-Lee, T. 1989, updated 1998. *Information Management: A Proposal*. https://www.w3.org/History/1989/proposal.html. Accessed 2017-06-24.

Berners-Lee, T., Hendler, J., & Lassila, O. 2001. The Semantic Web. *Scientific American*, **284**(5), 34–43.

Best, B. D., Halpin, P. N., Fujioka, E., Read, A. J., Song, S. Q., Hazen, L. J., & Schick, R. S. 2007. Geospatial web services within a scientific workflow: Predicting marine mammal habitats in a dynamic environment. *Ecological Informatics*, **2**, 210–223.

Bhogal, J., Macfarlane, A., & Smith, P. 2007. A review of ontology based query expansion. *Information Processing and Management*, **43**(4), 426–435.

Bilhaut, F., Charnois, T., Enjalbert, P., & Mathet, Y. 2003. Geographic reference analysis for geographic document querying. *Pages 55–62 of: Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References*, vol. 1. Stroudsburg, PA USA: Association for Computational Linguistics.

Blanco, R., Halpin, H., Herzig, D. M., Mika, P., Pound, J., Thompson, H. S., & Tran, T. 2013. Repeatable and Reliable Semantic Search Evaluation. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, **21**, 14–29.

Blei, D. M., Ng, A. Y., & Jordan, M. I. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, **3**(Jan), 993–1022.

Bobed, C., & Mena, E. 2016. QueryGen: Semantic Interpretation of Keyword Queries Over Heterogeneous Information Systems. *Information Sciences*, **329**, 412–433.

Bobillo, F., & Straccia, U. 2016. The fuzzy ontology reasoner *fuzzyDL*. *Knowledge-Based Systems*, **95**(1 March), 12–34.

Bogdanović, M., Stanimirović, A., & Stoimenov, L. 2015. Methodology for geospatial data source discovery in ontology-driven geo-information integration architectures. *Journal of Web Semantics*, **32**, 1–15.

Bone, C., Ager, A., Bunzel, K., & Tierney, L. 2014. A Geospatial Search Engine for Discovering Multiformat Geospatial Data Across the Web. *International Journal of Digital Earth*, **9**(1), 47–62.

Bradley, N. A., & Dunlop, M. D. 2005. Toward a Multidisciplinary Model of Context to Support Context-Aware Computing. *Human Computer Interaction*, **20**(4), 403–446.

Broder, A. 2002. A taxonomy of web search. *ACM SIGIR Forum*, **36**(2), 3–10.

Budanitsky, A., & Hirst, G. 2006. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, **32**(1), 13–47.

Büttcher, S., Clarke, C. L. A., & Cormack, G. V. 2010. *Information Retrieval: Implementing and Evaluating Search Engines.* MIT Press.

Cai, G. 2007. Contextualization of Geospatial Database Semantics for Human-GIS Interaction. *Geoinformatica*, **11**(2), 217–237.

Caliusco, M. L., & Stegmayer, G. 2010. Semantic Web Technologies and Artificial Neural Networks for Intelligent Web Knowledge Source Discovery. *Chap. 2, pages 17–37 of:* Badr, Y., Chbeir, R., Abraham, A., & Hassanien, A.-E. (eds), *Emergent Web Intelligence: Advanced Semantic Technologies.* London: Springer.

Carterette, B., Kanoulas, E., & Yilmaz, E. 2012. Evaluating Web Retrieval Effectiveness. *Chap. 5, pages 105–137 of:* Lewandowski, D. (ed), *Web Search Engine Research.* Library and Information Science. Emerald Group Publishing Limited.

Çelik, D., & Elçi, A. 2005. A semantic search approach: Finding appropriate semantic Web services based on user request term(s). *Pages 675–687 of: 3rd International Conference on Information and Communications Technology.* IEEE.

Çelik, D., & Elçi, A. 2006. Discovery and Scoring of Semantic Web Services based on Client Requirement(s) through a Semantic Search Agent. *Pages 273–289 of: Computer Software and Applications Conference (COMPSAC '06)*, vol. 2. IEEE.

Cesarano, C., d'Acierno, A., & Picariello, A. 2003. An Intelligent Search Agent System for Semantic Information Retrieval on the Internet. *Pages 111–117 of: 5th ACM International Workshop on Web Information and Data Management (WIDM '03).* ACM.

Chang, C.-C., & Lin, C.-J. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**(3).

Chen, T., Zhang, N. L., Liu, T., Poon, K. M., & Wang, Y. 2012. Model-based multidimensional clustering of categorical data. *Artificial Intelligence*, **176**(1), 2246–2269.

Cheung, C. K., Shi, W., & Zhou, X. 2004. A Probability-Based Uncertainty Model for Pont-in-Polygon Analysis in GIS. *Geoinformatica*, **8**(1), 71–98.

Chun, S. A., & Warner, J. 2008. Semantic Annotation and Search for Deep Web Services. *Pages 389–395 of: 2nd IEEE Conference on Enterprise Computing, E-Commerce and E-Services*. IEEE.

Cilibrasi, R. L., & Vitanyi, P. M. B. 2007. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, **19**(3), 370–383.

Cortes, C., & Vapnik, V. 1995. Support-Vector Networks. *Machine Learning*, 273–297.

Crabtree, D. W., Andreae, P., & Gao, X. 2007. Exploiting underrepresented query aspects for automatic query expansion. *Pages 191–200 of: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Daszykowski, M., Walczak, B., & Massart, D. L. 2001. Looking for natural patterns in data. Part 1. Density-based approach. *Chemometrics and Intelligent Laboratory Systems*, **56**(2), 83–92.

de Andrade, F. G., & Baptista, C. d. S. 2011. Using semantic similarity to improve information discovery in spatial data infrastructures. *Journal of Information and Data Management*, **2**(2), 181–194.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, **41**(6), 391–407.

Delfos, J., Veenendaal, B., & Tan, T. 2013. Enhancing accessibility to web mapping systems with technology-aligned adaptive profiles. *International Journal of Digital Earth*.

Dey, A. K. 2001. Understanding and Using Context. *Personal and Ubiquitous Computing*, **5**(1), 4–7.

Diggle, P. J. 2013. *Statistical analysis of spatial and spatio-temporal point patterns.* Third edn. CRC Monographs on Statistics & Applied Probability. CRC Press.

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R. S., Peng, Y., Reddivar, P., Doshi, V., & Sachs, J. 2004. Swoogle: a Search and Metadata Engine for the Semantic Web. *Pages 652–659 of: 13th ACM International Conference on Information and Knowledge Management (CIKM '04).* ACM.

Dong, H., Hussain, F., & Chang, E. 2008. A Survey in Semantic Search Technologies. *Pages 403–408 of: 2nd IEEE International Conference on Digital Ecosystems and Technologies.* IEEE.

Dong, H., Hussain, F., & Chang, E. 2010. A Framework Enabling Semantic Search in Health Service Ecosystems. *Pages 235–242 of: 6th International Conference on Semantics, Knowledge and Grids.* IEEE Computer Society.

Dragoni, M. 2015. Multilingual Ontology Mapping in Practice: A Support System for Domain Experts. *Pages 169–185 of: The 14th International Semantic Web Conference*, vol. Part II. Springer.

Egenhofer, M. J. 2002. Toward the Semantic Geospatial Web. *Pages 1–4 of: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems GIS'02.* ACM.

Egenhofer, M. J., & Franzosa, R. D. 1991. Point-set topological spatial relations. *International Journal of Geographical Information Science*, **5**(2), 161–174.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Pages 226–231 of:* Simoudis, E., Han, J., & Fayyad, U. (eds), *2nd International Conference on Knowledge Discovery and Data Mining (KDD).* The AAAI Press.

Farazi, F., Maltese, V., Fiunchiglia, F., & Ivanyukovich, A. 2011. A Faceted Ontology for a Semantic Geo-Catalogue. *Pages 169–182 of: 8th Extended Semantic Web Conference (ESWC 2011).* Springer Berlin Heidelberg.

Fielding, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures.* Ph.D. thesis, University of California, Irvine.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, **20**(1), 116–131.

Franzoni, V., Mencacci, M., Mengoni, P., & Milani, A. 2014. Semantic Heuristic Search in Collaborative Networks: Measures and Contexts. *Pages 141–148 of: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. IEEE.

Freitag, D. 2000. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, **39**(2), 169–202.

Gabrilovich, E., & Markovitch, S. 2009. Wikipedia-based Semantic Interpretation for Natural Language Processing. *Journal of Artificial Intelligence Research*, **34**, 443–498.

Gao, S., Mioc, D., Yi, X., Anton, F., Oldfield, E., & Coleman, D. J. 2009. Towards Web-based representation and processing of health information. *International Journal of Health Geographics*, **8**(1), 3.

Garla, V. N., & Brandt, C. 2012. Semantic similarity in the biomedical domain: an evaluation across knowledge sources. *BMC Bioinformatics*, **13**(261), 1–13.

Gruber, T. 2009. Ontology. *In:* Liu & Özsu (2009).

Guha, R., McCool, R., & Miller, E. 2003. Semantic Search. *Pages 700–709 of: 12th International Conference on World Wide Web (WWW '03)*. ACM.

Gulland, E.-K., Moncrieff, S., & West, G. A. W. 2015. Automated Calculation of Term Relatedness Weights for Semantic Searches. *Pages 313–316 of: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE.

Gulland, E.-K., Moncrieff, S., & West, G. A. W. 2016. Distributed Agents for Online Spatial Searches. *Spatial Information Research*, **24**(3), 191–202.

Guo, D., & Mennis, J. 2009. Spatial Data Mining and Geographic Knowledge Discovery — An Introduction. *Computers, Environment and Urban Systems*, **33**(6), 403–408.

Güting, R. H. 1994. An introduction to spatial database systems. *VLDB Journal*, **3**(4), 357–399.

Hadlak, S., Tominski, C., Schulz, H.-J., & Schumann, H. 2010. Visualization of attributed hierarchical structures in a spatiotemporal context. *International Journal of Geographical Information Science*, **24**(10), 1497–1513.

Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H., & Scheel, U. 2010. Faceted Wikipedia Search. *In: 13th International Conference on Business Information Systems (BIS2010).*

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & H, W. I. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, **11**(1).

Hassanpour, S., O'Connor, M. J., & Das, A. K. 2014. Clustering Rule Bases Using Ontology-Based Similarity Measures. *Journal of Web Semantics*, **25**, 1–8.

He, B., Patel, M., Zhang, Z., & Chang, K. C.-C. 2007. Accessing the Deep Web. *Communications of the ACM*, **50**(5), 94–101.

Hearst, M. A. 2006. Clustering versus faceted categories for information exploration. *Communications of the ACM*, **49**(4), 59–61.

Hendler, J. 2001. Agents and the Semantic Web. *IEEE Intelligent Systems*, **16**(2), 30–37.

Herlocker, J. L., Konstan, J. A., & Riedl, J. 2000. Explaining collaborative filtering recommendations. *Pages 241–250 of: Proceedings of the 2000 ACM conference on computer supported cooperative work (CSCW2000).* ACM.

Hill, L. L. 1990. *Access to geographic concepts in online bibliographic files: effectiveness of current practices and the potential of a graphic interface.* Tech. rept. University Microfilms International (UMI).

Hjaltason, G. R., & Samet, H. 1995. Ranking in Spatial Databases. *Pages 83–95 of:* Egenhofer, M. J., & Herring, J. R. (eds), *International Symposium on Spatial and Temporal Databases (SSD'95): Advances in Spatial Databases.* Berlin, Heidelberg: Springer.

Hjaltason, G. R., & Samet, H. 1999. Distance Browsing in Spatial Databases. *ACM Transactions on Database Systems*, **24**(2), 265–318.

Hofmann, T. 1999. Probabilistic Latent Semantic Indexing. *Pages 50–57 of: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99).* ACM.

Houle, M. E., Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. 2010. Can Shared-Neighbor Distances Defeat the Curse of Dimensionality? *Pages 482–500 of:* Gertz, M., & Ludäscher, B. (eds), *Scientific and Statistical Database Management*. Springer-Verlag Berlin Heidelberg.

Hovy, E., Navigli, R., & Ponzetto, S. P. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, **194**(January), 2–27.

Huang, W., & Webster, D. 2004. Enabling Context-Aware Agents to Understand Semantic Resources on the WWW and the Semantic Web. *Pages 138–144 of: IEEE/WIC/ACM International conference on Web Intelligence (WI'04)*. IEEE Computer Society.

Hutchinson, M. J., & Veenendaal, B. 2013. An agent-based framework for intelligent geocoding. *Applied Geomatics*, **5**(1), 33–44.

Jabeen, S. 2014. *Exploiting Wikipedia Semantics for Computing Word Associations*. Ph.D. thesis, Victoria University of Wellington.

Jain, A. K. 2010. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, **31**(8), 651–666.

Janowicz, K., Raubal, M., & Kuhn, W. 2011. The semantics of similarity in geographic information retrieval. *Journal of Spatial Information Science*, 29–57.

Janowicz, K., Bröring, A., Stasch, C., Schade, S., Everding, T., & Llaves, A. 2013. A RESTful proxy and data model for linked sensor data. *International Journal of Digital Earth*, **6**(3), 233–254.

Janowicz, K., Schade, S., Bröring, A., Keßler, C., Maué, P., & Stasch, C. 2010. Semantic Enablement for Spatial Data Infrastructures. *Transactions in GIS*, **14**(2), 111–129.

Jansen, B. J., Mullen, T., Spink, A., & Pedersen, J. 2006. Automated Gathering of Web Information: An In-Depth Examination of Agents Interacting with Search Engines. *ACM Transactions on Internet Technology*, **6**(4), 442–464.

Jimeno-Yepes, A., Berlanga-Llavori, R., & Rebholz-Schuhmann, D. 2010. Ontology refinement for improved information retrieval. *Information Processing and Management*, **46**(4), 426–435.

Johnson, R. A., & Wichern, D. W. 2002. *Applied Multivariate Statistical Analysis.* Fifth edn. Upper Saddle River, NJ USA: Prentice Hall.

Jones, R., Zhang, W. V., Rey, B., Jhala, P., & Stipp, E. 2008. Geographic intention and modification in web search. *International Journal of Geographical Information Science*, **22**(3), 229–246.

Joshi, A., de Araujo Novaes, M., Machiavelli, J., Iyengar, S., Vogler, R., Johnson, C., Zhang, J., & Hsu, C. E. 2012. A Human Centered GeoVisualization framework to facilitate visual exploration of telehealth data: A case study. *Technology and Health Care*, **20**(6), 457–471.

Kerschberg, L., Kim, W., & Scime, A. 2001. A Semantic Taxonomy-Based Personalizable Meta-Search Agent. *Pages 41–50 of: Proceedings of the Second International Conference on Web Information Systems Engineering (WISE '01)*, vol. 1. IEEE Computer Society.

Komlodi, A., & Lutters, W. G. 2008. Collaborative use of individual search histories. *Interacting with Computers*, **20**(1), 184–198.

Kozareva, Z., & Hovy, E. 2013. Tailoring the Automated Construction of Large-Scale Taxonomies Using the Web. *Language Resources and Evaluation*, **47**(3), 859–890.

Lacasta, J., Nogueras-Iso, J., & Zarazaga-Soria, F. J. 2010. A representation framework for terminological ontologies. *Chap. 2, pages 25–53 of: Terminological Ontologies: Design, Management and Practical Applications.* Semantic Web and Beyond, vol. 9. Springer eBooks.

Larson, R. R., & Frontiera, P. 2004. Spatial Ranking Methods for Geographic Information Retrieval (GIR) in Digital Libraries. *Pages 45–56 of:* Heery, R., & Lyon, L. (eds), *International Conference on Theory and Practice of Digital Libraries (ECDL 2004).* Lecture Notes in Computer Science, Vol 3232. Berlin, Heidelberg: Springer.

Lee, S. 2013. Understanding User Experience with Computer-Based Applications with Different Use Purposes. *International Journal of Human-Computer Interaction*, **29**(11), 689–701.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., & Bizer, C. 2015. DBpedia — a Large-

Scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, **6**(2), 167–195.

Lei, Y., Uren, V., & Motta, E. 2006. SemSearch: A Search Engine for the Semantic Web. *Pages 238–245 of:* Staab, S., & Svátek, V. (eds), *15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*. Springer.

Li, A. Q., Ahmed, A., Ravi, S., & Smola, A. J. 2014a. Reducing the Sampling Complexity of Topic Models. *Pages 891–900 of: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Li, L., & Horrocks, I. 2004. A Software Framework for Matchmaking Based on Semantic Web Technology. *International Journal of Electronic Commerce*, **8**(4), 39–60.

Li, L., Deng, H., Dong, A., Chang, Y., Baeza-Yates, R. A., & Zha, H. 2017. Exploring Query Auto-Completion and Click Logs for Contextual-Aware Web Search and Query Suggestions. *In: 26th International World Wide Web Conference*. IW3C2.

Li, W., Yang, C. P., Nebert, D. D., Raskin, R. G., Houser, P. R., Wu, H., & Li, Z. 2011. Semantic-Based Web Service Discovery and Chaining for Building an Arctic Spatial Data Infrastructure. *Computers and Geosciences*, **37**(11), 1752–1762.

Li, W., Goodchild, M. F., & Raskin, R. G. 2014b. Towards geospatial semantic search: exploiting latent semantic relations in geospatial data. *International Journal of Digital Earth*, **7**(1), 13–37.

Liu, L., & Özsu, M. T. (eds). 2009. *Encyclopedia of Database Systems*. Springer.

Liu, X., & Croft, W. B. 2004. Cluster-based retrieval using language models. *Pages 186–193 of: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information retrieval*. ACM.

Lu, H., & Sterling, L. 2000. Interoperability and semi-structured data in an open Web-based agent information system. *In: First International Conference on Web Information Systems Engineering*, vol. 1. IEEE.

Mac Aoidh, E., McArdle, G., Petit, M., Ray, C., Bertolotto, M., Claramunt, C., & Wilson, D. C. 2009. Personalization in adaptive and interactive GIS. *Annals of GIS*, **15**(1), 23–33.

Macal, C. M., & North, M. J. 2009. Agent-based modeling and simulation. *Pages 86–98 of:* Rossetti, M. D., Hill, R. R., Johansson, B., Dunkin, A., & Ingalls, R. G. (eds), *Winter Simulation Conference (WSC'09)*. ACM.

Madhavan, J., Afanasiev, L., Antova, L., & Halevy, A. 2009. *Harnessing the Deep Web: Present and Future*. arXiv:0909.1785 [cs.DB]. http://arxiv.org/abs/0909.1785. Accessed 2017-02-16.

Malo, P., Siitari, P., Ahlgren, O., Wallenius, J., & Korhonen, P. 2010. Semantic Content Filtering with Wikipedia and Ontologies. *Pages 518–526 of: Third International Workshop on Semantic Aspects in Dada Mining (SADM'10) in conjunction with the 2010 IEEE International Conference on Data Mining*. ACM.

Manoj, P., & Ghosh, S. K. 2006. An Approach for Service Oriented Discovery and Retrieval of Spatial Data. *In: Proceedings of the 2006 International Workshop on Service-Oriented Software Engineering*. ACM.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., & Sycara, K. 2008 (December). *OWL-S: Semantic Markup for Web Services*. Tech. rept. DARPA Agent Markup Language (DAML).

Mavoa, S., Witten, K., McCreanor, T., & O'Sullivan, D. 2012. GIS based destination accessibility via public transit and walking in Auckland, New Zealand. *Journal of Transport Geography*, **20**(1), 15–22.

McIlraith, S. A., Son, T. C., & Zeng, H. 2001. Semantic Web Services. *IEEE Intelligent Systems*, **16**(2), 46–53.

McInnes, B., Pedersen, T., & Pakhomov, S. 2009. UMLS-Interface and UMLS-Similarity : Open Source Software for Measuring Paths and Semantic Similarity. *Pages 431–435 of: Proceedings of the Annual Symposium of the American Medical Informatics Association*.

McInnes, B., Pedersen, T., Pakhomov, S., Liu, Y., & Melton-Meaux, G. 2013. UMLS::Similarity: Measuring the Relatedness and Similarity of Biomedical Concepts. *Pages 28–31 of: Proceedings of the 2013 NAACL HLT Demonstration Session*.

McNeill, F. 2013 (March). *Multi-Agent Semantic Web Systems: Lecture Notes.* University of Edinburgh. http://www.inf.ed.ac.uk/teaching/courses/masws/. Accessed 2017-03-20.

Medelyan, O., Milne, D., Legg, C., & Witten, I. H. 2009. Mining meaning from Wikipedia. *International Journal of Human-Computer Studies*, **67**(9), 716–754.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *In:* Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., & Weinberger, K. Q. (eds), *Advances in Neural Information Processing Systems 26 (NIPS 2013).*

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, **38**(11), 39–41.

Milne, D., & Witten, I. H. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. *Pages 25–30 of:* Bunescu, R., Gabrilovich, E., & Mihalcea, R. (eds), *Wikipedia and artificial intelligence: an evolving synergy.* AAAI Press.

Milne, D., & Witten, I. H. 2013. An open-source toolkit for mining Wikipedia. *Artificial Intelligence*, **194**, 18.

Mimno, D., & Blei, D. M. 2011. Bayesian checking for topic models. *Pages 227–237 of: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., & McCallum, A. 2011. Optimizing Semantic Coherence in Topic Models. *Pages 262–272 of: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics.

Minkov, E., Cohen, W. W., & Ng, A. Y. 2006. Contextual search and name disambiguation in email using graphs. *Pages 27–34 of: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06).*

Moncrieff, S., & West, G. A. W. 2013. A Web Service for the Dynamic Linkage and Visualisation of Multivariate Spatiotemporal Information. *In: 2nd ACM SIGSPATIAL*

International Workshop on the Use of GIS in Public Health (HealthGIS) 2013. ACM Press.

Moncrieff, S., Venkatesh, S., & West, G. 2008. Context Aware Privacy in Visual Surveillance. *Pages 1–4 of: 19th International Conference on Pattern Recognition (ICPR 2008)*. IEEE.

Moncrieff, S., Turdukulov, U., & Gulland, E.-K. 2016. Integrating geo web services for a user driven exploratory analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, **114**, 294–305.

Nelson, S. J., Johnston, W. D., & Humphreys, B. L. 2001 (November). *Chapter 11: Relationships in Medical Subject Headings (MeSH)*. https://www.nlm.nih.gov/mesh/meshrels.html. Accessed 2016-06-19.

Neuman, Y., Assaf, D., & Cohen, Y. 2013. Fusing distributional and experiential information for measuring semantic relatedness. *Information Fusion*, **14**(3), 281–287.

Noel, R., Pauchet, A., Grilheres, B., Malandain, N., Vercouter, L., & Brunessaux, S. 2014. Relevant Sources of Information Are Not Necessarily Popular Ones. *Pages 310–317 of: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1. IEEE.

Nwana, H. S. 1996. Software agents: an overview. *Knowledge Engineering Review*, **11**(3), 205–244.

OGC. 2005 (May). *OpenGIS® Web Feature Service Implementation Specification*. Tech. rept. OGC 04-094. Open Geospatial Consortium (OGC).

OGC. 2006 (March). *OpenGIS® Web Map Service Implementation Specification*. Tech. rept. OGC 06-042. Open Geospatial Consortium (OGC).

OGC. 2007 (June). *OpenGIS® Web Processing Service*. Tech. rept. OGC 05-007r7. Open Geospatial Consortium (OGC).

OGC. 2010 (November). *OpenGIS® Filter Encoding 2.0 Encoding Standard*. Tech. rept. OGC 09-026r1. Open Geospatial Consortium (OGC).

OGC. 2012 (July). *OpenGIS® Web Coverage Service 2.0 Interface Standard — Core: Corrigendum*. Tech. rept. OGC 09-110r4. Open Geospatial Consortium (OGC).

Page, L., Brin, S., Motwani, R., & Winograd, T. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Tech. rept. Stanford InfoLab.

Pakhomov, S., Pedersen, T., McInnes, B., Melton, G. B., Ruggieri, A., & Chute, C. G. 2011. Towards a Framework for Developing Semantic Relatedness Reference Standards. *Journal of Biomedical Informatics*, **44**(2), 251–265.

Pakhomov, S. V. S., McInnes, B., Adam, T., Liu, Y., Pedersen, T., & Melton, G. B. 2010. Semantic Similarity and Relatedness between Clinical Terms: An Experimental Study. *Pages 572–576 of: AMIA Annual Symposium.*

Palacio, D., Cabanac, G., Sallaberry, C., & Hubert, G. 2010. On the evaluation of Geographic Information Retrieval systems. *International Journal on Digital Libraries*, **11**(2), 91–109.

Pedersen, T., Pakhomov, S. V. S., Patwardhan, S., & Chute, C. G. 2007. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of Biomedical Informatics*, **40**, 288–299.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. 2014. Context Aware Computing for the Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, **16**(1), 414–454.

Pesquita, C., Faria, D., Falcão, P. L., & Couto, F. M. 2009. Semantic Similarity in Biomedical Ontologies. *PLS Computational Biology*, **5**(7).

Pesquita, C., Ferreira, J. D., Couto, F. M., & Silva, M. J. 2014. The epidemiology ontology: an ontology for the semantic annotation of epidemiological resources. *Journal of Biomedical Semantics*, **5**(1).

Petit, M., Ray, C., & Claramunt, C. 2007. A user context approach for adaptive and distributed GIS. *Pages 121–133 of:* Fabrikant, S. I., & Wachowicz, M. (eds), *The European Information Society.* Lecture Notes in Geoinformation and Cartography. Berlin Heidelberg: Springer.

Petit, M., Ray, C., & Claramunt, C. 2008. An Adaptive Interaction Architecture for Collaborative GIS. *Cartography and Geographic Information Science*, **35**(2), 91–102.

Petrakis, E. G. M., Varelas, G., Hliaoutakis, A., & Raftopoulou, P. 2006. Design and Evaluation of Semantic Similarity Measures for Concepts Stemming from the Same or Different Ontologies. *Pages 44–52 of: 4th Workshop on Multimedia Semantics.*

Rautenbach, V., Coetzee, S., & Iwaniak, A. 2013. Orchestrating OGC web services to produce thematic maps in a spatial information infrastructure. *Computers, Environment and Urban Systems*, **37**, 107–120.

Reed, T. W., Gulland, E.-K., West, G. A. W., McMeekin, D. A., & Moncrieff, S. 2016. Geographic Metadata Searching with Semantic and Spatial Filtering Methods. *Pages 85–92 of: GEOProcessing 2016: the 8th International Conference on Advanced Geographic Information Systems, Applications, and Services.*

Řehůřek, R., & Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. *Pages 45–50 of: Proceedings of the 2010 LREC Workshop on New Challenges for NLP Frameworks.* Valletta, Malta: ELRA. http://is.muni.cz/publication/884893/en.

Reichenbacher, T. 2001. Adaptive concepts for a mobile cartography. *Journal of Geographical Sciences*, **11**(Suppl 1), 43–53.

Ren, F., & Bracewell, D. B. 2009. Advanced Information Retrieval. *Electronic Notes in Theoretical Computer Science (ENTCS)*, **225**(January), 303–317.

Rinaldi, A. M. 2009. An Ontology-Driven Approach for Semantic Information Retrieval on the Web. *ACM Transactions on Internet Technology*, **9**(3).

Ristoski, P., & Paulheim, H. 2016. Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Journal of Web Semantics*, **36**, 1–22.

Robertson, S. E. 1977. The Probability Ranking Principle in IR. *Journal of Documentation*, **33**(4), 294–304.

Rowley, J. 2007. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, **33**(2), 163–180.

Russell, S. J., & Norvig, P. 2010. *Artificial Intelligence: A Modern Approach.* Third edn. Upper Saddle River, NJ USA: Prentice Hall.

Ruthven, I. 2011. Information Retrieval in Context. *Chap. 8, pages 187–207 of:* Melucci, M., & Baeza-Yates, R. A. (eds), *Advanced Topics in Information Retrieval*. Springer Berlin Heidelberg.

Sah, M., & Wade, V. 2016. Personalized concept-based search on the Linked Open Data. *Journal of Web Semantics*, **36**, 32–57.

Santos, D., Cardoso, N., Carvalho, P., Dornescu, I., Hartrumpf, S., Leveling, J., & Skalban, Y. 2008. GikiP at GeoCLEF 2008: Joining GIR and QA Forces for Querying Wikipedia. *Pages 894–905 of:* Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G. J. F., Kurimo, M., Mandl, T., Peñas, A., & Petras, V. (eds), *Workshop of the Cross-Language Evaluation Forum for European Languages CLEF 2008: Evaluating Systems for Multilingual and Multimodal Information Access*. Lecture Notes in Computer Science, volume 5706.

Sengupta, R. R., & Bennett, D. A. 2003. Agent-based modelling environment for spatial decision support. *International Journal of Geographical Information Science*, **17**(2), 157–180.

Sengupta, R. R., & Sieber, R. 2007. Geospatial Agents, Agents Everywhere. *Transactions in GIS*, **11**(4), 483–506.

Shadbolt, N. R., Hall, W., & Berners-Lee, T. 2006. The Semantic Web Revisited. *IEEE Intelligent Systems*, **2**(3), 96–101.

Shahid, R., Bertazzon, S., Knudtson, M. L., & Ghali, W. A. 2009. Comparison of distance measures in spatial and analytical modeling for health service planning. *BMC Health Services Research*, **9**(200).

Shlens, J. 2014. *A Tutorial On Principal Component Analysis*. 1404.1100 [cs.LG]. http://arxiv.org/abs/1404.1100. Accessed 2017-05-30.

Skarlatidou, A., Haklay, M. M., & Cheng, T. 2011. Trust in Web GIS: the role of the trustee attributes in the design of trustworth web GIS applications. *International Journal of Geographical Information Science*, **25**(12), 1913–1930.

Stevens, R., & Lord, P. 2009. Ontologies and Life Science Data Management. *In:* Liu & Özsu (2009).

Stojanovic, N. 2005. On the query refinement in the ontology-based searching for information. *Information Systems*, **30**(7), 543–563.

Sun, S., Wang, L., Ranjan, R., & Wu, A. 2014. Semantic analysis and retrieval of spatial data based on the uncertain ontology model in Digital Earth. *International Journal of Digital Earth*, **7**.

Supavetch, S., & Chunithipaisan, S. 2011. Interface independent geospatial services orchestration. *Information Technology Journal*, **10**(6), 1126–1137.

Tangi, R. I. 1998. Design and Implementation of the WordNet Lexical Database and Searching Software. *Chap. 4, pages 105–127 of:* Fellbaum, C. (ed), *WordNet: an Electronic Lexical Database*. MIT Press.

Tho, Q. T., Hui, S. C., Fong, A. C. M., & Cao, T. H. 2006. Automatic fuzzy ontology generation for semantic Web. *IEEE Transactions on Knowledge and Data Engineering*, **17**(6), 842–856.

Tomaszewski, B., & MacEachren, A. M. 2012. Geovisual analytics to support crisis management: Information foraging for geo-historical context. *Information Visualization*, **11**(4), 339–359.

Tsujii, J., & Ananiadou, S. 2005. Thesaurus or Logical Ontology, Which One Do We Need for Text Mining? *Language Resources and Evaluation*, **39**(1), 77–90.

Uren, V., & Motta, E. 2006. Semantic Search Components: A Blueprint for Effective Query Language Interfaces. *Pages 222–237 of:* Staab, S., & Svátek, V. (eds), *15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006)*. Springer.

Vaccari, L., Shvaiko, P., & Marchese, M. 2009. A geo-service semantic integration in Spatial Data Infrastructures. *International Journal of Spatial Data Infrastructures Research*, **4**(24–51).

Viktoratos, I., Tsadiras, A., & Bassiliades, N. 2015. A context-aware web-mapping system for group-targeted offers using semantic technologies. *Expert Systems with Applications*, **42**(9), 4443–4459.

Viroli, M., Ricci, A., & Omicini, A. 2006. Operating instructions for intelligent agent coordination. *Knowledge Engineering Review*, **21**(1), 49–69.

Vockner, B., Richter, A., & Mittlböck, M. 2013. From Geoportals to Geographic Knowledge Portals. *ISPRS International Journal of Geo-Information*, **2**(2), 256–275.

W3C. 2007 (26 June). *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. Tech. rept. World Wide Web Consortium (W3C).

W3C. 2012 (15 August). *OWL 2 Web Ontology Language Document Overview*. Tech. rept. World Wide Web Consortium (W3C).

W3C. 2014 (25 February). *RDF 1.1 Semantics*. Tech. rept. World Wide Web Consortium (W3C).

Walker, D. R. F., Newman, I. A., Medyckyj-Scott, D. J., & Ruggles, C. L. N. 1992. A system for identifying datasets for GIS users. *International Journal of Geographical Information Science*, **6**(6).

Waller, L. A., & Gotway, C. A. 2004. *Applied Spatial Statistics for Public Health Data*. New Jersey, USA: John Wiley & Sons.

Wang, W., De, S., Toenjes, R., Reetz, E., & Moessner, K. 2012. A Comprehensive Ontology for Knowledge Representation in the Internet of Things. *In: 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE.

Wang, Y., Huang, H., & Feng, C. 2017. Query Expansion Based on a Feedback Concept Model for Microblog Retrieval. *In: 26th International World Wide Web Conference*. IW3C2.

Weyns, D. 2010. *Middleware for Distributed Multi-Agent Systems*. Springer Berlin Heidelberg. Chap. 4, pages 93–122.

Willis, A., Gjersoe, N., Havard, C., Kerridge, J., & Kukla, R. 2004. Human Movement Behaviour in Urban Spaces: Implications for the Design and Modelling of Effective Pedestrian Environments. *Environment and Planning B*, **31**(6), 805–828.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. Fourth edn. Morgan Kaufmann.

Wooldridge, M., & Jennings, N. R. 1995. Intelligent agents: theory and practice. *Knowledge Engineering Review*, **10**(2), 115–152.

World Health Organisation. 2006. *International Classification of Diseases (ICD)*. 10th edn. Geneva: WHO.

WSMO-WG. 2008 (8 August). *Web Service Modeling Language (WSML).* Tech. rept. Web Service Modeling Ontology Working Group.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., & Steinberg, D. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems*, **14**(1), 1–37.

Xiong, C., Power, R., & Callan, J. 2017. Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding. *In: 26th International World Wide Web Conference.* IW3C2.

Yamada, I., Ito, T., Usami, S., Takagi, S., Takeda, H., & Takefuji, Y. 2014. Evaluating the Helpfulness of Linked Entities to Readers. *Pages 169–178 of: Proceedings of the 25th ACM Conference on Hypertext and Social Media (HT '14).* ACM.

Yi, X., Raghavan, H., & Leggetter, C. 2009. Discovering Users' Specific Geo Intention in Web Search. *Pages 481–490 of: Proceedings of the 18th International Conference on World Wide Web (WWW '09).* New York, NY USA: ACM.

Yue, P., Di, L., Yang, W., Yu, G., & Zhao, P. 2007. Semantics-based automatic composition of geospatial Web service chains. *Computers & Geosciences*, **33**(5), 649–665.

Zaila, Y. L., & Montesi, D. 2015. Geographic Information Extraction, Disambiguation and Ranking Techniques. *In: Proceedings of the 9th Workshop on Geographic Information Retrieval (GIS '15).* New York, NY USA: ACM.

Zhang, C., Zhao, T., & Li, W. 2010a. Automatic search of geospatial features for disaster and emergency management. *International Journal of Applied Earth Observation and Geoinformation*, **12**(6), 409–418.

Zhang, C., Zhao, T., Li, W., & Osleeb, J. P. 2010b. Towards logic-based geospatial feature discovery and integration using web feature service and geospatial semantic web. *International Journal of Geographical Information Science*, **24**(6), 903–922.

Zhang, C., Zhao, T., & Li, W. 2015. *Current and Future Challenges of Geospatial Semantic Web.* Springer. Chap. 7.

Zhang, Z., Gentile, A. L., & Ciravegna, F. 2012. Recent advances in methods of lexical semantic relatedness — a survey. *Natural Language Engineering*, **19**(4), 411–479.

Zhao, P., Foerster, T., & Yue, P. 2012. The Geoprocessing Web. *Computers & Geosciences*, **47**, 3–12.

Zhao, W., Chen, J. J., Perkins, R., Liu, Z., Ge, W., Ding, Y., & Zou, W. 2015. A heuristic approach to determine an appropriate number of topics in topic modeling. *BMC Bioinformatics*, **16(Suppl 13)**(S8).

Zhou, N. 2010. Ontological and Semantic Technologies for Geospatial Portals. *Chap. 10 of:* Zhao, P., & Di, L. (eds), *Geospatial Web Services: Advances in Information Interoperability.* IGI Global.

Appendices

Appendix A Test Term Set: MixMed50

These terms were each titles of Wikipedia articles at the time of collection. Terms were extracted from World Health Organisation (WHO) information webpages[78] and linked Wikipedia pages.

> Animal, Anus, arthritis, asthma, bacteria, Bee, Billion, Black Death, Caenorhabditis elegans, Caffeine, cancer, Cattle, Cheese, Cholera, Chopsticks, Cytokine, diabetes, disease, dragonfly, Enzyme, evaporation, fever, flu, genetic disorder, haemophilia, Hand washing, Health, HIV, hummingbird, illness, Immunity, infection, infectious disease, Lymphocyte, malaria, Masturbation, Mineral, mistletoe, obesity, Pediatrics, plow, poodle, Sexually transmitted disease, Stream, Tutankhamun, vaccination, virus, Waffle, Water cycle, yeast.

An extended set of 202 terms is used in some tests - that is, the fifty terms above plus 152 additional Wikipedia article titles, extracted as part of the same process described for the main fifty terms:

> Abdomen, Absinthe, Acarina, Adsorption, Algae, Analgesic, Apicomplexa, Atahualpa, Autoimmune disease, Autopsy, Autotroph, Bacillus, Basin (geology), Bee Hummingbird, Beta blocker, Bioinformatics, Biological hazard, Biological warfare, Biscuit, Botulism, Carboniferous, Carcinogen, Cardamom, Cell growth, Cell nucleus, Cell theory, Centers for Disease Control and Prevention, Chagas disease, Chemotherapy, Colony (biology), Complication, Crime and Punishment, Crystallography, Depressant, Dermatology, Developed country, Diabetes insipidus, Dough, Ear, Encephalitis, Endospore, Epinephrine, Era, Fern, Flatworm, Food poisoning, Gangrene, Genetic drift, Germ theory of disease, Gesneriaceae, Gonorrhea, Gram-negative, Gram-positive, Gut flora, Halteres, Headache, Helicobacter pylori, Herpes simplex, Hershey-Chase experiment, History of the Earth, Honeyeater, Hydrogen sulfide, Hypoglycemia, Indian independence movement, Indigenous peoples, Inhaler, Innate immune system, Jaundice, Joint, Larynx, Life expectancy, Lipase, Liver, Lung, Lyme disease, Mad cow disease, Medi-

[78] http://www.who.int/topics/en/, accessed 1/12/2014

cal emergency, Medication, Meningitis, Microbiology, Mimivirus, Model organism, Mosquito, Multicellular organism, Multiple myeloma, Mumps, Myocardial infarction, Myopia, Natural killer cell, Non-coding RNA, Odonata, Olfaction, Organ (anatomy), Pancreatic cancer, Pandemic, Pandoravirus, Parkinson's disease, Pathogen, Penicillin, Peptic ulcer, Pesticide, Pita, Pizza, Plague of Athens, Pneumothorax, Predation, Prion, Prokaryote, Prostate cancer, Protist, Protozoa, Pus, Quinine, Radiation, Refrigerant, Rhizobia, Right upper quadrant (abdomen), RNA interference, RNA virus, Saprophyte, Schistosomiasis, Seizure, Sequence analysis, Sickle-cell disease, Sleeping sickness, Spirochaete, Squanto, Staphylococcus aureus, Stethoscope, Stomach cancer, Streptococcus, Stromatolite, T cell, Teratogen, Thalassaemia, Titus, Trans fat, Transformation (genetics), Transposon, Tree of life (biology), Turbidity, Type 1 diabetes, Urethra, Vaccine, Vacuole, Virology, Vitamin, Wing, Wristband, X-ray, Yersinia pestis, Yun Poson.

The following pages include a table of manual relatedness weight judgements for the fifty terms, where 0 is 'unrelated' and 1 is 'very closely related, or synonymous'.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1. Animal	1	0.6	0	0	0.3	0.9	0	0	0.9	0	0	0.9	0.6	0	0	0.6	0	0.3	0	0.6	0	0	0	0.3	0
2. Anus	0.6	1	0.3	0.3	0.6	0.3	0	0	0	0	0.6	0.3	0	0.6	0	0	0.6	0.6	0	0	0	0.6	0.3	0.8	0.6
3. arthritis	0	0.3	1	0.8	0.8	0.3	0	0	0	0.6	0.6	0.6	0.6	0.3	0.3	0.6	0.6	0.9	0	0.3	0	0.3	0.3	0.9	0.8
4. asthma	0	0.3	0.8	1	0.6	0.3	0	0	0	0.6	0.8	0.3	0.3	0.3	0	0	0.8	0.9	0	0.8	0	0.8	0.8	0.9	0.3
5. bacteria	0.3	0.6	0.8	0.6	1	0.3	0	0.9	0.6	0	0.8	0.6	0.6	0.9	0	0.8	0.6	0.9	0	0.8	0.3	0.8	0.3	0.3	0
6. Bee	0.9	0.3	0.3	0.3	0.3	1	0	0	0	0.3	0.6	0.3	0	0	0	0.3	0.3	0.6	0.6	0.3	0	0.3	0	0.3	0
7. Billion	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8. Black Death	0	0	0	0	0.9	0	0	1	0	0	0	0	0	0.3	0	0	0	0.9	0	0	0	0.8	0.6	0	0
9. Caenorhabditis elegans	0.9	0	0	0	0.6	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10. Caffeine	0	0	0.6	0.6	0	0.3	0	0	0	1	0.6	0.3	0.3	0	0	0	0.6	0	0	0.6	0.6	0.6	0.3	0	0.3
11. cancer	0	0.6	0.6	0.8	0.8	0.6	0	0	0	0.6	1	0.6	0.3	0.3	0	0.9	0.8	0.9	0	0.6	0	0.8	0.3	0.6	0.3
12. Cattle	0.9	0.3	0.6	0.3	0.6	0.3	0	0	0	0.3	0.6	1	0.8	0	0	0.3	0	0.6	0	0.3	0	0.6	0.6	0.6	0
13. Cheese	0.6	0	0.6	0.3	0.6	0	0	0	0	0.3	0.3	0.8	1	0.3	0.3	0.3	0.3	0.6	0	0	0.8	0.3	0.3	0	0
14. Cholera	0	0.6	0.3	0.3	0.9	0	0	0.3	0	0	0.3	0	0.3	1	0	0.3	0	0.9	0	0.3	0.3	0.3	0.6	0	0.3
15. Chopsticks	0	0	0.3	0	0	0	0	0	0	0	0	0	0.3	0	1	0	0.3	0	0	0	0	0	0	0	0
16. Cytokine	0.6	0	0.6	0	0.8	0.3	0	0	0	0	0.9	0.3	0.3	0.3	0	1	0.3	0.9	0	0.6	0	0.8	0.8	0.3	0
17. diabetes	0	0.6	0.6	0.8	0.6	0.3	0	0	0	0.6	0.8	0	0.3	0	0.3	0.3	1	0.9	0	0.8	0	0.6	0.6	0.8	0
18. disease	0.3	0.6	0.9	0.9	0.9	0.6	0	0.9	0	0	0.9	0.6	0.6	0.9	0	0.9	0.9	1	0	0.6	0	0.9	0.9	0.9	0.9

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
19. dragonfly	0	0	0	0	0	0.6	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
20. Enzyme	0.6	0	0.3	0.8	0.8	0.3	0	0	0	0.6	0.6	0.3	0	0.3	0	0.6	0.8	0.6	0	1	0	0.6	0.6	0.6	0.3
21. evaporation	0	0	0	0	0.3	0	0	0	0	0.6	0	0	0.8	0.3	0	0	0	0	0	0	1	0	0	0	0
22. fever	0	0.6	0.3	0.8	0.8	0.3	0	0.8	0	0.6	0.8	0.6	0.3	0.3	0	0.8	0.6	0.9	0	0.6	0	1	0.9	0	0.3
23. flu	0	0.3	0.3	0.8	0.3	0	0	0.6	0	0.3	0.3	0.6	0.3	0.6	0	0.8	0.6	0.9	0	0.6	0	0.9	1	0	0.3
24. genetic disorder	0.3	0.8	0.9	0.9	0.3	0.3	0	0	0	0	0.6	0.6	0	0	0	0.3	0.8	0.9	0	0.6	0	0	0	1	0.9
25. haemophilia	0	0.6	0.8	0.3	0	0	0	0	0	0.3	0.3	0	0	0.3	0	0	0	0.9	0	0.3	0	0.3	0.3	0.9	1
26. Hand washing	0	0.3	0	0	0.6	0.3	0	0	0	0	0	0.3	0.3	0.6	0.3	0.3	0	0.8	0	0.3	0.8	0.3	0.8	0	0
27. Health	0.6	0.6	0.8	0.9	0.8	0.6	0	0.9	0.3	0.6	0.9	0.6	0.6	0.9	0.3	0.9	0.9	0.9	0	0.6	0	0.9	0.9	0.9	0.9
28. HIV	0	0.6	0.6	0.6	0	0	0	0	0	0.3	0.6	0.3	0	0.6	0	0.6	0.6	0.9	0	0.8	0	0.8	0.8	0	0.6
29. hummingbird	0.9	0.3	0	0	0	0.6	0	0	0	0	0	0	0	0	0	0	0.3	0.3	0.3	0	0.3	0	0	0	0
30. illness	0.3	0.6	0.9	0.9	0.9	0.6	0	0.9	0	0	0.9	0.6	0.6	0.9	0	0.9	0.8	0.9	0	0.6	0	0.9	0.9	0.9	0.9
31. Immunity	0	0	0.6	0.6	0.3	0	0	0.6	0	0	0.6	0.3	0.3	0.6	0	0.6	0.6	0.8	0	0	0	0.8	0.8	0.6	0.3
32. infection	0.3	0.8	0.8	0.8	0.9	0.6	0	0.8	0	0.6	0.8	0.6	0.6	0.8	0	0.9	0.6	0.9	0	0.6	0	0.9	0.9	0.6	0.6
33. infectious disease	0.3	0.8	0.8	0.8	0.8	0	0	0.9	0	0.3	0.8	0.6	0	0.9	0	0.9	0.3	0.9	0	0.8	0	0.9	0.9	0.6	0.6
34. Lymphocyte	0.6	0	0.6	0.3	0.8	0	0	0.3	0	0	0.6	0.3	0	0.3	0	0.9	0.3	0.8	0	0.8	0	0.6	0.6	0.6	0.3
35. malaria	0	0.6	0.6	0.6	0.3	0.3	0	0	0	0	0.8	0.6	0	0.6	0	0.3	0.6	0.9	0	0.6	0.3	0.9	0.8	0.8	0.6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
36. Masturbation	0	0.3	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0
37. Mineral	0.3	0	0.6	0.3	0.6	0.3	0	0	0	0.3	0.3	0.3	0.6	0	0	0	0.3	0.6	0	0.3	0.6	0.3	0.3	0.6	0.3
38. mistletoe	0.3	0	0	0	0	0.3	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0
39. obesity	0	0.6	0.8	0.9	0.8	0	0	0	0	0.3	0.9	0.3	0.8	0	0.3	0	0.8	0.9	0	0.6	0	0.3	0.3	0.8	0.3
40. Pediatrics	0	0.6	0.6	0.8	0.8	0.3	0	0	0	0	0.6	0	0.3	0.3	0	0.3	0.6	0.8	0	0.6	0	0.8	0.8	0.8	0.3
41. plow	0	0	0	0	0	0	0	0	0	0	0	0.6	0	0.3	0	0	0	0	0	0	0	0	0	0	0
42. poodle	0.9	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0	0
43. Sexually transmitted disease	0	0.6	0.3	0.3	0.8	0	0	0.6	0	0	0.8	0.6	0	0.3	0	0.8	0.3	0.9	0	0.3	0	0.6	0.6	0.3	0.3
44. Stream	0	0.6	0	0	0.3	0.3	0	0.3	0	0	0	0.3	0	0.6	0	0.6	0	0.3	0.3	0	0.8	0	0	0	0
45. Tutankhamun	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
46. vaccination	0	0.6	0.6	0.6	0.9	0.3	0	0.6	0	0.3	0.8	0.6	0	0.8	0	0.6	0.6	0.8	0	0.6	0	0.8	0.9	0.3	0.3
47. virus	0	0.3	0.9	0.6	0.6	0.6	0	0	0	0.6	0.9	0.6	0.3	0	0	0.8	0.6	0.9	0	0.8	0	0.9	0.9	0.3	0.3
48. Waffle	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0.3	0	0	0	0	0	0	0	0
49. Water cycle	0	0	0	0	0.3	0.3	0	0	0	0	0	0.3	0	0.3	0	0	0	0	0	0	0.9	0	0	0	0
50. yeast	0.3	0.6	0.8	0.6	0.6	0.3	0	0	0	0.6	0.3	0.3	0.6	0.3	0	0	0.6	0.8	0	0.8	0.3	0.6	0.6	0.3	0

	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1. Animal	0	0.6	0	0.9	0.3	0	0.3	0.3	0.6	0	0	0.3	0.3	0	0	0	0.9	0	0	0	0	0	0	0	0.3
2. Anus	0.3	0.6	0.6	0.3	0.6	0	0.8	0.8	0	0.6	0.3	0	0	0.6	0.6	0	0	0.6	0.6	0	0.6	0.3	0	0	0.6
3. arthritis	0	0.8	0.6	0	0.9	0.6	0.8	0.8	0.6	0.6	0	0.6	0	0.8	0.6	0	0.3	0.3	0	0	0.6	0.9	0	0	0.8
4. asthma	0	0.9	0.6	0	0.9	0.6	0.8	0.8	0.3	0.6	0	0.3	0	0.9	0.8	0	0	0.3	0	0	0.6	0.6	0	0	0.6
5. bacteria	0.6	0.8	0	0	0.9	0.3	0.9	0.8	0.8	0.3	0.3	0.6	0	0.8	0.8	0	0	0.8	0.3	0	0.9	0.6	0	0.3	0.6
6. Bee	0.3	0.6	0	0.6	0.6	0	0.6	0	0	0.3	0	0.3	0.3	0	0.3	0	0	0	0.3	0	0.3	0.6	0	0.3	0.3
7. Billion	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8. Black Death	0	0.9	0	0	0.9	0.6	0.8	0.9	0.3	0	0	0	0	0	0	0	0	0.6	0.3	0	0.6	0	0	0	0
9. Caenorhabditis elegans	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10. Caffeine	0	0.6	0.3	0	0	0	0.6	0.3	0	0	0	0.3	0	0.3	0	0	0	0	0	0	0.3	0.6	0	0	0.6
11. cancer	0	0.9	0.6	0	0.9	0.6	0.8	0.8	0.6	0.8	0	0.3	0	0.9	0.6	0	0	0.8	0	0	0.8	0.9	0	0	0.3
12. Cattle	0.3	0.6	0.3	0	0.6	0.3	0.6	0.6	0.3	0.6	0	0.3	0	0.3	0	0.6	0	0.6	0.3	0	0.6	0.6	0	0.3	0.3
13. Cheese	0.3	0.6	0	0	0.6	0.3	0.6	0	0	0	0	0.6	0	0.8	0.3	0	0	0	0	0	0	0.3	0.3	0	0.6
14. Cholera	0.6	0.9	0.6	0	0.9	0.6	0.8	0.9	0.3	0.6	0	0	0.3	0	0.3	0.3	0	0.3	0.6	0	0.8	0	0	0.3	0.3
15. Chopsticks	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0
16. Cytokine	0.3	0.9	0.6	0	0.9	0.6	0.9	0.9	0.9	0.3	0	0	0	0	0.3	0	0	0.8	0.6	0	0.6	0.8	0	0	0
17. diabetes	0	0.9	0.6	0.3	0.8	0.6	0.6	0.3	0.3	0.6	0	0.3	0	0.8	0.6	0	0.3	0.3	0	0	0.6	0.6	0.3	0	0.6
18. disease	0.8	0.9	0.9	0.3	0.9	0.8	0.9	0.9	0.8	0.9	0.3	0.6	0	0.9	0.8	0	0	0.9	0.3	0	0.8	0.9	0	0	0.8

	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
19. dragonfly	0	0	0	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0
20. Enzyme	0.3	0.6	0.8	0	0.6	0	0.6	0.8	0.8	0.6	0	0.3	0	0.6	0.6	0	0	0.3	0	0	0.6	0.8	0	0	0.8
21. evaporation	0.8	0	0	0.3	0	0	0	0	0	0.3	0	0.6	0	0	0	0	0	0	0.8	0	0	0	0	0.9	0.3
22. fever	0.3	0.9	0.8	0	0.9	0.8	0.9	0.9	0.6	0.9	0	0.3	0	0.3	0.8	0	0	0.6	0	0	0.8	0.9	0	0	0.6
23. flu	0.8	0.9	0.8	0	0.9	0.8	0.9	0.9	0.6	0.8	0	0.3	0	0.3	0.8	0	0	0.6	0	0	0.9	0.9	0	0	0.6
24. genetic disorder	0	0.9	0	0	0.9	0.6	0.6	0.6	0.6	0.8	0	0.6	0	0.8	0.8	0	0	0.3	0	0	0.3	0.3	0	0	0.3
25. haemophilia	0	0.9	0.6	0	0.9	0.3	0.6	0.6	0.3	0.6	0	0.3	0	0.3	0.3	0	0	0.3	0	0	0.3	0.3	0	0	0
26. Hand washing	1	0.8	0.6	0	0.6	0	0.6	0.8	0.3	0	0.6	0.3	0	0	0.6	0	0.3	0.6	0.3	0	0.3	0.6	0	0	0.6
27. Health	0.8	1	0.9	0.3	0.9	0.9	0.9	0.9	0.8	0.9	0.6	0.6	0.3	0.9	0.9	0	0.3	0.9	0.3	0	0.9	0.8	0.3	0.3	0.6
28. HIV	0.6	0.9	1	0	0.9	0.8	0.8	0.9	0.8	0.8	0.6	0.6	0	0.6	0.6	0	0	0.9	0	0	0.6	0.9	0	0	0.6
29. hummingbird	0	0.3	0	1	0.3	0	0	0.3	0	0	0	0	0	0.3	0	0	0	0	0	0	0	0	0.3	0	0
30. illness	0.6	0.9	0.9	0.3	1	0.8	0.9	0.9	0.8	0.9	0.3	0.6	0	0.9	0.6	0	0	0.9	0.3	0	0.8	0.9	0	0	0.8
31. Immunity	0	0.9	0.8	0	0.8	1	0.8	0.9	0.8	0.8	0	0	0	0.6	0.6	0	0	0.8	0	0	0.8	0.9	0	0	0.6
32. infection	0.6	0.9	0.8	0	0.9	0.8	1	0.9	0.8	0.9	0.6	0	0	0.8	0.8	0	0	0.9	0.3	0	0.9	0.9	0	0.3	0.9
33. infectious disease	0.8	0.9	0.9	0.3	0.9	0.9	0.9	1	0.8	0.9	0.3	0.6	0	0.8	0.8	0	0	0.9	0.3	0	0.9	0.9	0	0.3	0.9
34. Lymphocyte	0.3	0.8	0.8	0	0.8	0.8	0.8	0.8	1	0.6	0	0.3	0	0	0.3	0	0	0.6	0	0	0.6	0.8	0	0	0.3
35. malaria	0	0.9	0.8	0	0.9	0.8	0.9	0.9	0.6	1	0	0.3	0	0.3	0.6	0	0	0.3	0.3	0.3	0.6	0.3	0	0.3	0

	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
36. Masturbation	0.6	0.6	0.6	0	0.3	0	0.6	0.3	0	0	1	0	0	0	0	0	0	0.6	0	0	0	0	0	0	0.6
37. Mineral	0.3	0.6	0.6	0	0.6	0	0	0.6	0.3	0.3	0	1	0.3	0.6	0.3	0	0	0.6	0.6	0	0.3	0.6	0	0.6	0.6
38. mistletoe	0	0.3	0	0	0	0	0	0	0	0	0	0.3	1	0	0	0	0	0	0.3	0	0	0	0	0.3	0
39. obesity	0	0.9	0.6	0.3	0.9	0.6	0.8	0.8	0	0.3	0	0.6	0	1	0.6	0	0	0.3	0	0	0.3	0.8	0.6	0	0.6
40. Pediatrics	0.6	0.9	0.6	0	0.6	0.6	0.8	0.8	0.3	0.6	0	0.3	0	0.6	1	0	0	0.3	0	0	0.8	0.8	0	0	0.3
41. plow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0.3	0	0	0	0	0	0
42. poodle	0.3	0.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.3	0	0	0	0
43. Sexually transmitted disease	0.6	0.9	0.9	0	0.9	0.8	0.9	0.9	0.6	0.3	0.6	0.6	0	0.3	0.3	0	0	1	0	0	0.8	0.8	0	0	0.6
44. Stream	0.3	0.3	0	0	0.3	0	0.3	0.3	0	0.3	0	0.6	0.3	0	0	0.3	0	0	1	0	0	0.3	0	0.8	0
45. Tutankhamun	0	0	0	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
46. vaccination	0.3	0.9	0.6	0	0.8	0.8	0.9	0.9	0.6	0.6	0	0.3	0	0.3	0.8	0	0.3	0.8	0	0	1	0.9	0	0	0.3
47. virus	0.6	0.8	0.9	0	0.9	0.9	0.9	0.9	0.8	0.3	0	0.6	0	0.8	0.8	0	0	0.8	0.3	0	0.9	1	0	0	0.8
48. Waffle	0	0.3	0	0.3	0	0	0	0	0	0	0	0	0	0.6	0	0	0	0	0	0	0	0	1	0	0.6
49. Water cycle	0	0.3	0	0	0	0	0.3	0.3	0	0.3	0	0.6	0.3	0	0	0	0	0	0.8	0	0	0	0	1	0
50. yeast	0.6	0.6	0.6	0	0.8	0.6	0.9	0.9	0.3	0	0.6	0.6	0	0.6	0.3	0	0	0.6	0	0	0.3	0.8	0.6	0	1

Appendix B Test Term Set: WHO-105

Terms were extracted from World Health Organisation (WHO) information webpages[79]. Each term was also the title of a Wikipedia article at the time of collection (10/4/2015)[80]. N-grams consisting of more than one word are shown with underscores representing spaces.

> accident, ageing, anatomy, anemia, anthrax, anthropology, asthma, biological_hazard, biology, blindness, blood, burn, cancer, cardiovascular_disease, cataract, chemical_substance, childbirth, cholera, chromosome_abnormality, circulatory_system, Cleft_lip_and_palate, Club_foot, congenital_disorder, connective_tissue, deformity, dementia, dengue_fever, diabetes_mellitus, diagnosis, diarrhea, diet_(nutrition), disability, disease, drug_resistance, drug, ear, emergency, emotional_and_behavioral_disorders, encephalitis, endocrine_system, epidemic, epidemiology, epilepsy, eye, family_planning, food_safety, foodborne_illness, genetics, genitourinary_system, genomics, haemophilia, health_care, hearing_loss, hepatitis, HIV-AIDS, human_digestive_system, human_musculoskeletal_system, hygiene, Inborn_error_of_metabolism, infection, influenza, injury, leprosy, malaria, mastoid_process, measles, medical_sign, meningitis, mood_disorder, mumps, neoplasm, nervous_system, nutrition, obesity, occupational_safety_and_health, organism, parasitic_disease, pertussis, plague_(disease), pneumonia, poison, poliomyelitis, pollution, postpartum_period, pregnancy, Prenatal_development, psychiatry, psychology, public_health, rabies, respiratory_system, schizophrenia, sexually_transmitted_infection, sociology, stroke, suicide, surgery, symptom, tetanus, therapy, tropical_disease, tuberculosis, vaccine, youth_health, zoonosis.

Terms in this set can be loosely assigned to logical groups, for example:

- anatomy: biology, blood, connective_tissue, ear, endocrine_system, eye, genitourinary_system, human_digestive_system, human_musculoskeletal_system, mastoid process, nervous_system, respiratory_system.
- chronic condition: asthma, cancer, cardiovascular_disease, diabetes_mellitus, epilepsy, leprosy.

[79] http://www.who.int/topics, accessed 9/4/2015
[80] Two terms, 'Cleft lip and palate' and 'Club foot', were collected on 15/4/2015

- congenital anomaly: cleft_palate, club_foot, congenital_disorder, inborn_error_of_-metabolism.
- genetic disorder: chromosome_abnormality, haemophilia, inborn_error_of_metabolism.
- infectious disease: cholera, dengue_fever, hepatitis, HIV-AIDS, influenza, leprosy, malaria, measles, mumps, pertussis, pneumonia, poliomyelitis, tetanus, tuberculosis, zoonosis.
- medical care: diagnosis, drug, health_care, medical_sign, nutrition, occupational_-safety_and_health, surgery, symptom, therapy, vaccine.
- medical emergency: accident, biological_hazard, burn, chemical_substance, emergency, epidemic, poison, pollution, stroke, suicide.
- mental health: dementia, emotional_and_behavioral_disorders, mood_disorder, psychiatry, psychology, schizophrenia, suicide.
- noncommunicable disease: asthma, cancer, cardiovascular_disease, dementia, diabetes_mellitus, epilepsy, haemophilia, obesity.
- public and population health: ageing, diet_(nutrition), epidemiology, family_planning, food_safety, hygiene, nutrition, obesity, public_health, sociology, youth_health.
- tropical disease: dengue_fever, leprosy, malaria.
- vaccine-preventable condition: hepatitis, measles, mumps, pertussis, poliomyelitis, tetanus.

Appendix C DataAgent — Request Parameters

Requests sent to a DataAgent can use input parameters to specify what operation is required, for instance as part of a RESTful request. Different types of DataAgent can respond to a subset of parameters, as long as they handle the basic parameters: request, outputFormat, and request. If an unrecognised parameter is received, a warning will be returned, but processing based on the other parameters will not otherwise be affected.

Table C.1: Selection of possible request parameters for types of DataAgent.
* Parameters marked with an asterisk are valid for all agent types.

Type	Name	Examples
General	request *	getCapabilities, getWeight, search
	outputFormat *	JSON, XML
	query *	hospitals near Perth
Spatial	bbox	115.83, -31.98, 115.91, -31.93
	name	Perth
	originPoint *	115.83, -31.98
	boundary	115.83,-31.94 115.84,-31.95 ... 115.83,-31.94
	geom	Geometry in a text-based format
	geomType	JSON, GML
	distance	100
	units	m
	resultType	multipolygon, nearest, list
	maxFeatures	20
Other	attribute	suburb:Perth
	time	2010-05-30T09:00+08:00
	minWeight	0.75
	theme	health, education, travel

Appendix D DataAgent — Sample Output

The following listing shows example JavaScript Object Notation (JSON) output from a DataAgent source. The keys date, service, and request are present in every response. The error key only appears if required. The origin key, where present, represents a point location used for spatial searches.

The results key can contain an empty list, but will otherwise contain a list of results, each including at least a dataset name. A records key will only be included in a result for agents that carry out a deep search to return records from within a data source. All records include at least a label.

Listing D.1. Example JSON results from a search agent (synthetic data).

```
{"date": "29/06/2016",
 "request": {"request": "search", "service": "health",
             "query": "flu cases in Perth"},
 "service": "Perth Health",
 "origin": ((-32.0, 115.917), "Bentley"),
 "results": [
     {"name":    "ABS-SA3",
       "url":    "http://www.test.com",
       "type":   "health",
       "weight":  0.75,
       "records": [
           { "label":    "Perth Airport",
             "geometry": "<gml>...</gml>",
             "weight":    0.83,
             "type":     "Perth WA",          }, ...
       ], },
     {"name":    "World Health Organisation",
       "url":    "http://www.test2.com",
       "type":   "health",
       "weight":  0.71,
       "records": [
           { "label":    "WHO: Western Australia",
             "weight":    0.72,
             "type":     "Western Pacific",
             "lab confirmed flu":  "10%",     },
           { "label":    "WHO: Tasmania",
             "weight":    0.56,
             "type":     "Western Pacific",
             "lab confirmed flu":  "10%",     },
           { "label":    "WHO: Scotland",
             "weight":    0.48,
             "type":     "Europe",
             "lab confirmed flu":  "10%",     }, ...
       ], },  ]
 "error": ["Could not access WA Dept Health service"]
}
```

Appendix E DataAgent — Code

Models

The case study implementation of data agents used the Django web framework. This code is for a template data agent (DataAgent model) and includes basic requirements that specialist agents can build upon. It also defines DataAgentSource, which manages hierarchical agents that use other search agents as source(s).

Listing E.1. DataAgent and DataAgentSource Django Models.

```
#====================  Models  =========================
# DataAgent model plus DataAgentSource for linking
#     agents to sources of agent type.

from django.db import models
from model_utils.managers import InheritanceManager
from django.apps import apps

from django.contrib.contenttypes.models import ContentType
from django.contrib.contenttypes.fields import
    GenericForeignKey
from django.contrib.contenttypes import generic

import logging
import lxml
from lxml import etree
from StringIO import StringIO
from unidecode import unidecode
import json

import service.utils as utils
from service.requestSource import logParams,
    getOriginFromRequest

#============================
#----------------------------
# DataAgent to DataAgent sources facilitator.
# Allows for many-to-many links.
class DataAgentSource(models.Model):
    agent_content_type = models.ForeignKey(ContentType,
        editable=False,
        related_name='%(app_label)s_%(class)s_as_agent')
    agent_object_id = models.PositiveIntegerField()
    agent = generic.GenericForeignKey(
        'agent_content_type', 'agent_object_id')

    source_content_type = models.ForeignKey(ContentType,
        editable=False,
        related_name='%(app_label)s_%(class)s_as_source')
    source_object_id = models.PositiveIntegerField()
    source = generic.GenericForeignKey(
```

```
38          'source_content_type', 'source_object_id')
39
40      def __str__(self):
41          try:
42              return "%s-->%s" %(self.source, self.agent)
43          except Exception:
44              return ""
45
46
47  #============================
48  #----------------------------
49  # Parent class for a data search agent
50  class DataAgent(models.Model):
51      _DEFAULT_ORIGIN = [-31.9590090265, 115.85211772800001] #
            Lat,Long (Perth WA)
52
53      #model fields
54      url = models.URLField() #base URL for requests
55      name = models.CharField(max_length=200) #brief
            description of purpose e.g. 'disease rates'
56      serviceType = models.CharField(max_length=100,
57          default="data") #overwrite the default value in
                subclasses
58      #separate x,y coordinates because not all agents are
            geospatial
59      originLatitude = models.FloatField(null=True, default=
            None) #y
60      originLongitude = models.FloatField(null=True, default=
            None) #x
61      def _getOrigin(self):
62          "Returns latitude, longitude coordinate pair of
                origin"
63          if not (self.originLatitude and self.originLongitude)
                :
64              return ()
65          else:
66              return (self.originLatitude, self.originLongitude
                    )
67      def _setOrigin(self, latlonvalue):
68          try:
69              #allow for string "15.3,25.3"
70              yx = latlonvalue.split(',')
71              self.originLatitude = float(yx[0])
72              self.originLongitude = float(yx[1])
73              self.save()
74          except (AttributeError, ValueError, IndexError):
75              #Not a (valid) string - assume a y,x list of
                    values
76              try:
77                  self.originLatitude = latlonvalue[0]
78                  self.originLongitude = latlonvalue[1]
79                  self.save()
80              except ValueError:
81                  logging.error("Could not set agent origin to
                        %s" %latlonvalue)
82      agentOrigin = property(_getOrigin, _setOrigin)
83
84      #parameter keys
```

```
85      (QUERY, REQUEST, OUTPUT_FORMAT, ORIGIN) = range(4)
86      Parameters = ((QUERY, "query"),             #initial full
            text query
87                    (REQUEST, "request"),         #type e.g.
                          search
88                    (OUTPUT_FORMAT, "outputFormat"), #e.g. XML,
                           JSON
89                    #central location e.g. manual or from IP
                          address:
90                    (ORIGIN, "originPoint"),      #lat/long
                          request origin
91      )
92
93      #request types
94      RequestTypes = ('getCapabilities', 'search', 'getWeight')
95
96      #output formats
97      XML, JSON = range(2)
98      Output_Formats = (
99          (XML, "XML"),
100         (JSON, "JSON")
101     )
102
103     # how to sort a list of agents
104     class Meta:
105         ordering = ["name"]
106
107     #define managers
108     objects = models.Manager() #default manager
109     agents = InheritanceManager() #return all subclass
            objects
110
111     #***To manage any data sources (other DataAgent object/s)
            ****
112
113     # Property: a query set of DataAgent sources, or [] if
            none
114     def _getSources(self):
115         sourceQuerySet = DataAgentSource.objects.filter(
116             agent_object_id = self.id)
117         if sourceQuerySet:
118             sources = [das.source for das in sourceQuerySet]
119             return sources
120         else:
121             return []
122     dataSources = property(_getSources)
123
124     #-----------------------------------
125     # Return a weight (0-1) for the match likelihood of
126     # each source of this agent, based on input filter(s).
127     # Override to add functionality.
128     # INPUT: filter (dict) e.g. {'theme': 'health', 'origin
            ':[lat,long]}
129     #        threshold - ignore any sources with a weight
            below this
130     # OUTPUT: (float) 0.0-1.0
131     def weightedSources(self, filter, threshold=0.0):
132         #check for lack of sources
```

```
133         if not self.dataSources:
134             return []
135 
136         sources = [(agent, agent.weight(filter)) for agent in
                self.dataSources]
137         sources = [item for item in sources if item[1] >=
               threshold] #drop weights < T
138         sortedSources = sorted(sources, key=lambda item:item
               [1],
139                                reverse=True) #descending sort by
                                   weight
140         return sortedSources
141 
142     #-----------------------------------
143     # Add a DataSource as a new source
144     def addSource(self, newsource):
145         try:
146             if not isinstance(newsource, DataAgent):
147                 raise TypeError("Source must be a type of
                       DataAgent")
148             result = DataAgentSource.objects.get_or_create(
149                 agent_object_id = self.id,
150                 agent_content_type = ContentType.objects.
                       get_for_model(self),
151                 source_object_id = newsource.id,
152                 source_content_type = ContentType.objects.
                       get_for_model(newsource)
153             )
154             if not result[1]:
155                 #already created
156                 logging.warning("Source '%s' already added to
                        agent %s"
157                     %(newsource.name, self.name))
158         except TypeError:
159             logging.exception("Cannot add source to agent %s"
160                     %self.name)
161     #************************************************
162 
163     #-----------------------------------
164     # String representation of agent
165     def __str__(self):
166         if self.serviceType and self.name:
167             return "%s (%s)" %(self.name, self.serviceType)
168         else:
169             return "DataAgent object"
170 
171     #-----------------------------------
172     # Override to return capabilities metadata
173     # INPUT: outputFormat: JSON by default. Other option XML
174     #        sourceInfo: If True (default), include info
           about source agent(s)
175     def getCapabilities(self, outputFormat="JSON", sourceInfo
           =True):
176         #return ""
177         if outputFormat=="JSON":
178             datadict = {"Service URL": self.url, #
                   targetNamespace
179                 "Service Name": self.name,
```

```
180                 "Service Type": self.serviceType,
181                 "Recognised Requests": self.RequestTypes,
182                 "Recognised Parameters": self.paramList(),
183             }
184             if sourceInfo:
185                 sources = self.getSourcesInfo()
186                 datadict.update({"data sources": sources})
187             capabilities = json.dumps(datadict)
188         else:
189             #ignore any unknown formats; assume default XML
190             root = etree.Element("Capabilities")
191             child = etree.SubElement(root, "ServiceName")
192             child.text = self.name
193             child = etree.SubElement(root, "ServiceType")
194             child.text = self.serviceType
195             child = etree.SubElement(root, "ServiceURL")
196             child.text = self.url
197             #list parameters
198             paramElem = etree.SubElement(root, "ParameterList
                    ")
199             for param in self.paramList():
200                 child = etree.SubElement(paramElem, "
                        Parameter")
201                 child.text = param
202                 if param.lower() == "request":
203                     for reqtype in self.RequestTypes:
204                         grandchild = etree.SubElement(child,
                                "Type")
205                         grandchild.text = reqtype
206             #list data source info
207             if sourceInfo:
208                 try:
209                     sources = self.dataSources
210                     sourceroot = etree.Element("Data_sources"
                            )
211                     for source in sources:
212                         sourceCapabilities = source.
                                getCapabilities(outputFormat)
213                         sourceXml = etree.parse(StringIO(
                                sourceCapabilities))
214                         sourceroot.append(sourceXml)
215                     root.append(sourceroot)
216                 except (TypeError, etree.XMLSyntaxError):
217                     logging.warning("Could not add data
                            source capability data")
218             capabilities = etree.tostring(root, pretty_print=
                    True)
219
220         return capabilities
221
222     #---------------------------------
223     # Return a weight (0-1) for the likelihood of this agent,
224     # based on input filter(s)
225     # <Override to add functionality>
226     # INPUT: filter (dict) e.g. {...'theme': 'health', '
            originPoint':[lat,long]}
227     # OUTPUT: (float) 0.0-1.0
228     def weight(self, filter):
```

```
229             return 1.0
230
231         #---------------------------------
232         # Return filtered features as a dictionary
233         # {source1:{...}, source2:{...}, ...}
234         # <Override to add functionality>
235         def process(self, queryParams):
236             #log request
237             logParams(queryParams, agent=self.process)
238
239             serviceInfo =  {
240                 'service': self.serviceType,
241                 'request': queryParams,
242                 'date': utils.today(),
243                 'results': []
244             }
245
246             #check that input parameters are valid
247             validParams = self.paramList()
248             isOK = True
249             for key in queryParams.keys():
250                 if key not in validParams:
251                     message = "[{}] - '{}'".format(
252                               self.serviceType, key) \
253                         + " is not a recognised parameter"
254                     logging.warning(message)
255
256             #get capabilities request
257             if self.isCapabilitiesRequest(queryParams):
258                 outputFormat = queryParams.get(
259                                   'outputFormat', 'JSON')
260                 capabilities = self.getCapabilities(
261                                       outputFormat=outputFormat)
262                 return capabilities
263             #data source relevance weight request
264             elif self.isWeightRequest(queryParams):
265                 weight = self.weight(queryParams)
266                 result = {'name':self.name, 'url':self.url,
267                           'weight':weight}
268                 serviceInfo['results'] = [result]
269             #query request
270             else:
271                 pass #In subclasses, update queryparams then call
                        :
272                 # resultlist = self.getSourceResults(queryParams)
273                 # serviceInfo['results'] = resultlist
274
275             return serviceInfo
276
277         # Utilities----------------------
278
279         #---------------------------------
280         # Attempt to extract location from query. Priority order:
281         # 1. web request 2. query parameters 3. this agent's
               default
282         # If none found, returns ()
283         # INPUT: request (optional) django.http.HttpRequest
284         #        queryParams (optional) input query parameters
```

```
285     # OUTPUT: (lat,lon) - may be empty
286     def queryLocation(self, request=None, queryParams={}):
287         latlon = ()
288         #1. query parameter
289         queryorigin = self.getParam(self.ORIGIN, queryParams)
290         if queryorigin:
291             strlatlon = queryorigin[0] #assume only one
292             try:
293                 yx = strlatlon.split(',')
294                 latlon = (float(yx[0]), float(yx[1]))
295                 #latlon = tuple([float(x) for x in yx[:2]])
296                 return latlon
297             except IndexError, ValueError:
298                 logging.error("Invalid format for origin
                        coordinates in query")
299         #2. web request
300         if request:
301             origin = getOriginFromRequest(request)
302             if origin:
303                 return origin[0] #(lat, long)
304         #3. this agent's default, if any
305         latlon = self.agentOrigin
306
307         return latlon
308
309     #---------------------------------
310     # Check if request includes getCapabilities (overrides
           any others)
311     def isCapabilitiesRequest(self, queryParams):
312         try:
313             if queryParams[self.param(self.REQUEST)].lower()
                    == "getcapabilities":
314                 return True
315         except KeyError:
316             pass
317         return False
318
319     #---------------------------------
320     # Check if request includes getWeight (overrides any
           others)
321     def isWeightRequest(self, queryParams):
322         try:
323             if queryParams[self.param(self.REQUEST)].lower()
                    == "getweight":
324                 return True
325         except KeyError:
326             pass
327         return False
328
329
330     #---------------------------------
331     # Get result list from sources. If no sources, returns []
332     def getSourceResults(self, queryParams):
333         sources = self.dataSources
334         #ASSUMES each data source featureType has a unique
               name
335         #   (if not, last match overrides any previous ones)
336         resultlist = []
```

```
337             for source in sources:
338                 result = source.process(queryParams)
339                 if result and result.has_key('results'):
340                     for res in result['results']:
341                         dataset = {'name': source.name,
342                                    'url': source.url}
343                         if result.has_key('error'):
344                             dataset['error'] = result['error']
345                         if res.has_key('records'):
346                             dataset['records'] = res['records']
347                         resultlist.append(dataset)
348             return resultlist
349         #end getSourceResults
350 
351         #-----------------------------------
352         # Redirect agent's URL to a new one (if valid)
353         def redirect(self, newUrl):
354             #check if valid
355             if utils.validURL(newUrl):
356                 self.url = newUrl
357                 self.save()
358 
359         #-----------------------------------
360         # Return agent service info as a dictionary
361         def getInfo(self):
362             info = {'url': self.url,
363                     'name': self.name,
364                     'serviceType': self.serviceType
365             }
366             return info
367 
368         #-----------------------------------
369         # Return data source info as a list of dicts
370         def getSourcesInfo(self):
371             info = []
372             sources = self.dataSources
373             for source in sources:
374                 info.append(source.getInfo())
375             return info
376 
377         #-----------------------------------
378         # Return a list of valid parameters
379         def paramList(self):
380             return sorted([p[1] for p in self.Parameters])
381 
382         #-----------------------------------
383         # Get parameter key as text from class constant,
384         # where key matches a Parameter value
385         def param(self, index):
386             paramDict = dict(self.Parameters)
387             try:
388                 return paramDict[index]
389             except KeyError:
390                 return ""
391 
392         #-----------------------------------
393         # Get a list of query values for a recognised
394         # parameter (case-insensitive).
```

```
395     # If no match, returns an empty list.
396     # INPUT: target - parameter name (or ID integer)
397     #         request - dict of query parameters
398     def getParam(self, target, queryParams):
399         try:
400             key = int(float(target))
401             target = self.param(key)
402         except ValueError:
403             pass #already a string
404         if target.lower() in map(str.lower, self.paramList())
                :
405             return [v for (k,v) in queryParams.items()
406                     if k.lower() == target.lower()]
407         return []
```

Handlers

Handlers adding general functionality to DataAgent instances were also implemented, including TextQueryHandler for agents that need to extract place names and spatial operators from a text query, and SpatialHandler for agents managing geospatial data. One or more handlers can be incorporated into a specialised data agent by multiple inheritance, for example:

```
class WFSAgent(DataAgent, TextQueryHandler, SpatialHandler)
```

Listing E.2. Handler classes for search agents.

```
1  #====================  Handlers  ========================
2  # For DataAgent model and its subclasses (optionally):
3  #     classes for multiple inheritance where needed.
4
5  import logging
6
7  import json
8  import re
9  import math
10
11 from django.contrib.gis.geos import GEOSGeometry,
       GEOSException
12 from django.contrib.gis.geos import Polygon, MultiPolygon
13 from django.contrib.gis.gdal.error import GDALException
14
15 from collections import Sequence
16 from itertools import chain, count
17
18 #----- CONSTANTS -----------------------------
19 DEFAULT_GML = "2.0" #"3.2.1"
20 DEFAULT_SRS_CODE = "EPSG:4326" #WGS84 Long Lat
21 DEFAULT_GEOMFIELD = "the_geom"
22 DEFAULT_OPERATOR = "Within"
```

```
23 EARTH_MAX_KM = 20200.0 #approx 20,200km max distance between
       2 points on Earth
24 EARTH_RADIUS_KM = 6371 #approx 6371km radius of Earth (or
       6367)
25
26 DEGSYM = u'\N{DEGREE SIGN}'
27 UNIT_NAMES = {'m':   ['m', 'metre', 'metres', 'meter', '
       meters'],
28               'km':  ['km', 'kilometre', 'kilometres',
29                       'kilometer', 'kilometers'],
30               'mi':  ['mi', 'mile', 'miles'], #assume
                     international: =1,760yards=5,280ft
31               'ft':  ['ft', 'foot', 'feet'],
32               'deg': ['deg', 'decideg', 'degree', 'degrees',
33                       'decimal degree', 'decimal degrees',
                            DEGSYM ]
34              }
35 SUPPORTED_UNITS = UNIT_NAMES.keys()
36
37 #============================
38 #----------------------------
39 # Handler for DataAgents that need to interpret query text.
40 class TextQueryHandler(object):
41     # Name variations, suffixes etc
42     NAME_VARIATIONS = ( ('Mount', 'Mt'),
43         ('North', 'Nth', 'N'), ('South', 'Sth', 'S'),
44         # ---- etcetera ----
45         ('Street', 'St', 'Str'), ('Station', 'Sta', 'Statn',
               'Stn'),
46       )
47
48     # Get a query string from field name and search pattern
49     def getTextFilterParameter(self, field, pattern,
           wildcardsOK=False):
50         if not wildcardsOK:
51             try:
52                 pattern = pattern.replace('*', ' ').strip()
53             except AttributeError:
54                 pass
55
56         if isinstance(field, basestring):
57             return {"query": "%s:%s" %(field, pattern)}
58         else:
59             #to override, e.g. find best attribute from input
                   list
60             return {"query": ""}
61
62     # Return possible values for region Y in patterns:
63     #   X <operator> Y      "shops in London"
64     #   Y X                 "East London shops"
65     @staticmethod
66     def getRegionName(query, operations=["in", "near"]):
67         result = {'operation':'', 'feature':[], 'region':[]}
68         opPos = [query.find(op) for op in operations]
69         if any(pos >0 for pos in opPos):
70             #X <op> Y: y is area. Ignore if query starts with
                   <op>
71             idx = 0
```

```
72              for pos in opPos:
73                  if pos >= 0:
74                      break
75                  idx += 1
76              op = operations[idx]
77              result['operation'] = op
78              terms = query.split(op)
79              result['feature'] = [terms[0].strip()]
80              result['region'] = [terms[1].strip()]
81          else:
82              #X Y ('Mt Lawley bus stops'): X is area
83              terms = query.split(" ") #TODO: check for '='
                    also
84              names = [" ".join(terms[:n+1]).strip()
85                  for n in range(len(terms))]
86              targets = [" ".join(terms[n:]).strip()
87                  for n in range(len(terms))]
88              result['region'] = names
89              result['feature'] = targets
90          return result
91
92      # Get a list of alternative names from a single name,
93      # (e.g. Mt/Mount or Rd/Road)
94      @staticmethod
95      def _lowertuple(tup):
96          return tuple([str(t).lower() for t in tup])
97      @classmethod
98      def getAltNames(TestQueryHandler, name,
99                      namelist=TestQueryHandler.NAME_VARIATIONS
                            ):
100         testname = name.lower()
101         index = return [i for i,x in
102             enumerate(testname in _lowertuple(namelist))]
103         return list(chain(*[namelist[i] for i in index]))
104
105
106 #============================
107 #----------------------------
108 # Handler for DataAgents dealing with location data
109 class SpatialHandler(object):
110     #CONSTANTS
111     RESULT_TYPES = ('JSON', 'GML', 'WKT', 'WKB', 'KML',
112                     'GML2', 'GML3')
113     DEFAULT_WEIGHT = 0.5 #default weight (probability)
114     DEFAULT_SRID = 4326 #WGS84 Long Lat (x,y) - unprojected
115
116     # Get description, e.g. from spatial dataset metadata
117     def getDescription(self):
118         return ""
119
120     # Get spatial reference and SRID
121     def getSpatialRef(self):
122         #Spatial ref code as string e.g. GML EPSG code + SRID
                number
123         return ("", -1)
124
125     # Get bounding box of data, in degrees lat & long
126     # OUTPUT: (minx, miny, maxx, maxy) in degrees
```

```
127     def getBBox(self):
128         return self._getBBoxFromSources()
129 
130     # Get bounding box from data sources (if any)
131     def _getBBoxFromSources(self):
132         try:
133             sources = self.dataSources
134             if not sources:
135                 return () #no sources to get box from
136         except AttributeError:
137             return () #no dataSources attribute
138 
139         boxes = []
140         for source in sources:
141             try:
142                 sourceBox = source.getBBox()
143                 if sourceBox:
144                     boxes.append(sourceBox)
145             except AttributeError:
146                 pass #ignore sources without bounding boxes
147         newbox = mergeBBoxes(boxes)
148         return newbox
149 
150     # Get centroid of data, in latitutde/longitude.
151     # Currently calculated from bounding box: override to add
        accuracy.
152     # OUTPUT: (lat,long) or empty tuple on error
153     def getCentroid(self):
154         bbox = self.getBBox()
155         try:
156             x = (bbox[0] + bbox[2])/2.0
157             y = (bbox[1] + bbox[3])/2.0
158             return (y,x) #lat/long
159         except IndexError:
160             logging.error("Could not find centroid for agents
                ' data")
161             return ()
162 
163     # Return a weight (0.0-1.0) of the likely relevance of
164     # this agents' data sources to a reference spatial
        feature.
165     # Override to add functionality.
166     # INPUT: feature (may be an origin point, bounding box or
        polygon region)
167     def weightToReferenceFeature(self, feature):
168         bbox = self.getBBox()
169         if bbox:
170             #This agent has spatial information
171             featuretype = geometryType(feature)
172             if featuretype.upper() == 'POINT':
173                 return self._weightToOrigin(feature)
174             elif featuretype.upper() == 'POLYGON':
175                 return self._weightToRegion(feature)
176             else:
177                 #get bounding box if necessary
178                 return self._weightToBBox(feature)
179         else:
180             #No spatial information: always give max
```

```
                likelihood
            return self.DEFAULT_WEIGHT

    # Weight (0.0-1.0) of source data to an origin point
    # INPUT: [lat,long]
    def _weightToOrigin(self, origin):
        try:
            x = origin[0]
            y = origin[1]
            #z = origin[2]
        except KeyError:
            try:
                xy = getGeosGeom(origin)
                x,y = xy.centroid.coords[:2]
            except (IndexError, AttributeError):
                logging.error("Origin not a valid point;
                    cannot calculate weight")
                return self.DEFAULT_WEIGHT

        bbox = self.getBBox()
        try:
            #case 1: origin is in bbox
            if ( (bbox[0] <= x <= bbox[2]) and (bbox[1] <= y
                <= bbox[3]) ):
                #compare x,y
                return 1.0
            else:
                centroid = self.getCentroid()
                xy = (x,y)
                distRatio = getGCD(origin, xy, asRatio=True)
                return 1.0-distRatio
        except IndexError, TypeError:
            logging.error("Could not calculate weight to
                origin point")
        return self.DEFAULT_WEIGHT

    # Weight (0.0-1.0) of source data to a bounding box of
        interest
    # INPUT: [minx, miny, maxx, maxy]
    def _weightToBBox(self, bbox):
        newbox = getFeatureBBox(bbox)
        testbox = self.getBBox()
        try:
            #case 1: box is in bbox, or vice versa
            if boxInBox(newbox, testbox) or boxInBox(testbox,
                newbox):
                return 1.0
            else:
                boxCentre =  ((newbox[1]+newbox[3])/2.0,
                              (newbox[0]+newbox[2])/2.0) #lat
                                  ,lon
                return self._weightToOrigin(boxCentre)
        except IndexError, TypeError:
            logging.error("Could not calculate weight to
                bounding box")
        return self.DEFAULT_WEIGHT

    # Weight (0.0-1.0) of source data to a region of interest
```

```
231     # INPUT: polygon object or string representation
232     def _weightToRegion(self, region):
233         #TODO: use distance to feature polygon, not
                boundingbox
234         regionbox = getFeatureBBox(region)
235         return self._weightToBBox(regionbox)
236 #============================================
```

Views

Web services were implemented as Django views. Example Python code for the WFSAgent class is listed below. In this case, a WFSAgent instance has been initialised as a global variable (`_myWFSAgent`), although more sophisticated data discovery services could be substituted.

This view is accessed from a Uniform Resource Locator (URL) of `.../ogc/wfs`, as defined in urls.py for the agent:

Listing E.3. Django urls.py file for WFSAgent.

```
1 from django.conf.urls import patterns, url
2 from agent_OGC import views as ogcViews
3
4 urlpatterns = patterns('agent_OGC.views',
5     #return filtered points from WFS
6     url(r'wfs$', ogcViews.wfsRecords,
7         name='wfs'),
8 )
```

Listing E.4. Python code to implement a Web service for a WFSAgent via a Django view.

```
1 from django.http import JsonResponse
2 from agent_OGC.models import WFSAgent
3
4 # get-or-create an agent model
5 def initAgent(agentName, agentURL, wfsURL):
6     global _myWFSAgent
7
8     if not _myWFSAgent:
9         created = False
10        if not agentURL:
11            logging.error("Cannot find or create new WFS
                  Agent")
12            return
13        else:
```

```
14              _myWFSAgent, created = WFSAgent.objects.
                    get_or_create(
15                  url=agentURL, name=agentName, wfsUrl=wfsURL)
16          if created:
17              #not previously set up
18              logging.info("Created new WFSAgent")
19
20  # view for agent web service
21  def wfsRecords(request, wfsagent=None):
22      logRequest(request, wfsRecords) #log request
23
24      global _myWFSAgent
25      if wfsagent and not (isinstance(wfsagent, WFSAgent)):
26          logging.warning("Invalid WFSAgent input. Using
                default")
27          wfsagent = _myWFSAgent
28      if not wfsagent:
29          initAgent(...)
30          wfsagent = _myWFSAgent
31
32      # get request parameters
33      params = {}
34      if request.method == 'POST':
35          params = request.POST.dict()
36      elif request.method == 'GET':
37          params = request.GET.dict()
38
39      response = {}
40      if wfsagent:
41          #get results from agent
42          response = wfsagent.process(params)
43      else:
44          logging.error("Invalid WFSAgent")
45          #return information about faulty call
46          response = {
47              'request':params,
48              'error': 'Invalid MyNewAgent agent',
49              'date': utils.today(),
50              'results': []
51          }
52
53      try:
54          response = json.loads(response)
55      except (TypeError, ValueError):
56          #assume already a valid dictionary
57          pass
58
59      return JsonResponse(response)
```