ASTON UNIVERSITY

Discovering Knowledge Structures in Mind Maps of Mental Health Risks

Keith Priscott

For the Degree of Doctor of Philosophy, August, 2011.

Abstract

Keith Priscott, Aston University.

"Discovering Knowledge Structures in Mind Maps of Mental Health Risks"

For the Degree of Doctor of Philosophy, 2011.

This thesis addressed the problem of risk analysis in mental healthcare, with respect to the GRiST project at Aston University. That project provides a risk-screening tool based on the knowledge of 46 experts, captured as mind maps that describe relationships between risks and patterns of behavioural cues. Mind mapping, though, fails to impose control over content, and is not considered to formally represent knowledge. In contrast, this thesis treated GRiSTs mind maps as a rich knowledge base in need of refinement; that process drew on existing techniques for designing databases and knowledge bases.

Identifying well-defined mind map concepts, though, was hindered by spelling mistakes, and by ambiguity and lack of coverage in the tools used for researching words. A novel use of the Edit Distance overcame those problems, by assessing similarities between mind map texts, and between spelling mistakes and suggested corrections. That algorithm further identified *stems*, the shortest text string found in related word-forms. As opposed to existing approaches' reliance on built-in linguistic knowledge, this thesis devised a novel, more flexible text-based technique.

An additional tool, Correspondence Analysis, found patterns in word usage that allowed machines to determine likely intended meanings for ambiguous words. Correspondence Analysis further produced clusters of related concepts, which in turn drove the automatic generation of novel mind maps. Such maps underpinned adjuncts to the mind mapping software used by GRiST; one such new facility generated novel mind maps, to reflect the collected expert knowledge on any specified concept.

Mind maps from GRiST are stored as XML, which suggested storing them in an XML database. In fact, the entire approach here is "XML-centric", in that all stages rely on XML as far as possible. A XML-based query language allows user to retrieve information from the mind map knowledge base. The approach, it was concluded, will prove valuable to mind mapping in general, and to detecting patterns in any type of digital information.

Keywords: GRiST, FreeMind, Mind Map, XML, WordNet, Levenshtein Distance, Correspondence Analysis, Spelling, Stemming, Metadata.

List of Abbreviations

L, L'	The Levenshtein Distance, and a version adjusted for word-length.
1NF - 3NF	First, Second, and Third Normal Forms: steps in refining databases.
BOI	Basic Organising Idea: the main node in a mind map.
BOW	Bag Of Words: an extreme reductionist approach to documents.
CA	Correspondence Analysis: a multivariate analytical tool.
DL	Description Logics: applied to machine reasoning.
GRiST	Galatean Risk Screening Tool.
HC	Hierarchical Clustering: one of the outputs from CA.
IR	Information Retrieval.
KB	Knowledge Base.
LISP	LISt Processor: a logic-based programming language.
MG	Membership Grade: associated with GRiST's concept hierarchy.
NLP	Natural Language Processing.
POS	Part Of Speech: e.g. nouns, verbs, adjectives and adverbs.
RTO	Risk To Others.
SQL	Structured Query Language: for querying relational databases.
WLD	Weighted Levenshtein Distance.
XML	eXtensible Mark-up Language: a format for digital information.
XSLT	Extensible Style-sheet Language Transformations: formats raw XML.

Dedicated to my children, Henry, Emily and Lucy,

and to my parents, Barry and Barbara.

Acknowledgements

Sincere thanks to Dr. Chris Buckingham, for his supervision and for sharing the mind maps around which this thesis revolves. Thanks, too, to Dr. David Evans, and to fellow student and friend, Abu Ahmed, for help, tea, and sympathy. Last, but not least, to my family for their unstinting support and encouragement during my studies.

Contents

List of Figures

List of Tables

Part I

Extracting Knowledge from GRiST Mind Maps

1

Defining the Problem Domain

The domain of this research is the problem of assessing risks posed by people suffering from mental health problems. As a starting point, this chapter reflects on the many people in the U.K. affected by mental health problems, and who face further difficulties after legislation focussed more on care in the community, rather than in hospitals. That move made risk assessment a very important topic, especially given the media attention received by the few high-profile cases of mentally ill people becoming violent, or even committing murder.

The GRiST project, though, helps to overcome the difficulties that non-specialists encounter in deciding when to refer such patients to expert clinicians. That project, in part, used the technique of mind mapping to record knowledge from mental health experts. Rather than spending considerable manual effort on refining the resulting mind maps, though, this thesis seeks to process them automatically. In that respect, certain challenges arise, solutions to which are given in an overview of this work. Firstly, then, to introduce the incidence and types of mental health problems in the United Kingdom.

1.1 Mental Health Problems in the U.K.

Mental health problems cause enormous distress to individuals and families alike. Along with the obvious human costs comes an appreciable financial burden; just by itself, England annually spends over £12 billion on mental health care (Social Exclusion Unit, 2003), suggesting that many people in Great Britain have mental disorders of some kind. Indeed, depression and anxiety affect one in six adults in the United Kingdom; a further five in every thousand people are afflicted by schizophrenia or by bipolar affective disorder, sometimes called 'manic depression'. Those illnesses are known collectively as personality disorder, or psychopathic disorder; people suffering from such conditions are more likely to commit serious violent offences (ONS, 2000).

On the other hand, psychological problems might result in self-harm or even suicide; in fact, seventeen in every hundred thousand men in the U.K. killed themselves in just one year, with a third as many women doing so (British Psychological Society, 2002). Given a U.K. population of around sixty million, that amounted to over six thousand people committing suicide in a single year. In just one city, Bridgend in Wales, twenty young people died at their own hand between 2007 and 2008, with that total continuing to grow during the course of this thesis (Sky News, 2008). Nationwide, many more people inflict lesser forms of so-called para-suicide, including cutting themselves, taking overdoses, and pulling out hair (British Psychological Society, 2002).

In 1990, though, the NHS and Community Care Act changed the nation's approach to handling mental health problems. That year, the care of people suffering from severe psychiatric disorders devolved

away from hospitals towards local services. As a result, patients find it harder to obtain the necessary psychiatric help (Mueser, Drake, & Resnic, 1997). Indeed, Discharges from hospital in response to the Act are seen as premature and inappropriate, with local services forced to assess people who might formerly have been in institutions. As a result, people that constitute mental health risks might present at front-line agencies such as the police and emergency services, which generally lack any expertise for making assessments. Experienced risk managers, in contrast, coordinate long term treatment, rather than participating in day-to-day care (Buckingham et al., 2004). Instead of service users disappearing with their difficulties unacknowledged, a risk-screening process is required that empowers front-line agencies; that would allow non-specialists to determine the nature of any risks, and whether referrals to specialists are needed (Eastman, 1997).

The NHS and Community Care Act, then, moved emphasis from care in institutions to care in the community. As a result, the problem of assessing risks posed by service users, both to the public at large and to themselves, has become more pressing. In that light, there now follows an overview of attempts to quantify risks arising from mental health problems, after which an existing project is introduced that helps non-specialists to make risk assessments. That project recorded knowledge from mental health specialists by means of mind mapping; variation in expressing ideas within those mind maps motivates this thesis. This chapter closes with a preview of forthcoming chapters that rectify that problem.

1.2 Risk Analysis in Mental Health

Risk analysis, then, serves to identify people that present a danger either to themselves or to others. To that end, risk analyses identify cues that are correlated with such risks; some cues, though, are better predictors than others. With regard to violent behaviour in people diagnosed with personality disorder, patients' abuse of alcohol or of street drugs increases the risk of injuring other people. Even so, a history of violence best predicted any Risk To Others (RTO). In a similar vein, previous self harm better predicted suicide than did, say, unemployment (British Psychological Society, 2002). In children, a risk of self harm arose from cues that included bullying, parental neglect, serious physical illness, and physical or sexual abuse. Risk analysis, then, must consider patients' lives and lifestyles, in addition to any specific medical conditions (The Children's Society, 2008).

Analysing mental health risks bears comparison with predicting volcanic eruptions; the problem in respect of the latter lies in gauging the probability of a certain type event, of a given magnitude, occurring within a specified period of time. The answer lies in considering combinations of cues that, by themselves, do not suggest any immediate risk (Booth, 1979). That well expresses the problems facing community care, and reflects a similar approach to resolving them. Such multi-faceted problems demand identifying

combinations of cues, so as to head off eruptions either in volcanoes or in people.

A further difficulty arises, though, in that the most important cue for any given risk might, in fact, be absent. Indeed, just a small proportion of people classed as RTO have been violent in the past. So-called false negatives arise from assessing such cases, when low-risk patients actually become violent. Further, public reaction to high-profile murders by mental health patients has led to avoiding false negatives at all costs, which accordingly reduces the threshold for intervention. Conversely, false positives raise problems regarding civil liberties: people free of mental disorder might yet become violent if, say, intoxicated. It would, though, be wrong to incarcerate such people in mental institutions pending further investigation (Petch, 2001).

All the same, it is not just extreme violence against other people that is of concern; alleviating self-harm and general misery are valid reasons for wanting to analyse risk factors. An additional complication, though, is that any particular cue might correspond to several types of risk; front-line services, then, face a difficult task in discerning cues that suggest a need for more detailed, specialist assessment. For that reason, the actual need is for a risk screening tool that helps non-specialists to determine what combinations of cues justify such referrals. Such a tool would identify patterns of cues integrated into a single, accurate risk prediction (Buckingham et al., 2004).

Although it is essential that mental-health practitioners make risk assessments, that skill yet lacks a precise scientific basis. Although partly due to low incidences of suicide and violence among service users, factors influencing clinical judgements are inherently difficult to study. So-called actuarial approaches to predicting risks, though, emphasise enduring static factors, rather than any dynamic ones that are, in fact, more important. Further, pattern of cues, rather than any particular one, better predict mental health risks. For example, young single mothers present combined cues of gender, parental status, marital status, and age; such combinations are better indicators of risk than should any particular cue taken in isolation (Buckingham, Adams, & Mace, 2007). Having introduced the role of risk analysis in mental health care, then, attention now turns to a project that elicited experts' knowledge of risk assessment.

1.3 The GRiST Project

This thesis builds on a joint project between the universities of Aston and Warwick that addressed the difficult problem of screening mental health risks. To that end, the Galatean Risk Screening Tool (GRiST) helps front-line services to interpret cues presented by patients. Inspired by Galatea, the mythical perfect woman sculpted by Pygmalion, GRiST focuses on hypothetically 'perfect' representatives of given classes of risk; galateas, then, allow the dissemination of expert advice about risk assessment to front-line services. By integrating patterns of cues, GRiST gives a single, accurate risk prediction which, in turn, promotes the earlier and more accurate detection of patients at risk (Buckingham et al., 2004).

With respect to service-users, GRiST aimed to promote earlier detection of mental health problems, and to improve any ensuing risk assessment. That, in turn, should lead to more timely interventions and better targeted referrals to specialists in the field; indeed, GRiST might be used for self-assessment. Overall, GRiST promotes long-term improvements in mental health status and social integration (Buckingham, 2007).

Knowledge held in galateas came, in fact, from a multidisciplinary panel of about fifty full-time mental health clinicians, including nurses, general practitioners, social workers, psychiatrists, and psychologists, in response to a letter from Buckingham and Adams (2005) inviting participation in GRiST. Galateas improved on existing approaches by indicating the likelihood of any risk actually resulting from observed cues. That was done by means of Membership Grades (MGs) varying from 0 to 1, indicating respectively that a particular cue never, or always, predicted a corresponding risk. The galatea for suicide, for example, would comprise all cues whose largest MG was associated with that risk; alternative models, on the other hand, would seek a typical personal profile that predicts suicide (Buckingham, 2007).

Galateas, then, represent 'perfect' combinations of cues associated with particular risk factors, including self-harm, suicide and RTO. In that way, galateas constitute a database of client cues and associated risk judgements, based on knowledge provided by practitioners as part of their clinical practice (Buckingham et al., 2004). Knowledge encapsulated in that way was initially gathered by questionnaire, the current version of which is available on the web site for The GRiST Project (2007). Subsequent less structured information, though, came from transcripts of interviews with mental health specialists. Qualitative results from each such interview were further recorded as mind maps, by means of a free software package called FreeMind (Buckingham & Adams, 2006). The forty six resulting mind maps constitute the domain of this thesis, which a subsequent overview of GRiST put in context of the overall project in Figure 1.1. That diagram depicts the stages involved in collecting and refining knowledge about mental health risks:

Figure 1.1: A diagram of the GRiST project (Buckingham, Ahmed, & Adams, 2007)

In fact, the diagram of GRiST from Figure 1.1 reflects two main stages, the first of which identified low-level cues that might be recognised by people lacking a mental health background, while the second stage quantified associations between those cues and particular risks (Buckingham, 2007). Figure 1.1 shows that first stage, then, to start with transcripts of interviews with GRiST panellists; those transcripts were subsequently coded as mind maps, which appear in the top-left corner. Forty-six such interviews with mental-health practitioners were subsequently integrated into a single mind map, which was subsequently transformed by the LISt Processor (LISP) into an alternative tree structure. That initial hierarchy, though, was too large, and was savagely pruned by software that used Extensible Style-sheet Language Transformations (XSLT) due to the limited time available to panellists; in fact, experts just monitored any proposed cuts, and annotated the pruned knowledge structure with their opinions by means of web-based software (Buckingham, Ahmed, & Adams, 2007).

The tree from the first stage of GRiST, then, identified information that might be collected while assessing people at risk. Concepts higher up the resulting pruned tree reflected concordance between higher numbers of experts than did more detailed concepts lower down that hierarchy. Following that first phase, the second stage in developing GRiST quantified associations between cues and risk factors, which started with the data-gathering tree shown at the bottom-right corner of Figure 1.1. Further, responses to questions attached to any identified cues prompted additional improvements, and yielded the final knowledge hierarchy. XSLT transformed that tree structure on demand, to report just relevant associations for particular users. The resulting knowledge structure showed, for example, that previous episodes of risk behaviour by service users impacted all risks save self neglect; in contrast, experts saw self neglect as almost wholly dependent on current circumstances (Buckingham, Ahmed, & Adams, 2007).

That same overview of GRiST gave the combined mind map in Figure 1.2 overleaf, which integrated those created in FreeMind by particular experts; note that the number of experts raising any concept is encoded in each node, to give a rough measure of its importance. Further, nodes enclosed by borders, and appended by a small circle, indicate hidden sub-hierarchies available on clicking such 'folded' nodes in FreeMind (Buckingham, Ahmed, & Adams, 2007):

Figure 1.2: Detail from a GRiST mind map (Buckingham, Ahmed, & Adams, 2007)

Mind map nodes from Figure 1.2 reflect observable cues for the top-level risk of suicide, which further comprises underlying sub-concepts such as any past client episodes. That concept, in turn, branches to patterns of episodes, both in terms of when they happen, and in respect of any changes over time. In that way, mind mapping allowed GRiST to display precisely how low-level cues are assessed by mental health experts in terms of their impact on top-level risk categories. That approach, then, provided standards for governing the collection and the storage of information, and offered a universally comprehensible risk language. By means of web-based resources, mental health expertise encoded in GRiST is available to all interested parties (Buckingham, 2007). Because mind mapping was so important to GRiST, that technique is now discussed in more detail.

1.4 The Technique of Mind Mapping

Although information is commonly presented as text, people differ in the ways that they prefer to absorb information. While 'sensing', for example, emphasises concrete, practical thinking, the 'intuitive' mode reflects a preference for abstract, innovative thinking. In particular, people might prefer visual representations such as pictures and diagrams (Felder & Spurlin, 2005). Indeed, mind mapping is a strong form of such a visual style. For people having alternative learning styles, though, mind maps might not be the best way to present information. That said, human viewers are not of prime importance, here; rather, it is the way that GRiST used mind mapping as a precursor to a more formal representation of machine-readable knowledge. All the same, the background to that technique is important in understanding such knowledge structures.

When reflecting on human thought processes, key ideas have been envisaged as spheres, which are covered in hooks for attaching related themes. The way that hooks issue from the entire surface of any sphere gave rise to the term 'radiant thinking'; mind mapping was invented to record ideas in that way, as a non-linear alternative to taking ordinary notes. Although originally drawn freehand on paper, mind maps can nowadays be created on personal computers by means of, say, the iMindMap™ package sold on the inventor's web site. A mind map from that web site appears overleaf as Figure 1.3; it was created in iMindMap™ to describe that very web site (Buzan, 1974, 1996, 2003, 2008):

Figure 1.3: A mind map devised by Buzan (2008) of the iMindMap™ web site

The mind map from in Figure 1.3 conforms to rules dictated by the technique's originator; specifically, a 'true' mind map requires that:

1. images and colours should be used, in addition to words;
2. a single Basic Organising Idea (BOI) should appear at the centre;
3. each branch should contain just a sole key image or word;
4. branches should be connected to form a nodal structure;
5. the resulting structure should be hierarchical (Buzan, 1974, 1996, 2003).

Mind maps created by the GRiST project, henceforth simply called 'GRiST mind maps', will shortly be discussed in relation to those rules. That will require an appreciation of how those rules yield a 'perfect' mind map such as that from Figure 1.3, above. The overall need for colours and images is clearly met in that mind map. The single BOI specified by rule two is a composite image of Buzan and the World Wide Web, WWW. Single related concepts such as software, books and training then radiate from that key idea, satisfying rule three. Rule four demands a structure based on nodes, which occurs in Figure 1.3 as ideas branch into more specific concepts. Such branching additionally fulfils rule five, which requires a hierarchy of nodes. By conforming to rule five, the mind map in Figure 1.3 constitutes what Sowa (1992) describes as an inverted tree structure.

Broadly speaking, GRiST mind maps follow those rules: using recognised mind mapping software helped to ensure that. Buckingham and Adams (2006) actually employed an open-source programme called FreeMind (Polansky & Foltin, 2010) in preference to proprietary tools such as iMindMap™. The screen-shot in Figure 1.4 shows how ideas about risk analysis were captured as a mind map in FreeMind. That mind map represents the transcript of an interview with one of forty six mental health specialists, who was assigned the number 16 to ensure anonymity. Nodes highlighted with shading in Figure 1.4 overleaf will shortly help to explain some key terminology:.

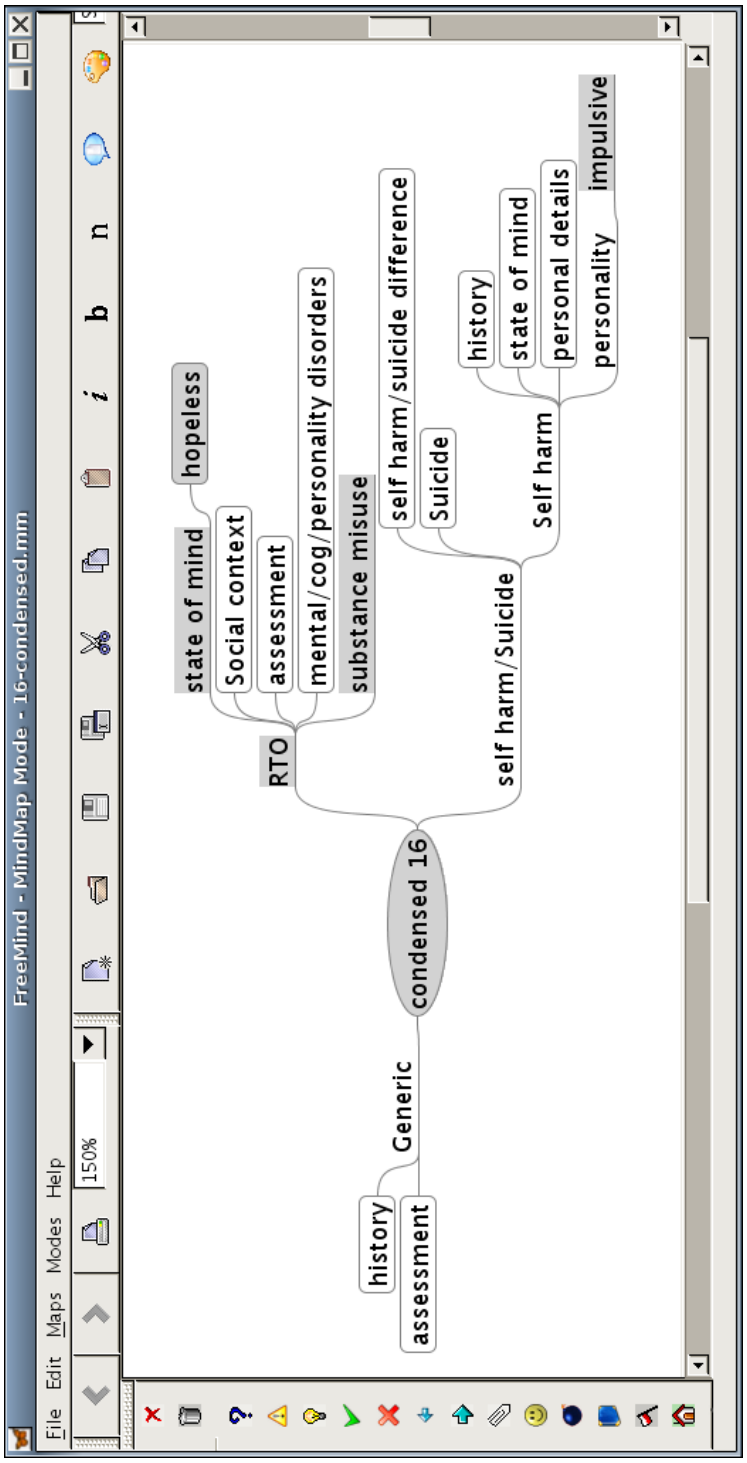

Figure 1.4: A mind map from GRiST

In FreeMind, the terms 'node' and 'concept' refer to the text in any branch, and will be represented thus: `[a mind map node]`. The BOI at the centre of any mind map is the root node, or simply the root. Various main risk categories radiate from the root in Figure 1.4; one such concept expresses Risk To Others, `[RTO]`. That so-called parent node branches into various children, such as `[state of mind]`, which in turn is the parent of `[hopeless]`. In that respect, a path comprises the nodes that must be traversed from the root to reach any specific node. Along with `[hopeless]`, two further shaded nodes, `[substance misuse]` and `[impulsive]`, are called 'leaf nodes' due to having no branches. The GRiST mind map in Figure 1.4, then, appears to conform to the rules of mind mapping, being a hierarchy of nodes radiating from a central root. Closer inspection, though, reveals differences of varying importance, as shown next.

1.5 Challenges Posed by GRiST Mind Maps

An important difference between the GRiST mind map from Figure 1.4 and the ideal from Figure 1.3 is a lack of images and colours. Such lack of adherence to the first rule of mind mapping, though, is irrelevant; rather than mind maps as an aid to memory, or for making presentations, GRiST mind maps serve as a precursor to a database of mental health knowledge. The emphasis here, then, is on allowing machines, rather than humans, to retrieve and process knowledge from mind maps. In that respect, just concepts expressed as words are important; omitting the images required by rule one is of no consequence. Neither is the lack of adherence to rule two, which demands a single BOI; the root node `[condensed 16]` is not really a BOI, but a label in the emerging database. Rather, it is the single-word concepts demanded by rule three that are of crucial importance; concepts in GRiST mind maps often contain several words, contravening that requirement. In illustration, long nodes from the sub-hierarchy under `[RTO]` that were hidden earlier in Figure 1.4 are revealed overleaf in Figure 1.5:

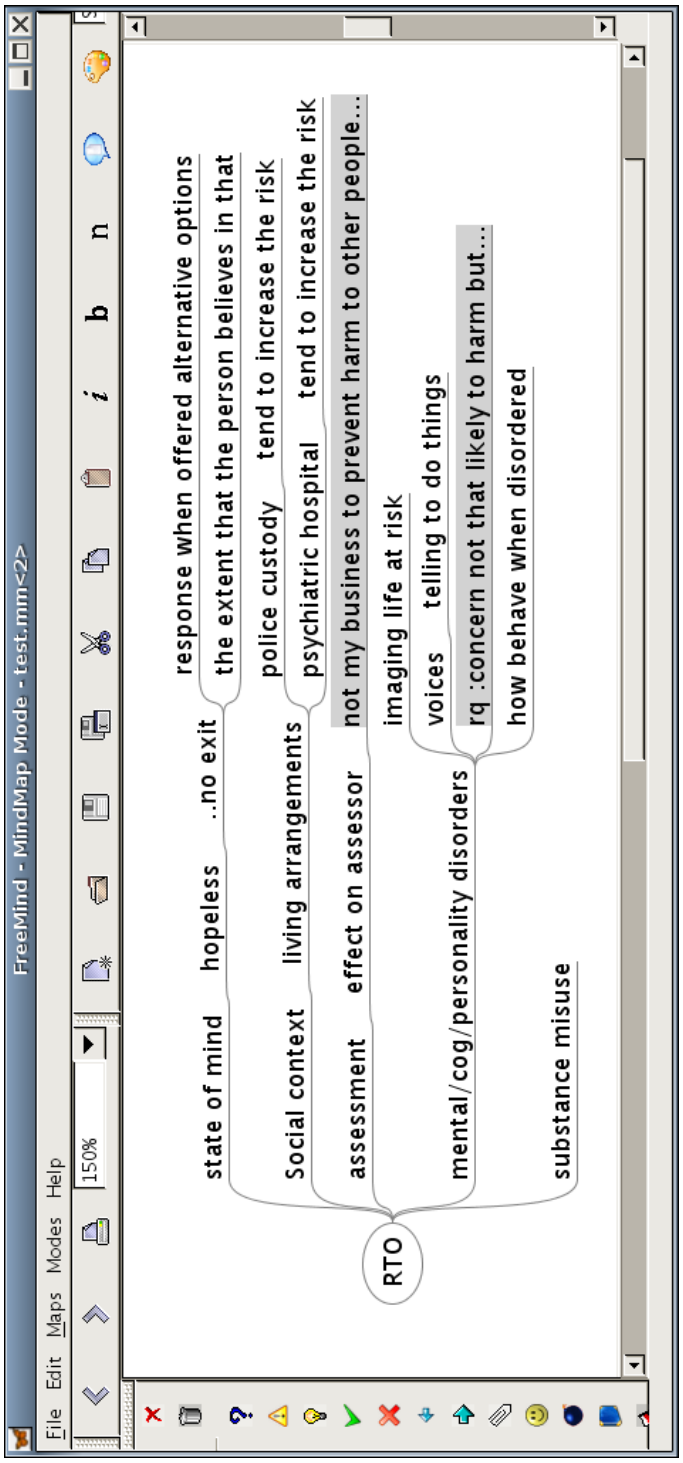

Figure 1.5: Detail from a mind map from GRiST

The mind map from Figure 1.5 shows that some 'concepts' in GRiST mind maps were actually quite complex ideas. While such richness provided GRiST researchers with a wealth of information about mental health risks, it hinders any fully automated approach to interpreting mind maps. Indeed, the two nodes highlighted in Figure 1.5 contained so many words that they are shown truncated. Even in shorter nodes, several concepts might contribute to an overall idea. This thesis does not aim to reconfigure such nodes, as the manual refinement reported by Buckingham and Adams (2006) had already rendered them into an acceptable form. Accordingly, the root of the mind map from Figure 1.4 contained the word 'condensed'.

Rewriting phrases as single words would, in fact, reduce mind maps to a Bag-Of-Words (BOW) that fails to represent semantic relationships (Bekkerman & Allan, 2005). In that light, insisting on single-word nodes risks losing information. Although key concepts in GRiST mind maps must be identified, the aim here is to map them, rather than to reconfigure individual nodes. Indeed, that rule is counter-intuitive, as detail of the 'perfect' mind map from Figure 1.3 shows; that concerns the idea of news being 'hot off [the] press', as Figure 1.6 demonstrates:

Figure 1.6: Detail from Figure 1.3 of the 'perfect' mind map devised by Buzan (2008).

The separate nodes `[hot]`, `[off]` and `[press]` from Figure 1.6, though, do not capture the intended meaning. Indeed, insisting on single-word nodes would require humans and machines alike to reconstruct any original meaning, during which nodes, and accordingly ideas, might be recombined incorrectly. Permitting more expressive nodes avoids that problem of interpretation for humans, and faithfully represents intended ideas. Nodes such as `[hot off the press]` should be permitted, as those words collectively represent a specific idea; the same argument applies to nodes in GRiST mind maps.

Although GRiST allowed such multi-word nodes, the transformed knowledge tree had to be pruned in order to remove redundant nodes, and to ensure that structure's integrity; that reflected principles from designing relational databases (Buckingham, Ahmed, & Adams, 2007). That association might, in fact, be taken further; indeed, mind maps and galateas are seen here as belonging to a wider family of 'semantic networks' that are refined by a process of normalisation (Mylopoulos, 1998). Although GRiST researchers started that process, mind maps from that project might themselves be normalised further; that would encourage their use as formal repository of knowledge, rather than just as precursors to the database of mental health knowledge comprised by galateas.

Richer ideas in mind map nodes, though, hinder any automated interpretation. While optimal for viewing by humans, such longer nodes hide key concepts from machines; such concepts must, then, be isolated by researching words in various ways. In fact, what might seem a trivial problem detracts from such lexical analyses. Specifically, spelling mistakes mean that concepts might be overlooked; the problem deepens on examining responses from a freely available spelling checker; inappropriate corrections that arise would be misleading if allowed to pass unchallenged.

A further obstacle to analysing mind maps automatically concerns the various forms that words take; nodes might contain related words that are hidden from machines by textual variations. The term *stem* will be used from this point on to denote such short forms. The process of stemming, then, involves mapping words to some base form (Brants, 2003). Determining the longest sub-string that identifies related words, while excluding others, will lead to groups of words that express any particular underlying concept. Stems derived on linguistic principles, though, are restricted to specific languages; the desire here is for a more flexible approach based on textual similarities alone. Such an ability to quantify differences between text strings will further help machines to select appropriate spelling corrections.

In addition to stemming related word forms, the actual meanings of words affect the treatment of concepts from GRiST mind map nodes. Distinctions between, say, nouns and verbs indicate whether concepts represent things or actions, respectively. Unfortunately, the tool chosen for analysing words in this thesis cannot perform that task reliably; ambiguous words, in particular, impede attempts to determine exact concepts. To overcome that problem, a novel approach is applied to deciding appropriate word-usage automatically. That approach analyses words that appear immediately before and after certain prepositions; subsequently, patterns in the distribution of known actions and things around those prepositions will help to resolve ambiguous cases. However, the term 'heuristic' best describes applying reliable knowledge in that way: the search is for guidelines, rather than for absolute rules.

A further problem concerns structural variations between individual GRiST mind maps. In fact, the template used to create mind maps evolved over time, and took several iterations to stabilise. Ultimately,

agreement between three independent researchers exceeded 90% on the template categories (Buckingham & Adams, 2006). While that evolutionary process helps to explain differences in hierarchical structure evident in GRiST mind maps, it remains a challenge in respect of those collected mind maps as a formal semantic network. In addition to repeating identical nodes at various levels in those mind maps, nodes that express similar concepts often do so in slightly different wording. Processing mind maps automatically, then, demands that machines reconcile such variations.

In order to overcome that problem, the lexical tool already mentioned will be combined with a multivariate statistical analysis; resulting clusters of related nodes emerging from that analysis will identify nodes bearing words of similar forms, or of related meanings. Those clusters indicate node hierarchies for automatically generated, idealised mind maps. The result will constitute a refined information base, as well as being a more formal representation for human researchers. Rather than producing a sole, static combined mind map, though, the proposed approach generates such idealised node hierarchies on demand. New mind maps will be created automatically to represent the expression of particular concepts across the collection of GRiST mind maps, by means of an enhanced FreeMind interface. Having described the domain of this thesis, then, the overview that follows describes the contents of forthcoming chapters.

1.6 Research Questions to be Answered

The overall aim here is to provide a much-needed theoretical framework for mind mapping, and to overcome or justify the aspects of GRiST mind maps just addressed in Section 1.5. Currently, mind maps enjoy little attention as formal representations of knowledge, and are relegated to personal use, for "brainstorming", and for making presentations. The primary research question, then, asks whether mind maps might be viewed collectively as a type of database. Should that be the case, mind mapping would be allowed into the family of recognised formats for machine-readable knowledge that are evident in existing research.

Further questions, though, arose during the early part of this research. An initial scan of GRiST's collection of mind maps revealed spelling mistakes that would detract from the primary aim of refining those mind maps into a formal database. It was assumed that existing spelling checkers would easily perform the desired corrections, but preliminary experiments showed several such algorithms to perform quite badly. Clearly, that had to be addressed in order to determine reliable concepts, on which the new database of mind maps would be built.

A further difficulty hindered that endeavour, namely, that of identifying related forms of words; any emerging knowledge structure would best treat related word forms as instances of a common underlying concept. That process, called "stemming", identifies related word forms while keeping apart any unrelated ones. However, attempts at stemming encountered problems similar to those found for spelling correction, in this case that existing algorithms depended on embedded linguistic knowledge. That posed the further research question of how stemming might be performed on a purely textual basis, with no prerequisite for knowledge of any particular language.

Yet another research question arose from initial attempts to have machines derive the meaning of words from GRiST mind maps. The tool selected for researching words, WordNet, suffered from ambiguity and from a restricted coverage of English. That, in turn, hindered the identification of specific concepts, and of different word forms that yet have related meanings. Treating mind maps as a database demands greater certainty about the meanings of words.

The final research question addressed in this thesis concerns variations in how GRiST's panel of experts expressed ideas. Each of the 46 mind maps at the heart of this research was created in isolation by a particular individual. Subsequently, the GRiST project subjected those mind maps to intense manual reformulation, which yielded an consensual view of mental health risks, and any associated personal, behavioural and social indicators evident in patients. Such a high degree of manual intervention begs

the question of whether that process of refinement might be automated. Making that possible, though, depended on answering the questions raised already.

Here, then, is a summary of the research questions to be answered in subsequent chapters:

- *Can a theoretical framework be provided for mind mapping?* That is the overriding aim of this thesis: to raise the humble status of mind mapping in Knowledge Engineering, by discovering ways of formalising knowledge held as mind maps.
- *Is it possible to improve on existing spelling correction algorithms?* Automated spelling correction commonly fails to recognise less common words, such as the medical terminology found in GRiST mind maps. Corrections offered by existing checkers are often ludicrous, as any user of word processors might have found. Is there, then, a way to better select appropriate corrections, and to recognise words that, although missing from any dictionary employed, yet exist as valid English words?
- *Is it possible to improve on existing stemming techniques?* Existing algorithms depend on linguistic knowledge embedded by developers. Instead of limiting such "stemmers" to specific languages, it would be much better should machines deduce related word forms without resort to linguistic rules.
- *What can be done to resolve the ambiguity evident in WordNet?* Although WordNet is a powerful and ubiquitous tool, it has definite limitations. In particular, ambiguity presents a major hurdle to having machines determine exact meanings for words used by human authors.
- *How might the structure of mind maps be refined automatically?* In sharp contrast to accepted digital formats for representing knowledge, no control is exerted over the structure of mind maps. If mind mapping is to be treated as a formal representation of knowledge, the resulting variation in expressing ideas must be overcome.

The degree to which those questions were answered will be reviewed at the end of this thesis, in the summary provided by Chapter 11. For now, there follows an overview of the topics covered by subsequent chapters.

1.7 Thesis Overview

Items in the following list describe the chapters comprising this thesis; for each chapter, a brief overview of the topics to be covered is offered:

Part I: Extracting Knowledge from GRiST Mind Maps.

- Chapter 1: *Defining the Problem Domain.* This introductory chapter.
- Chapter 2: *A Theoretical Framework for Mind Mapping* addresses mind maps in the wider context of 'semantic networks' such as concept maps and relational databases. As the poor relation of that family, though, mind maps are confined to gathering knowledge for inclusion in more formal semantic networks. In sharp contrast, GRiST mind maps are seen here as constituting an information base of metal health knowledge, in turn making them amenable to the process of 'abstraction' by which related representations are refined. Applying abstraction to GRiST mind maps yields a format for storing knowledge in addition to any inherent hierarchical associations.
- Chapter 3: *Spelling Correction for GRiST Mind Maps* deals with spelling mistakes that detract from identifying key concepts in GRiST mind maps. A review of automated spelling correction, though, reveals that 'non-words' might be valid words that standard dictionaries happen to lack. In fact, support for taking words as presented, or for determining acceptable corrections, will come from words encoded in GRiST mind maps themselves. Spelling corrections will be further refined by means of an algorithm widely used for comparing texts: the Levenshtein Distance, L. Applications of and refinements to L suggest an adjusted version, which experiments show to improve spelling correction for GRiST mind maps, in addition to revealing valid, novel terms.
- Chapter 4: *Stemming for GRiST Mind Maps.* Locating key concepts in GRiST mind maps further involves identifying related words, at first by means of a process called 'stemming'. That process identifies any invariant portion of related word forms, often by means of rules specific to particular languages. In contrast, the Levenshtein Distance that Chapter 3 applied to spelling correction will further help in extracting stems from mind map nodes, though without linguistic rules; experiments subsequently show the utility of that approach to extracting key concepts from GRiST mind maps.

Part II: Applying Knowledge to GRiST Mind Maps.

- Chapter 5: *Resolving Ambiguity in GRiST Mind Maps.* Key concepts related by virtue of shared stems are complemented by researching the WordNet lexical database. That reveals related meanings between words from GRiST mind maps, rather than related forms. After an overview of WordNet, certain challenges in using it are raised. The first is that WordNet covers just certain types of words, which is easily overcome by existing lists of such words. A more difficult problem remains, though, in that WordNet might report competing interpretations of any given word. In that respect, a technique called 'clustering' is proposed for resolving ambiguous words, especially in conjunction with prepositions. Resulting nouns and verbs further reveal relationships between subjects, actions, and objects intended or inferred by GRiST panellists.

- Chapter 6: *Clustering Concepts from GRiST Mind Maps* describes the actual tool employed for clustering, which Chapter 5 introduced as a means of resolving ambiguity in WordNet; that tool is Correspondence Analysis (CA). After describing the underlying mechanisms of CA, which draws on classical mechanics, studies are reviewed of applying CA to researching plain text. Those studies suggest a way to determine patterns of word usage around prepositions, which will subsequently resolve ambiguous cases in GRiST mind maps. Interpreting CA results, though, is generally a human activity; in contrast, automating CA for GRiST mind maps allows machines to derive such patterns unaided.

- Chapter 7: *Experiments in Resolving Ambiguity* presents results from applying the approach from Chapter 6 to overcoming problems posed by WordNet, described in Chapter 5. Experiments start by identifying 'triples' of words, each triple comprising a word, followed by a preposition, with a further word after that. CA on triples composed of just unambiguous words reveal reliable patterns of usage, which subsequently suggest most likely interpretations of ambiguous words from further, novel triples. Instead of words themselves, though, CA involves so-called 'meta types' that reflect roles that words play in phrases, as 'things', as 'actions', or as 'modifiers'. That knowledge further augments the hierarchical associations inherent in mind maps, enriching the emerging information base of mental health knowledge.

- Chapter 8: *Refining the Structure of GRiST Mind Maps* is concerned with generating idealised, combined versions of the collected GRiST mind maps, from nodes identified by an amalgamation of CA, stemming, and WordNet; the ensuing mind maps provide overall knowledge structures for particular concepts. After drawing related meanings from WordNet, the chapter describes the hierarchical nature of CA clusters that contribute nodes to any combined mind map. Experiments in refining mind map structures first address just nodes grouped by stems; subsequently, those mind maps are supplemented by semantically related nodes identified by WordNet.

Part III: Implementation, Summary and Conclusions.

- Chapter 9: *Mind Maps, Metadata and an XML Database* introduces the eXtensible Markup Language (XML) in which FreeMind encodes mind maps. Because of that, a native XML database is used to store both GRiST mind maps and any knowledge gathered by experiments in earlier chapters; in fact, such additional knowledge constitutes 'metadata' that describe existing data encoded in mind maps. That database further responds to queries from both machines and humans, about either type of data.
- Chapter 10: *Benefits to GRiST* demonstrates the gains accruing from this thesis. Such benefits include extensions to FreeMind that communicate with the XML database, and provide a work space for GRiST researchers. One such extension generates refined mind maps on demand, while a further one aids navigation between related nodes. Importantly, metadata that support any automated decisions are made available to humans. The chapter closes by considering the implementation of this approach in Java and XML.
- Chapter 11: *Summary and Conclusions* ends this thesis by offering a summary of the approach taken here to normalising and enriching GRiST mind maps. Accordingly, the conclusion is that GRiST will benefit from this research, as will the application of mind mapping in general.

1.8 Chapter Summary

This first chapter introduced risk assessment in mental health care, and the GRiST project that disseminates expert knowledge to front-line services. That research initially used mind maps to capture ideas about mental health risks; those mind maps were discussed in terms of rules proposed by the inventor of the technique. It was argued that mind maps be considered more formally as a type of semantic network, and might benefit from techniques used to refine, say, relational databases.

To that end, refinement starts by identifying specific concepts, by means of a popular textual comparison algorithm that will further aid in refining spelling corrections. An additional obstacle to determining precise concepts is ambiguity; sometimes, the exact meaning of a word cannot be deduced by linguistically-based approaches. That will be overcome by a multivariate analysis of words surrounding various prepositions, from which will arise heuristics for deciding the precise meaning of ambiguous words.

The final problem raised in this chapter was that of structural variation in GRiST mind maps. Any idealised structure will demand breaking down existing nodes and reassembling them into novel hierarchies, which is accomplished by a combination of textual, linguistic and multivariate analyses. Resulting clusters form the basis of combined mind maps that best represent particular concepts across the GRiST collection. The present chapter ended with an overview of the remainder of this thesis.

2

A Theoretical Framework for Mind Mapping

2.1 Of Mind Maps and Semantic Networks

Mind maps, then, comprise a single root node that diverges into a hierarchy of concepts. That notation, in turn, describes what Sowa (1992) calls a *semantic network* that represents knowledge as hierarchies of inter-connected nodes. In fact, semantic networks arose as early as the third century AD, when a Greek philosopher gave his name to the Tree of Porphyry. An English version of that early semantic network appears in Figure 2.1:

Figure 2.1: The Tree of Porphyry, from Sowa (1992).

Figure 2.1 shows the concept Substance as the top of a hierarchy, with increasingly specific concepts appearing at progressively lower levels. At each level, 'differentiae' allow for opposing attributes that concepts might express. Near the bottom of the tree comes the concept Animal, which encompasses both rational and irrational types. In that way, humans and beasts alike are animals; humans, though, are rational ones, while beasts are irrational. Importantly, concepts inherit differentiae from succeeding levels in the hierarchy. In addition to being rational, then, humans are sensitive, animate, and ultimately, a material Substance (Sowa, 1992).

In fact, galateas created by Buckingham et al. (2004) might be seen as a further type of semantic network, based on knowledge collected from mental health experts. Such knowledge concerned the risks that patients pose both to themselves and to other people, as well as the cues associated with those

risks. For example, the galatea below in Figure 2.2 reveals cues that were deemed to indicate a risk of self-neglect:

Figure 2.2: A galatea for self neglect, from Buckingham et al. (2004).

Figure 2.2 decomposes the risk of self neglect into its constituent elements, which include mental state and diet. Although non-specialists might identify some of those cues unaided, GRiST contributes by showing how experts organise ideas about mental health risks. Identifying patterns of cues is a major advantage of the GRIST approach that helps, say, general practitioners to make risk assessments. To that end, the GRIST web site grants access to the knowledge held in galateas to front line services[1]. The galatea just presented as Figure 2.2 suggests that a client with a dietary problem, for example, might be at risk of general self-neglect. Observing related cues, such as a lack of compliance with medication, would compound that concern. Several indicators occurring together might precipitate specialist intervention.

That depends on the likelihood of a risk actually resulting from observed behaviour. Indeed, more than one type of risk may be associated with a single cue. That problem was overcome by assigning a Membership Grade (MG) to each cue in a galatea. MGs reflect the degree of association between any cue and a particular risk, indicating the most likely interpretation for any combination of cues. That, in turn, yields a single, accurate risk assessment (Buckingham et al., 2004; Hegazy & Buckingham, 2008).

The Galatean approach was complemented by mind mapping as an alternative way of gathering expert knowledge (Buckingham & Adams, 2006). Indeed, the aim here is not to supplant galletean semantic networks, nor yet to reproduce them. Rather, it is the mine of non-hierarchical knowledge in those mind maps that is of particular interest. All the same, hierarchical knowledge will be reflected in standardised structures, resembling the combined mind map created from those of individual mental health experts.

[1]See `http://www.galassify.org/grist/development/docs/grist-v3.pdf`.

In fact mind maps and galateas alike have much common, as shown next by considering mind maps in relation to more formal structures called 'semantic networks'.

2.1.1 Mind Maps in Relation to Semantic Networks

Mind maps from GRiST, then, are to be addressed in terms of alternative ways of storing knowledge. Such representations will provide a more formal framework for mind mapping; applying that framework will help to identify and organise related concepts from GRiST mind maps. The first such related format, concept maps, are addressed next, after which more formal representations are introduced.

Mind Maps in Relation to Concept Maps

Mind mapping, then, bears comparison with more formal approaches to representing knowledge. One such alternative technique is called concept mapping, which Cañas and Carvalho (2004) see as closely related to mind mapping. Indeed, the recurring word 'mapping' suggests the similarity of those two forms. Take, for example, the concept maps reproduced overleaf in Figure 2.3 that depict alternative meanings of the word 'chair'. The concept map to the left denotes chairs as furniture, and describes relationships with concepts such as 'room' and 'table'. The concept map to the right, though, gives an academic interpretation of 'chair' that involves the related concepts of 'department' and 'professors':

Figure 2.3: Concept maps for two meanings of 'chair', from Cañas and Carvalho (2004).

Relationships from concept maps in Figure 2.3, then, appear as annotated links between nodes. Such links in the concept map of household chairs from Figure 2.3 showed that rooms 'have' tables and chairs, which 'are' furniture, while rooms 'are part of a' building. From an academic viewpoint, though, a chair 'is one of the' professors that, along with a secretary and students, belong within a department. Despite that slight improvement over mind mapping, concept maps suffer from a lack of formalism (Cañas & Carvalho, 2004). By implication, that drawback affects mind maps to a greater degree.

Concept maps about chairs from Figure 2.3, though, do resemble mind maps in that they comprise nodes connected by branching lines. On the other hand, those concept maps highlight an important difference, in that they permit relationships resembling the differentiae from the Tree of Porphyry. In that respect, the sole similarity between mind maps and concept maps is a shared hierarchical nature; general concepts appear around the top of any hierarchy, and more specific concepts toward the bottom. Mind maps, though, lack any cross-links by which concept maps depict related areas of thought (Novak & Cañas, 2006).

Concept maps about chairs from Figure 2.3, then, differ from mind maps in being well annotated. That, indeed, is one of the benefits of concept maps: they are unbound by the invented rules of mind mapping. Those rules insist that branching should occur just downwards in any mind map; cross-linking between nodes is forbidden. Neither do mind maps allow annotated links (Buzan, 1974, 1996; Novak & Cañas, 2006). Obeying that rule would disallow the lateral link in Figure 2.3 which declares that a chair 'is one of the' professors. Now, although concept maps might be enriched by labelled links, those from Figure 2.3 might be considered too long. Indeed, it has been argued that labels should be kept to just one or two words (Novak & Cañas, 2006).

In contrast to such specific relationships allowed by concept maps, mind map nodes are just associatively linked (Sure et al., 2002). In other words, unlabelled branches in mind maps indicate merely that one concept is associated with another. Such criticism over mind maps' associative nature, though, is too severe. Admittedly, mind maps cannot describe the nature of relationships between nodes. All the same, node hierarchies in mind maps reflect dependencies between concepts, just as they do in concept maps. By means of an appropriate node hierarchy, any mind map could reflect, say, that chairs belong in rooms rather than rooms in chairs.

Mind Maps in Relation to Formal Representations of Knowledge

A crucial difference between GRiST mind maps and the concept maps from Figure 2.3 is the use of well-defined concepts. Whereas those concept maps comprised single-word nodes, those from GRiST mind maps express ideas as phrases, or even as complete sentences. That goes against the tendency for representing knowledge as discrete classes evident in concept mapping, or indeed, in the Tree of Porphyry.

In fact, that tree has been described more formally as a definitional network of *is-a* relationships between concept types and sub-types. In that sense, a human *is-a* animal. Known alternatively as subsumption hierarchies, such semantic networks emphasise inherited properties between concepts and sub-concepts. Indeed, more recent forms of semantic networks expressed in Description Logics (DL) closely resemble the Tree of Porphyry (Sowa, 1992).

Subsumption hierarchies allow machines to derive structured knowledge from an otherwise unstructured soup (Sowa, 2006). While not detracting from the rich content of GRiST mind maps, they do resemble such a soup in terms of any formal representation of knowledge. Taking that analogy slightly further, the aim here is to classify the ingredients of that soup after having served it. To that end, attention turns next to how semantic networks couched in DL might reveal an approach to handling mind maps.

That more recent type of semantic network, DL, has been described as a formalism for representing knowledge. So-called Knowledge Bases (KBs) constructed from DL hold subsumption relationships in a machine-readable format. Rather than dictating subsumption in a rigid tree such as Porphyry's, DL actively derive sub-concepts by examining corresponding instances in the KB. For example, a user might want to determine if concept C is a sub-concept of D, which would be clear from a fixed structure such as the Tree of Porphyry. In DL, though, subsumption must be proved by inspecting instances of concepts C and D. Should all instances of C in addition qualify as instances of D, then subsumption has been proved, and C is a true sub-concept of D; class C is fully contained within class D, that is, $C \sqsubseteq D$. The more general concept D is the subsumer, while sub-concept C is the subsumee. Subsumption then, involves checking all instances of C and D within any KB; only then can an answer can be provided (Nardi & Brachman, 2002)[1].

Because retrieving information from a KB is such an active process, response times could be slow. That problem might be alleviated by default rules that impose subsumption directly, removing the need to deduce it from scratch. The former example of subsumption between concepts C and D, then, could be dictated when creating any KB, rather than deduced at run-time (Nardi & Brachman, 2002). In that way, we have come full-circle from subsumption in a strict tree-like structure, by way of dynamically derived relationships, to an alternative dictated form.

2.1.2 Instances of Mind Mapping in Knowledge Engineering

Mind maps receive sparse treatment as formal repositories of knowledge, and have even been considered on a par with Excel™ spreadsheets as a means of capturing knowledge from domain experts. Although useful in early stages of gathering knowledge, mind maps and spreadsheets alike are seen as having limited capacity for capturing any problem domain (Nagypál, 2007). In that respect, mind mapping is

[1]Note that encountering an instance of C that is not an instance of D would prove subsumption false.

seen as best suited to so-called brainstorming, as a means of exchanging and developing ideas. That said, integrating mind maps with better controlled knowledge representations is an attractive idea (Sure et al., 2002). The consensus is that although mind maps cannot formally represent knowledge, they might yet serve as useful precursors.

Mind maps, then, are seen as unsuited to holding knowledge in support of reasoning by machines. Mind mapping, it is said, is incapable of representing knowledge structures perfectly and completely. Indeed, mind maps might hold knowledge in any form that domain experts find acceptable (Biplab, Wallace, Wallace, & Gill, 2008). In fact, a similar problem arose in creating mind maps for GRiST (Buckingham & Adams, 2006); chapter 1 noted similar variation between participating mental health experts. Before introducing a novel approach that helps to raise the status of mind mapping, attention turns to studies that have used that technique to whatever degree.

Mind Mapping in relation to OntoEdit, an Ontology Editor

The first such study concerns OntoEdit, a tool created by Sure et al. (2002) that has import and export facilities for transferring knowledge to and from mind maps. OntoEdit, though, holds knowledge as atomic concepts, and loads just such mind maps that comprise similarly well-defined ideas. In that way, mind map node hierarchies might be exchanged between equivalent structures within OntoEdit. Specifically, branching mind map nodes are taken to express concepts and sub-concepts (Sure et al., 2002). To illustrate, Figure 2.4 depicts overleaf a mind map that is loaded into OntoEdit:

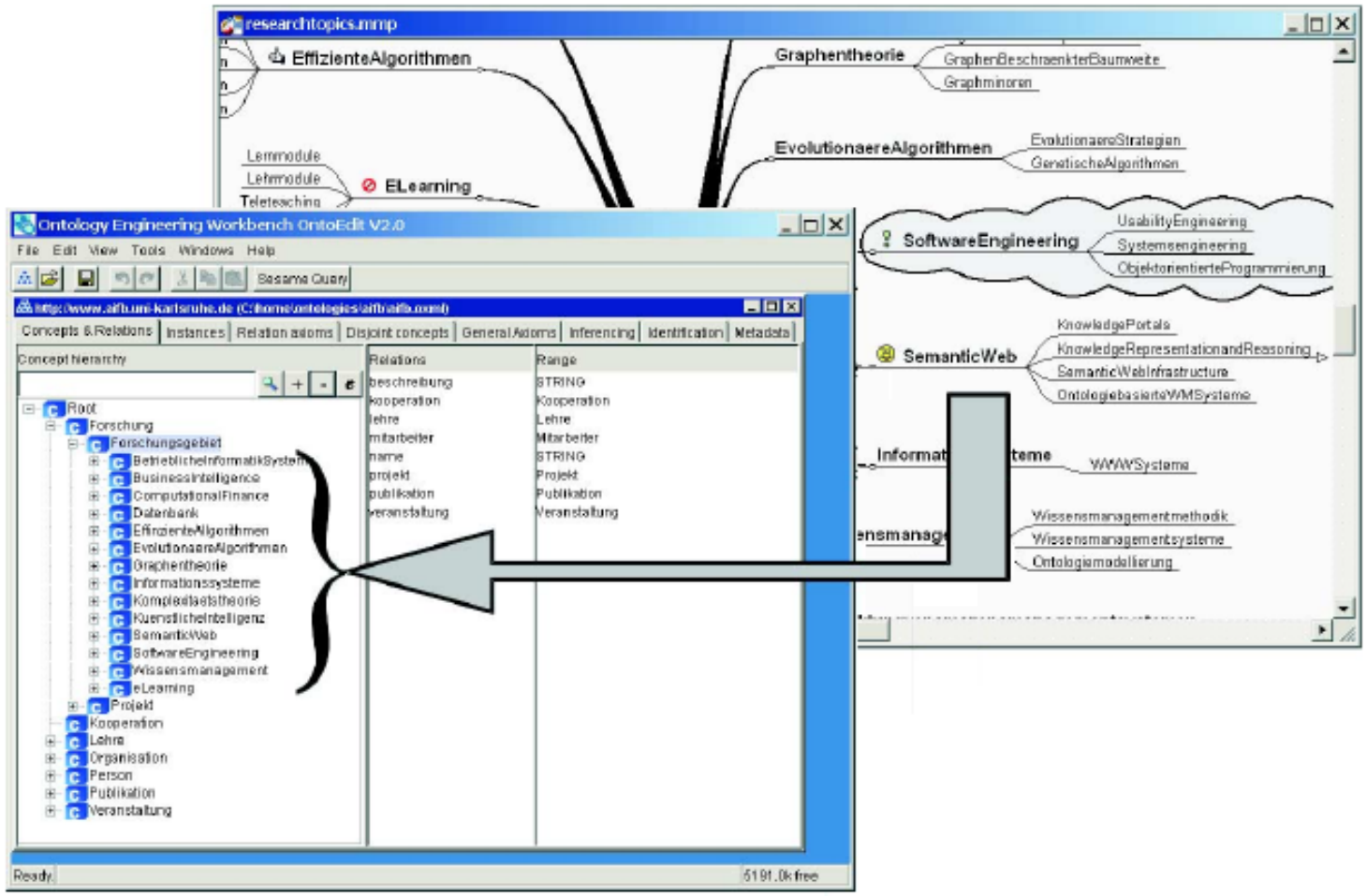

Figure 2.4: Loading a mind map into OntoEdit, adapted from Sure et al. (2002).

Nodes from a mind map in the background of Figure 2.4 were loaded into an OntoEdit concept hierarchy in the foreground. In fact, both mind map and ontology from that figure express atomic concepts that resemble database fields. In marked contrast, GRiST mind maps created by Buckingham and Adams (2006) recorded expansive knowledge about mental health risks. Such relatively verbose ideas could not readily be loaded into OntoEdit, due to its dependence on atomic concepts. A general term for such well-structured knowledge is the word 'ontology', which is defined as an explicit conceptualisation of a domain. In other words, ontologies explicitly describe whatever concepts and relationships that might represent knowledge from a particular area of interest (Gruber, 1995). Indeed, from that word 'ontology' comes the 'Onto' in OntoEdit.

The strict concept hierarchy in any ontology, though, cannot be inferred from mind map nodes; such nodes are seen as no more than associated. That weak relationship meant restricting mind mapping to the early stages of building a stricter representation. Although mind maps proved useful for capturing relevant knowledge, OntoEdit's import and export facilities constituted just valuable add-ons (Sure et al., 2002). A separate OntoEdit report describes an ontology that arose from manually refining knowledge recorded as mind maps. That knowledge resulted from brainstorming sessions which required consensus from so-called domain experts (Sure, Staab, & Studer, 2002). In that respect, the process was similar to the one described for GRiST by Buckingham and Adams (2006).

Mind Mapping for Brainstorming Ontologies

Mind mapping for brainstorming a problem domain is a recurring theme. Work by Reich, Brockhausen, Lau, and Reimer (2002) used such an approach to agree basic concepts for an ontology about skills management. That ontology embodied a systematic approach to identifying and classifying skills within the work force at Swiss Life, an insurance company. During early phases of development, concept hierarchies were refined in mind maps by means of a proprietary tool called MindManager™. Mind map nodes were reorganised in that editor by drag and drop operations; such refinement was necessary due to mind mapping tools' failure to check for duplicate nodes. Mind maps created in that way contained atomic concepts well suited to a more formal ontology (Reich et al., 2002). In that respect, such concepts resembled those loaded into OntoEdit by Sure et al. (2002).

The topic of managing human resources recurs in a study by Hefke and Stojanovic (2004), in which mind maps recorded answers to a questionnaire about company employees' skills. Knowledge from those mind maps was transferred manually to a Requirements Specification Document (RSD); concepts isolated in that RSD were subsequently loaded into OntoEdit. In that way, mind maps constituted precursors to an ontology (Hefke & Stojanovic, 2004). Figure 2.5 next shows detail from the resulting skills ontology:

Figure 2.5: A concept map of competencies and skills, from Hefke and Stojanovic (2004).

Figure 2.5 reveals an obvious difference between mind maps and OntoEdit ontologies, in the latter's use of attributes such as 'hasCompetence', which were emphasised by darker shading. That ontology subsequently formed the backbone of a skills requirements system, and supported users' requests regarding the competences of employees. Should an appropriate employee be available, just that person would be reported. Otherwise, employees were ranked by the degree to which they matched the required competences (Hefke & Stojanovic, 2004).

Mind Mapping for Visualising Ontologies

A novel approach to visualising ontologies by means of mind maps was taken by Lin, Wei, Lee, and Lee (2005). The ontology in question served to display site maps, as an aid to website design. Websites having dynamic content, though, could not be mapped in advance. Instead, site maps were generated on demand by means of an ontology. So-called knowledge objects in web-pages were further associated with elements from the ontology, which described the structure of any site map. Now, although the ontology in question comprised a concept map, mind maps were offered as an output option. A 'Mindmap' function on a user interface invoked FreeMind to print sub-hierarchies of concepts from the main ontology (Lin et al., 2005).

In fact, various ontology editors use mind maps as an aid to visualising knowledge structures, although a review of five such editors was disappointing. Even for such limited display purposes, support for mind mapping in those editors was rated as poor (Biplab et al., 2008). Existing studies, then, offer little insight into treating GRiST mind maps as knowledge hierarchies. Whereas nodes in mind maps created by Buckingham and Adams (2006) were expressive and complex, mind maps devised by Hefke and Stojanovic (2004) used well-defined concepts, as did those created by Sure et al. (2002) and by Reich et al. (2002).

Mind maps then, when used at all, comprised just inputs to, or outputs from, any ontology. As input, brainstorming by means mind maps contributed concepts to a more formal ontology. As output, mind maps exported from OntoEdit presented ontologies in a readily comprehensible way. Lin et al. (2005) further offered a function that used FreeMind to print concept sub-hierarchies from a main ontology. In contrast to that lowly status, mind maps in themselves will be shown to constitute an information base of mental health knowledge, as discussed next.

2.2 GRiST Mind Maps as an Information Base

In fact, semantic networks and relational databases have been collectively termed information bases. Within any such information base exist so-called atoms that stand for generic concepts (Mylopoulos, 1998). That overlap between relational databases and semantic networks leads, in turn, to comparisons with mind maps, DL, and concept maps. Having brought mind mapping under a more general heading of semantic networks, it follows that mind maps might, in some way, be treated as information bases. Approaches to refining information bases of whatever type should indicate ways to organise knowledge from GRiST mind maps. Attention turns next, then, to established principles that are applied to designing and using information bases.

2.2.1 The Characteristics of Information Bases

As Sowa (1992) points out, knowledge bases differ from relational databases in terms of organising knowledge. All the same, those formats share marked similarities. Consider, for example, the rote memory that underpins most important computer systems. Such systems rely on exact records stored largely in relational databases, although networks might fulfil that role. A further essential feature of both types of repository lies in retrieving data that are similar, yet not identical (Sowa, 1992).

Databases and knowledge bases further suffer a fragility in respect of changes in any data recorded (Sowa, 2006). A further similarity arises between ways that those information bases organise information. In any KB, so-called default rules allow subsumption relationships to be imposed directly. Procedural rules, on the other hand, govern inferences about existing KB individuals, which leads to new facts (Nardi & Brachman, 2002). In an analogous way to default rules, indexes within relational databases provide more direct access to any underlying information (Mylopoulos, 1998). In addition, procedural rules perform a function that resembles Structured Query Language (SQL) within a relational database.

Similarities between relational databases and semantic networks, then, permits viewing them from a common theoretical standpoint (Mylopoulos, 1998). Practical applications provide further evidence of that close relationship. Relational databases were, in fact, used to store galatean knowledge for the GRiST project (Buckingham et al., 2004). In the realm of DL, performance problems have been overcome by holding individuals from a KB in relational database. That approach avoided the repeated computations required to identify instances of concepts (Horrocks, Li, Turi, & Bechhofer, 2004). Indeed, closer integration of knowledge base principles with relational theory is to be encouraged. Rather than using those repositories side by side, a consolidated methodology might treat them as identical (Debenham, 1996).

The term ‘information base’, then, well suits relational databases and semantic networks alike. Such

networks, in turn, include mind maps, DL, and concept maps. Indeed, the KB of competencies and skills created by Hefke and Stojanovic (2004) was stored as a concept map. Although Section 2.1.1 raised labelled links as a difference between concept maps and mind maps, approaches to validating the former might yet help in refining the latter. That, in turn, bears comparison with techniques for formalising database and knowledge base structures.

Mind Mapping in respect of Concept Mapping

There are, in fact, two ways of building concept maps of any domain. So-called normative concept maps hold just single instances of specific concepts, giving complete consensus over any domain. Conversely, descriptive concept maps allow differing representations of a single domain. In fact, normative and descriptive approaches are complementary, and provide a comprehensive view from different perspectives. All the same, a descriptive approach is preferable to demanding a single correct concept map (Albert & Steiner, 2005).

Descriptive concept maps, then, need not agree on representations of any domain, bringing to mind the variation between GRiST mind maps from the domain of risk analysis in mental health. Then again, the combined mind map for GRiST created by Buckingham and Adams (2006) fulfilled a role that Albert and Steiner (2005) describe for normative concept maps, in providing a single correct view of the domain in question. That raises the possibility of treating GRiST mind maps in a similar way to concept maps.

While abundant descriptive knowledge exists in both individual and combined mind maps from GRiST, there is less of the normative variety. Take, for example, the problem of duplicated nodes raised by Reich et al. (2002); atomic concepts such as 'abuse' recur throughout the combined GRiST mind map. That is contrary to the cross-domain consensus expressed by any normative concept map. A further lack of normative knowledge arises from what Biplab et al. (2008) noted as mind maps' ability to take any form whatever, as long as they are hierarchical. In addition, both descriptive and normative concept maps are physical entities, as are GRiST's individual and combined mind maps. That two separate yet complementary concept or mind maps co-exist is important to the more fundamental distinction discussed next.

Mind Mapping in respect of Relational Databases and DL

Normative and descriptive concept maps, then, might exist side by side. A further dichotomy exists, though, between the idea of any information base and its actual embodiment. In that respect, atoms in any information base stand for generic concepts, while concrete applications comprise individuals, or tokens, of such generic concepts (Mylopoulos, 1998). In a similar way, relational database terminology distinguishes between relations and tables: relations are theoretical constructs that correspond to tables in any actual database. Individual relations, designed for a particular purpose, are known as relation

variables, or relvars. At a more detailed level, relations are comprised of tuples, whereas actual tables contain rows or records[1]. In relational databases, all such intensional knowledge is stored in so-called schemata (Date, 1975, 2003).

A corresponding separation exists in DL. The distinction there is between intensional knowledge about any problem domain, and extensional knowledge specific to a particular problem; the form that extensional knowledge might take is regulated by intensional knowledge. What are called a TBox[2] and an ABox respectively hold intensional and extensional knowledge in a KB (Nardi & Brachman, 2002). As do relational schemata, DL TBoxes dictate what form records might take within an information base; mind maps, in contrast, are subjected to no such control.

The Lack of Normative and Intensional Knowledge from GRiST Mind Maps

GRiST mind maps have already been shown to resemble descriptive rather than normative concept maps. In a similar vein, those mind maps constitute extensional, rather than intensional, knowledge about risk assessment. Indeed, the very nature of mind mapping rejects any notion of intensional knowledge. Such freedom of expression led to the variations in structure and in content noted in Chapter 1. Although GRiST researchers agreed on categories for a template mind map, compliance could not be ensured; the FreeMind tool had no facility for enforcing such a template (Buckingham & Adams, 2006). Any lacking intensional knowledge in GRiST might, in fact, be added retrospectively, by noting techniques that aid the design of information bases. That will involve the dual processes of abstraction and normalisation, by which key concepts might be identified in GRiST mind maps.

2.2.2 Applying Abstraction to Information Bases

Designing information bases entails a process called abstraction, which suppresses irrelevant detail in order to emphasise what generic concepts, or classes, have in common. The first stage in abstraction, classification, yields generic concepts that constitute intensional knowledge. Those concepts, or atoms, govern the creation of individuals in an information base (Mylopoulos, 1998). Subsequently, the second stage of abstraction applies a technique known as normalisation, which is described as putting one thing in one place. With respect to relational databases, that means passing through successive stages called normal forms. Those forms progressively refine any database's design, with each level assuming that preceding forms have been achieved. The first normal form, 1NF, operates on tuples, while remaining forms address entire relations (Date, 2003).

[1] Terms such as 'tuples' and 'record' are often, though incorrectly, interchanged (Date, 1975).
[2] The 'T' in TBox may be taken as either 'Terminology' or 'Taxonomy' (Nardi & Brachman, 2002).

Normal Forms

Achieving 1NF involves excising redundant information from individual tuples. That problem arises when fields in any tuple comprise lists of values known as repeating groups. Because tuples must contain atomic fields, attaining 1NF decomposes such groups into single-field tuples. In terms of relational calculus, any tuple x that contains concepts $c_0 \dots c_n$ is expressed as $x \to \{c_0,\ c_1 \dots\ c_n\}$. In that expression, x is the determinant, while $\{c_0, c_1 \dots c_n\}$ are dependants of x. Such multiple concepts within a sole tuple must be restated as singletons[1]. For the current example, that would result in $x \to c_0$, $x \to c_1 \dots x \to c_n$. The relationship between determinant and dependants from 1NF is, then, one-to-many (1:M). In that way, a single value of x associates various atomic concepts held as separate tuples (Date, 1975, 2003).

While 1NF addressed multiple classes within a single tuple, the second normal form (2NF) deals with duplicated fields within whole relations. Once 1NF is achieved, 2NF ensures that such fields arise just once. Say that a relvar for holding shipping orders stores a city for any delivery address. Now, cities might be specified for every dispatch note in the database. That, though, raises problems of accuracy and conformity; spelling errors or a poor grasp of geography might yield multiple names for any particular location. Eliminating such duplication would ensure that city names conformed to recognised values, and improve the accuracy of dispatches. Individual city names would be stored just once in a separate relation, and be assigned an unique code. Any shipping address would subsequently need just that code, with actual names coming from the newly created relation. Such codes in the main shipping orders relation are called foreign keys, which correspond to a primary key of the code from the city names relation (Date, 2003).

Having attained 2NF, the third normal form (3NF) ensures that columns in any relation depend on the key, the whole key and nothing but the key (Date, 2003). For example, the destination of any dispatch depends solely on the associated order number. Should orders further require the name of a delivery firm, 3NF would insist on recording that separately; delivery details depend more on availability than on any particular order number.

Applying Normal Forms

In fact, normal forms that refine relational databases apply to information bases in general; removing redundant or duplicated fields enhances performance and avoids making conflicting updates (Mylopoulos, 1998; Date, 2003). An analogue of 2NF for knowledge bases can be illustrated by means of concepts A and B, which the axiom $A \equiv B$ equates as synonyms. That equivalence makes redundant the axiom $B \equiv A$; all occurrences of B, in fact, can be replaced by concept A, imposing uniformity across any KB (Tsarkov, Horrocks, & Patel-Schneider, 2007). Indeed, normal forms introduced by Date (1975) apply

[1] Singletons may be represented without enclosing braces (Date, 2003).

as much to rules in knowledge bases as they do to relations in more traditional databases. In contrast, knowledge that is left un-normalised remains unnecessarily hard to understand (Debenham, 1996, 1998). Attention turns now to overcoming the sparsity of intensional knowledge in GRiST mind maps by means of abstraction.

2.2.3 Applying Abstraction to GRiST Mind Maps

This thesis addresses the lack of intensional knowledge in GRiST mind maps, and in mind mapping generally. Intensional knowledge, though, usually precedes extensional knowledge; abstraction refines the design for any information base before creating actual records (Date, 1975; Mylopoulos, 1998). Conversely, abstraction here will applied to existing extensional knowledge in GRiST mind maps, to derive what might be called reverse-engineered intensional knowledge.

Any resulting intensional knowledge will, though, be stored separately from those mind maps, in a more traditional type of database. That arrangement more resembles the adjacent relational and knowledge-based stores used by Tsarkov et al. (2007) than it does ontologies from Sure et al. (2002) or the dual normative and descriptive concept maps of Albert and Steiner (2005). Such a separate store of intensional knowledge will allow existing mind maps to act as a normalised information base. Although normalisation improves IR performance, it is the organisation imposed that is important here.

Normalising Knowledge from GRiST Mind Maps

Atomic concepts from the first stage in abstraction, classification, are of particular interest here. The goal, though, is not to make GRiST mind maps conform to some abstract representation; that would reflect the extreme reductionism of the BOW approach rejected in Chapter 1, along with the insistence by Buzan (1996) on single-word nodes. That is not to deny the importance of atomic concepts, but to argue against rewriting individual nodes as isolated words. Rather, narrowly-defined concepts held as intensional knowledge will mediate access to extensional knowledge in mind maps. Classifying important words from mind map nodes will reveal where recurring ideas appear; atoms permit references to specific ingredients of what Sowa (2006) might call mind map soup.

The second phase of abstraction, normalisation, builds on the results of classification to yield the desired intensional knowledge. The question arises, then, of what intensional knowledge might be gleaned from GRiST mind maps? Remembering that Sure et al. (2002) view branches in mind maps as mere associations, intensional knowledge must first reflect such associations. Just a single atom accomplishes that: a unique identifier, from here on called $nodeID$. A tuple of that atom is expressed in relational calculus as $nodeID \rightarrow nodeID$. Actual records from that tuple represent a link between two GRiST nodes having the specified values of $nodeID$. GRiST nodes, though, regularly contain several important words, constituting what Date (2003) called repeating groups. Applying 1NF overcomes that with a

further tuple, $nodeID \rightarrow concept$, where instances of *concept* hold particular ideas from a specific node.

Tuples such as $nodeID \rightarrow concept$ reflect the lossless nature of normalisation; discrete concepts might be addressed individually, as a set by means of the $nodeID$ field, or as the full node text. No data have been discarded; indeed, much new information will have arisen. Such intensional knowledge permits queries against mind maps, which is a characteristic of information bases. For example, a user might ask for nodes that express a given concept, which in turn might be augmented by nodes containing related concepts. Individual nodes, then, will largely remain intact; they express ideas in a way that Buckingham and Adams (2006) saw as optimal for human interpretation. Intensional knowledge arising from normalisation, though, will promote a more detailed automated analysis of those mind maps.

Treating nodes as composed of tuples further suggests viewing mind maps as analogous to relations, which comprise such tuples. Whereas 1NF operates at the level of tuples, 2NF removes redundant or duplicated fields from relations. Following 2NF, foreign keys from any relation correspond to primary keys in a separate relation. That approach will be taken for nodes that express related yet specific concepts anywhere in GRiST's mind maps. A representative will be selected from such related nodes, expressed in tuples of the form $nodeID \rightarrow nodeID$. Any given representative node appears to the left of the arrow; those to the right are equivalent in meaning. The result is very much in the spirit of what Date (2003) called putting one thing in one place.

The Benefits Normalising GRiST Mind Maps

Queries made against mind maps would benefit from retrieving just representative nodes. Performance, though, is a minor issue in face of opportunities arising from intensional knowledge held in 2NF. One such opportunity involves entire paths rather than individual nodes. Recall that paths denote branches leading from any mind map's root node to lower-level nodes. Normalisation, then, allows paths comprising $nodeID$s of representative nodes rather than actual node texts. In that way, paths expressing related concepts at equivalent levels in any hierarchy yield identical sequences of $nodeID$ values. That, in turn, allows direct comparisons between paths.

Such directly comparable paths further allows machines to identify nodes that are inserted or deleted between any two paths. That, in turn, will allow paths to be refined into a standardised knowledge structure of hierarchical mind map nodes. Inserted or deleted nodes, though, yield less common hierarchies, due to a lack of intensional knowledge that might have imposed an agreed template. In such cases, nodes form differing hierarchies that depict related meanings might be refined into a single, representative version by means of 2NF. The lossless nature of that process, though, will leave any original mind maps intact; such revised hierarchies will be reflected in intensional knowledge, held separately.

In addition, sorting normalised paths of node keys will group together paths that employ similar nodes; further eliminating duplicate paths yields a list of unique paths, which will be reassembled automatically into a combined knowledge structure. That, in turn, alleviates the manual task endured by Buckingham and Adams (2006). Such novel mind maps will contain just a single instance of any particular path, with any related paths integrated by a degree of reconfiguration. The third normal form, 3NF, ensures that any fields in a tuple depend solely on the key of the containing relation. The process of classification suggested a tuple for representing related words in the form $concept \rightarrow concept$. That tuple lacks a $nodeID$ field: the meaning of any specific word does not depend on what node contains it. The outcome of applying 3NF is similar to that obtained from meeting 2NF, in that separate relations are indicated for holding data in a more usable form.

2.3 Identifying Related Concepts

The process of abstraction, then, refines and optimises intensional knowledge, which regulates what might be stored in any information base, and in what manner. Classification starts that process by identifying any atoms needed to hold such intensional knowledge. Subsequently, successive stages of normalisation determine structures of such atoms. The first stage, 1NF, reorganises repeating groups within any tuple, or record. After that, 2NF and 3NF refine relations that store such tuples. Field in any tuple, though, must hold just atomic values; mind map nodes, in contrast, often express several concepts. Such nodes might be split into constituent words held as separate nodes. That BOW approach, though, was rejected in Chapter 1 because of the ensuing loss of meaning.

A Normalised View of Concepts from GRiST Mind Maps

Rather than rewriting mind maps nodes in that way, tuples of the form $nodeID \rightarrow concept$ represent key concepts as atoms of intensional knowledge. A further tuple, $concept \rightarrow concept$, allows for words that express related meanings. In fact, two types of related words are considered here. The first type concerns morphological variations that yet share a common sub-string, from here on termed a *stem*. Such word variations arise from appending pre- and suffixes to stems. Words that lack any common stem, though, might yet be related. Although completely different in terms of text, people recognise such words are synonymous. Machines must emulate that human knowledge in order to discern relationships between such words.

It is important that machines concur with humans when identifying related words. From a morphological viewpoint, over-short stems will group together words that are not truly related. Conversely, stems that are too long will fail to wholly identify related words. Machines, then, must determine stems of optimum length. Although so-called stemmers exist, they suffer disadvantages that must be overcome,

such as being designed for handling specific languages. An analogous problem arises from analysing word meanings, in that words might be just remotely related. Given that tools exist to detect the meanings of words, the challenge lies in determining the degree of any reported synonymy. Concepts related by morphology or by meaning will, though, be held as stems rather than as actual words. Records based on the tuple $nodeID \rightarrow concept$ will group together nodes that express morphological variations of a particular concept. Records from the $concept \rightarrow concept$ tuple will further be populated by stems to indicate synonymy between entire groups of words.

Spelling Mistakes in GRiST Mind Maps

An further aspect of GRiST mind maps, though, detracts from identifying the required atoms; misspellings and a sometimes specialised mental-health vocabulary obscures human authors' intended meanings from machines. Although absent from any digital dictionaries, it would be wrong to treat such uncommon words as spelling mistakes; indeed, that they are used by specialists makes them particularly important. Even for true mistakes, any spelling checker might suggest inappropriate replacements which, if accepted, would introduce false knowledge. Machines, then, face two problems. The first is whether novel words should be treated as they stand, or corrected by means of a spelling checker. That second option raises the further problem of determining appropriate suggestions from any offered.

A Tool for Refining Spelling Corrections and for Stemming

Having introduced problems that machines face in processing GRiST mind maps, attention now turns to one of the tools chosen to overcome those problems. In fact, acceptable stems and spelling corrections alike will depend on measuring any similarity between words. To that end, the chosen measure is the Levenshtein Distance, referred to as L. Before describing that algorithm in detail, though, the next chapter starts by looking at spelling errors in more detail. After explaining the origins of spelling mistakes, an overview of existing research raises various issues concerning approaches to automated correction. Following that comes a proposal for handling spelling errors in GRiST mind maps.

Themes encountered during spelling correction will recur in respect of stemming. Chapter 4, then, continues by describing approaches to deriving stems from isolated words. After that, studies are reviewed that account for context during stemming, before turning to stemming for GRiST mind maps. Both spelling correction and stemming will rely on the Levenshtein Distance, to which various refinements have been made in the past. In particular, the effect of word length on any judged similarity must be taken into account. Such an adjustment for use in analysing GRiST mind maps closes that chapter.

2.4 Chapter Summary

This chapter, then, placed mind maps in the context of semantic networks, and described various approaches that used mind mapping in knowledge engineering, albeit to a limited degree. By dint of being semantic networks, GRiST mind maps were further seen as a potential information base. That led to considering just what characterises information bases; in particular, the process of abstraction was seen to improve any design. Accordingly, abstraction was considered in respect of GRiST Mind Maps, in an attempt to identify related concepts. This summary, then, closes the chapter, and attention moves to a means of resolving unrecognised words found in GRiST mind maps.

3

Spelling Correction for GRiST Mind Maps

3.1 Introduction

One of the problems facing any automated analysis of GRiST mind maps, then, concerns spelling mistakes made by mental health specialists. Unaided, though, machines are prone to offering inappropriate, and sometimes amusing, corrections. That is because such suggestions are mere character strings that resemble any spelling error. A contrasting problem further arises should so-called spelling errors, in fact, be words that are missing from whatever dictionary is researched. This chapter, then, presents an approach to correcting spelling errors in GRiST mind maps, while retaining valid, though unrecognised, words. That will involve assessing any similarity between spelling errors and suggested corrections.

Before addressing mistakes from GRiST, though, an overview describes current approaches to automated spelling correction in general. That review starts by identifying the causes of spelling errors, before presenting approaches to automated correction. In that respect, various challenges arise in detecting and resolving spelling mistakes. Meeting those challenges will involve a measure called the Edit Distance, which quantifies similarity between errors and any proposed alternatives. In addition to the standard algorithm, though, adjustments to the Edit Distance commonly allow for words of differing lengths. After giving examples of studies that do that, the particular adjustment proposed by this thesis will be presented. That measure, in turn, will be shown to greatly improve spelling correction for GRiST mind maps. Now, then, to the review of approaches to automated spelling correction in general.

3.2 An Overview of Automated Spelling Correction

As users of the Internet and of word processors might well be aware, spelling corrections offered by machines are sometimes inappropriate, or even bizarre. For example, spell-checking the LaTeX source of this thesis offered 'parsnip' as a replacement for the mark-up command '\parskip', and 'shortcake' for the citation mark-up, '\citeA'. While not suggesting any built-in culinary bias, accepting such 'corrections' automatically would have been completely wrong. Those suggested corrections are quite dissimilar to the intended words, but were the best that the spelling checker could find. With regard to GRiST's mind maps, such inappropriate replacements would undermine any emerging knowledge structures. A more discerning approach must be sought if those mind maps are to be checked automatically for spelling errors.

Kukich (1993) sees correcting errors automatically as much harder than simply identifying them. The so-called morphological productivity of the English language, though, hinders both of those processes; new words regularly enter the language, while existing ones leave it. That nouns are often 'verbified' illustrates

that tendency - for example, the noun 'Balkans' yielded the novel verb 'balkanisation'. While interactive spelling checkers allow users to evaluate suggested corrections, automating that process requires machines to take such decisions unaided. For that reason, existing spelling correction techniques are of limited scope and accuracy (Kukich, 1993).

In fact, three challenges are inherent in automated spelling correction. By ascending difficulty, the first concerns detecting non-words that are strings missing from any dictionaries used. That is not to say, though, that such non-words are invalid; in fact, therein lies the challenge: is any particular non-word a spelling error, or a valid yet uncommon word? Having identified non-words, the second challenge arises of correcting them in isolation. That might involve overcoming the third challenge of considering the context of any misspelled words. Most existing techniques are said to ignore that last and most difficult problem of context, concentrating instead on isolated misspellings. That approach, though, ignores information that might be gleaned from the linguistic or textual context of misspelled words (Kukich, 1993).

3.2.1 Detecting Non-Words

The simplest approach to automated spelling correction is to search lists of acceptable words. Non-words are those that are missing from such electronic dictionaries, and are at first treated as invalid, pending further evidence to the contrary. That approach, though, suffers increased response times when searching larger dictionaries. A compromise must be struck, then, between the richness of such dictionaries, and the time taken to search them. A further problem is that removing non-words fails to offer possible valid interpretations, and disregards what might otherwise be important information (Kukich, 1993).

Despite increased response times, it is tempting to assume that larger dictionaries will reduce the incidence of non-words. Unfortunately, a second problem arises from that approach: that of accuracy. Unsuitable matches have been shown to rise from 10% for a 50,000-word dictionary to almost 16% for one of 350,000 words. That reflects an increasing probability of any misspelled word inappropriately matching a dictionary entry, especially for shorter misspellings (Kukich, 1993). Clearly, suggested corrections must be analysed further before they are accepted. Rather than abandoning non-words or replacing them by inappropriate corrections, as many as possible valid novel words must be retained.

3.2.2 Correcting Isolated Misspellings

Non-words, then, can be identified by consulting electronic dictionaries. Although any given non-word will not be found, there might yet exist valid words that it resembles. Such valid words constitute possible corrections for any non-word in question. Rather than taking the first correction from any proffered list, such suggestions are better treated as competing candidates. The challenge lies in selecting the most appropriate candidate, which in turn demands an understanding of how errors arise (Kukich, 1993).

The Sources of Spelling Mistakes

In fact, spelling errors result from inserting, deleting, substituting, or transposing characters in any word. Such mistakes reflect either simple typographic errors, or so-called cognitive errors; the former are accidental mistakes by keyboard users, whereas the latter reflect misconceptions about any language used (Kukich, 1993). Spelling errors further stem from minor phonetic alterations (Crowell, Zeng, & Kogan, 2003). In fact, such phonetic alterations are examples of the cognitive errors mentioned by Kukich (1993).

Although Kukich (1993) sees substituted letters as a source of spelling errors, such transformations are commonly made deliberately. The pastime of word-ladder puzzles requires players to transform a given word into a different one, by replacing one letter at a time. The example from Adamchik (2009) presented next as Figure 3.1 shows how to change the word 'sail' into 'rail'; substituted letters are highlighted in bold type, and described opposite the 'rungs' between words in the ladder:

	sail	
Replace 'l' in 'rail' with 'r', giving 'rain'	**r**ail	Replace 's' in 'sail' with 'r', giving 'rail'
	rai**n**	Replace 'a' in 'rain' with 'u', giving 'ruin'
	r**u**in	

Figure 3.1: A word-ladder puzzle, adapted from Adamchik (2009)

.

Puzzles such as the one from Figure 3.1 must use words of identical length, which allows transformation just by replacing letters; should arbitrary-length words be permitted, letters would have to be inserted or deleted as well as substituted. Although solving word-ladder puzzles might seem trivial, it demonstrates the principle behind most approaches to correcting isolated misspellings. Specifically, the separation between any two words might be expressed in terms of the number of single-letter transformations required to change one into the other (Adamchik, 2009). That measure, called the Edit Distance (Navarro, 2001), will shortly be described in more detail. For now, suffice to say that machines can quantify the similarity of two text strings, in terms of the number operations required to transform one into the other.

Applying the Edit Distance to Suggested Spelling Corrections

According to Brill and Moore (2008), most spelling checkers use the Edit Distance as a way of comparing words. That was the approach taken by Kernighan, Church, and Gale (1990) in devising a program called `correct`, which processes words rejected by the `spell` program found on Unix® systems. In a preliminary phase, three human judges assessed so-called triples of words. The first word of any triple was a spelling mistake, while the remaining two words were candidate replacements from `correct`. All such triples comprised words having an Edit Distance of one, that is, the three words of any triple differed

from one another by just a single inserted, deleted or substituted letter (Kernighan et al., 1990).

In that study, judges recorded a percentage probability of any candidate correction being the best one. Candidates with the highest probabilities were chosen as replacements for misspellings in associated triples. That task proved more difficult than had been expected, with each judge taking about half a day to inspect the 564 triples used. Of those 564 triples, 235 were rejected because a majority of judges could not agree on the most likely correction. The remaining 329 triples formed the input to `correct` (Kernighan et al., 1990).

Having set benchmark probabilities, the next step required `correct` to calculate so-called conditional probabilities. That involved deriving a score for each candidate correction, in terms of two factors. The first factor, $Pr(c)$, was the probability of a candidate c existing in the 1988 AP corpus, expressed as:

$Pr(c) = (freq(c) + 0.5)/N$, where:

$freq(c)$ = the frequency of any candidate c in the AP corpus, and
$N = 44$ million, the number of words in that corpus

(Kernighan et al., 1990).

In that way, $Pr(c)$ reflected relative frequency in the AP corpus of suggested corrections, allowing `correct` to accept common suggestions in preference to uncommon, though valid, ones (Kernighan et al., 1990).

The second factor used by `correct` accounted for words transformed by insertions, deletions, substitutions and reversals. That process relied on a training dataset, held separately from the selected 329 test triples. Given a list of corrections judged appropriate by humans, `correct` was able to determine patterns of transformed letters, or of pairs of letters, in misspelled words. That gave rise to a measure expressed as $Pr(t|c)$: the probability of a letter t from the training set being transformed into c in the test set. For any misspelling, `correct` selected the candidate having the greatest product of factors $Pr(c)$ and $Pr(t|c)$. Of the 329 novel triples investigated, that approach gave 87% agreement with a majority of the human judges (Kernighan et al., 1990).

Notable Points about Using the Edit Distance

Several points of interest arise from the work of Kernighan et al. (1990). Most importantly, that study demonstrated the idea of transforming words by inserting, deleting or otherwise substituting letters when making typing mistakes. That said, the test phase addressed candidate corrections separated from any misspelling by just one such transformation, which surely underestimates the severity of errors that people make. Furthermore, the reliance on triples meant that two, and no more, candidates were considered for any misspelling. Again, that is unrealistic given the tendency that Kukich (1993) noted for shorter

misspellings to yield many candidates.

A further point concerns the required intervention by human judges. In contrast, the preferred approach to spell-checking GRiST mind maps would be fully automatic. Next comes the need to train the `correct` program, which is impractical given the relatively small sample of mind map nodes compared to the 44 million-word AP corpus. Simply, there are insufficient nodes in GRiST mind map to commit a good proportion to training. A final point is that the measure $Pr(t|c)$ was language-specific, being based on cross-referencing letters from the English alphabet. GRiST's medical specialists were indeed British; all the same, a linguistically neutral approach would facilitate work in other E.U. countries.

Analysing Sub-Strings of Words

Although that study by Kernighan et al. (1990) has various shortcomings, its application of $Pr(t|c)$ to transposed letters is of great interest; it raises the possibility of analysing letters and sub-strings of words, in addition to words themselves. Kukich (1993) refers to such sub-strings as n-grams, where 'n' is the number of letters contained. She split non-words into n-grams, each of which was researched in a pre-compiled table of n-gram statistics. That determined the likelihood of any particular n-gram occurring deliberately; the summation of probabilities from all of a non-word's n-grams indicated its likelihood of being a valid word (Kukich, 1993). In that way, candidate corrections for non-words were treated as likely or unlikely combinations of letters.

Work by Kukich (1993) demonstrated the use of n-grams to generate reliable corrections for non-words. Despite being absent from any dictionaries used, likely combinations of letters were treated as valid words. In a similar way, enhancing spelling correction by means of n-grams was one aspect of U.S. patent application 7366983. A process of partitioning split any non-word into segments of varying numbers of characters. Any proposed spelling correction was similarly partitioned, and the resulting segments compared with those from the associated non-word. A prior training exercise on valid corrections had determined probabilities of any particular segment being replaced by another (Brill & Moore, 2008). In similar way to Kernighan et al. (1990), novel arrangements of segments were deemed more or less likely in light of summing those probabilities.

The patent application of Brill and Moore (2008) is just one means of correcting spelling errors automatically. Having shown various approaches to identifying non-words, and to selecting appropriate corrections for isolated misspellings, attention now turns to the third challenge raised by Kukich (1993): that of accounting for context.

3.2.3 Accounting for the Context of Misspellings

According to Kukich (1993), existing approaches to automated spelling correction tend to ignore the difficult problem of context. All the same, Wilcox-O'Hearn, Hirst, and Budanitsky (1998) made a useful step in that direction by considering entire sentences, rather than correcting isolated misspellings. Specifically, sentences that contained just a single suspected spelling error were analysed. The non-word in any such sentence became the middle word of a trigram. The remaining two words of any trigram comprised words immediately before or after the corresponding non-word. Having generated a trigram for the erroneous part of any sentence, further trigrams were assembled from remaining words. Subsequently, researching an existing list of trigram frequencies gave a sequence of probabilities, which described patterns of trigrams in any sentence (Wilcox-O'Hearn et al., 1998).

The next step retrieved candidate corrections for spelling errors found in sentences. Following that, any original sentence was repeatedly rewritten; successive novel sentences used candidate corrections in place of the corresponding non-word. Such novel sentences were ranked by probability of component trigrams that depicted actual usage. In that way, a machine deemed sentences arising from a given suspect word to be more or less likely. Any preferred sentence had the highest overall probability of trigram usage in the associated group. In turn, such selected sentences bore preferred words, whether those were spelling suggestions or unchanged suspect words. That approach did, though, raise erroneous results that were considered inevitable from any statistical model. Indeed, a number of correctly identified spelling mistakes received inappropriate replacements (Wilcox-O'Hearn et al., 1998).

Treating Texts as Sources of Knowledge

While Kernighan et al. (1990) expressed a hope to account for context in future work, Wilcox-O'Hearn et al. (1998) made real steps in that direction. Both of those studies used trigrams to assist spelling correction; there was, though, an important difference in what those trigrams comprised. In the former study, trigrams held a spelling mistake and two candidate replacements. Human judges assessed those trigrams to yield statistics about trigram usage, which subsequently formed the input to the `correct` programme. The latter study, on the other hand, used trigrams comprising a non-word and any immediate neighbouring words. Once created, such trigrams were not treated differently to those from valid parts of sentences. Trigrams comprising parts of words, then, did not reflect the wider context of any non-word as did trigrams of whole words.

Rather than relying on just edit transformations *or* the context of misspellings, Mcnamee, Mayfield, and Piatko (2001) produced a fusion of those two approaches. That study gave rise to the Hopkins Automated Information Retriever for Combing Unstructured Text (HAIRCUT), which addressed misspellings in a test dataset of 1,588,374 words. Importantly, that dataset itself constituted a source of spelling

corrections. Words appearing in three or less of those documents were treated as spelling mistakes. In such cases, candidates comprised words that appeared in at least five of the test dataset's documents. Candidates further had to be separated by just one insertion, deletion or substitution from any original misspelling. The most common such suggestion was taken as the preferred correction. That combination of edit distance and context was said to help dramatically on two queries, but to be wrong for 'tartin'. That non-word was corrected to the more common 'martin', rather than to desired word 'tartan' (Mcnamee et al., 2001).

Treating Candidate Spelling Corrections as Edit-Neighbours

Having earlier criticised the work of Kernighan et al. (1990) for underestimating the severity of spelling errors, a similar issue arises from the study by Mcnamee et al. (2001). In both cases, experiments addressed candidate corrections separated by just one edit operation from any misspelling, a matter that recurred in U.S. patent application 7366983. In a combination of edit distance and context, that patent introduced the idea of edit-neighbours: candidate corrections separated from an associated non-word by a single inserted, substituted or deleted letter. By that definition, edit-neighbours were equally likely replacements for any suspected spelling mistake. A most likely correction arose from inspecting words that preceded and followed any suspect word. That gave the probability $P(w|context)$ of any word w appearing with particular preceding and following words. The candidate having the highest value of $P(w|context)$ constitutes the best correction (Brill & Moore, 2008).

Accounting for the Meanings of Words

Generally timid approaches, then, handled spelling corrections that closely resembled any offending word. Context, though, was important in assessing any such corrections. In a broader sense still, context has extended to cover the meanings of words. In that respect, Kukich (1993) assessed approaches that used Natural Language Processing (NLP) to determine appropriate spelling corrections. Instead of any textual edit distance, semantic distance measures separation between words' meanings. Although effective, such NLP approaches as seen as over-complex (Kukich, 1993). Indeed, results from purely textual analysis are noticeably better than from such semantic approaches (Wilcox-O'Hearn et al., 1998).

3.3 Improving Spelling Correction for GRiST Mind Maps

Suggested spelling corrections, then, might be rejected should they differ greatly from perceived misspellings; instead, offending words will be retained. The approach here is to assume that spelling errors will roughly resemble any intended words. Suggestions close to any word that they mean to replace will be accepted, while those differing markedly will be rejected. Indeed, such offending words will be retained in an extensible extra dictionary. In particular, suggestions much longer or shorter than any misspelled word are less likely to be acceptable.

Further, misspelt words and proposed corrections alike will be researched in GRiST mind maps, which themselves constitute an extra dictionary. Suggestions might receive support should they exist in those mind maps. Conversely, words wrongly identified as spelling errors will be accepted should several authors use them; such words were likely typed deliberately. Should no alternate authors use any given word, the relative similarities of perceived errors and suggested corrections are considered. As Idzelis (2005) and Brill and Moore (2008) point out, assessing similarities between words commonly involves the Edit Distance, which reflects the numbers of letters that might be rearranged to yield any spelling mistake.

In fact, the Levenshtein Edit Distance is used here; the standard version, though, reflects just a total number of transformations between any two words, regardless of where such changes occur. That shortcoming might be overcome by modifying the algorithm, as approaches to automating spelling correction will show. Following that comes the adjustment applied by this thesis, which is assessed experimentally.

3.3.1 The Levenshtein Distance, L

In fact, textual analysis was not the original application for the edit distance. Rather, Levenshtein (1966) devised it to automatically correct transmission errors between early computers. Words in that context were strings of binary digits, or bits, from a binary alphabet described as $\{0, 1\}$. Unfortunately, hardware channels used to transmit sequences of binary words were prone to corruption. Words at the receiving end sometimes differed in various ways to what was sent; put another way, such channels were noisy. The Levenshtein Distance, L, helped to alleviate that problem of noisy channels (Levenshtein, 1966) [1].

The types of errors inherent in noisy channels have been described already, while discussing the challenge of spelling correction. Transmission errors involving deletion arose when a so-called empty word replaced valid bits. Using the symbol $\wedge$ to represent any such empty word, deletions were expressed as $0 \rightarrow \wedge$ and $1 \rightarrow \wedge$. Conversely, insertions occurred when a valid bit replaced an empty word: $\wedge \rightarrow 0$ or $\wedge \rightarrow 1$. A third type of error, reversal, arose when $1 \rightarrow 0$ or $0 \rightarrow 1$. To that end, L identified such transmission

[1] The terms Levenshtein Distance and Edit Distance are, in fact, synonymous (Navarro, 2001).

errors, and further attempted to determine what was actually sent (Levenshtein, 1966).

In practice, rectifying transmission errors involved what were termed *binary codes*. Such codes comprised valid combinations of characters from the original alphabet $\{0, 1\}$, and the empty word $\wedge$. By specifying allowable words, binary codes identified any bad segments emerging from a channel. Any word with, say, a deleted bit would fail to find a corresponding code. In such cases, the closest matching code was taken as the intended content, the word 'closest' being expressed in terms of two binary words, x and y, in the function $r(x, y)$. That function reflected the smallest number of insertions, deletions and reversals that transformed a transmitted segment x into received segment y. Bits deemed to have been, say, inserted into y were deleted from x to reveal the most likely intended binary word (Levenshtein, 1966).

Applying the Levenshtein Distance to Further Alphabets

That consideration of insertions, deletions and reversals led to the metric L: a quantitative measure of any difference between strings from a particular alphabet. Although L was conceived for handling just the characters 1 and 0, strings from larger alphabets might be compared in the same way. Indeed, codes could be constructed for any alphabet of r letters, where $r \geqslant 2$ (Levenshtein, 1966). The importance of L, then, transcends any ability to recreate corrupted binary information, and has become an important way of comparing words constructed from any alphabet. Indeed, studies that were reviewed in Section 3.2 constituted approaches based on a noisy channel model (Kernighan et al., 1990; Wilcox-O'Hearn et al., 1998; Brill & Moore, 2008).

Moving from a binary alphabet to English does, though, require one small change in terminology. While retaining the notions of insertion and deletion, it is better to refer to 'replacements' rather than to 'reversals'. That is because reversals cannot occur in words from larger alphabets, as letters have no specific inverse. Any specific character, though, might be replaced by another. In that respect, transposition is a special type of substitution that resembles reversal, in that two characters change places between any given pair of words. Regardless of that difference in terminology, small values of L between two strings from any alphabet mean that such words are close variants of one another (Levenshtein, 1966; Navarro, 2001). Having introduced L, the algorithm will now be discussed in more detail.

Calculating the Levenshtein Distance

Calculating L involves a process called dynamic programming, for which Navarro (2001) gives the algorithm reproduced below as Figure 3.2. That algorithm computes L between a source string x of length $|x|$ and a target string y of length $|y|$, by means of a matrix expressed as $C0..|x|, 0..|y|$. The number of rows in that matrix depends on the length $|x|$ of the source, while the number of columns depends on the length $|y|$ of the target. In that way, rows in such matrices correspond to letters from any source word, and columns to those from any corresponding target word. Here, then, is how L arises from words x and

y, which are respectively indexed by variables i and j:

$C_{i,0} = i$ $C_{0,j} = j$	Set top-row cells to i. Set left-hand column cells to j.
$C_{i,j} = if(x_i = y_j)\ then$ $\quad Ci-1, j-1$ $else$ $\quad 1 + min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})$	Letter i of word x = letter j of word y: current cell = cell above & to the left: $\nwarrow$. Letter i of word x ! = letter j of word y: find the neighbouring $\uparrow$, $\leftarrow$, or $\nwarrow$ cell having the lowest value.

Figure 3.2: Dynamic programming for L, adapted from Navarro (2001).

The first two entries in Figure 3.2 prime the matrix by setting the first row to the series $0..|x|$, and the first column to $0..|y|$. That accounts for either of words x or y being an empty string; a blank target string, say, means that all characters in x must be deleted to make y. In that case, the minimum number of deletions is the number of characters in x, that is, the length of the source string (Navarro, 2001).

The remainder of Figure 3.2 depicts an iterative process that assigns successive cells in the matrix. Any given cell receives a value that depends on neighbouring cells above, to the left, and diagonally above and to the left. Because that involves cells at positions $i-1$ and $j-1$, the algorithm starts those subscripts at 1, rather than at zero. Note further that such upper, left, and upper-left neighbours must be computed before any given cell can be assigned. In general, that is accomplished by traversing the matrix one row at a time, from left to right (Navarro, 2001). In fact, an identical value of L would arise regardless of which order strings processed. Because the algorithm is symmetrical, matrices might be traversed in any direction without affecting L. In addition, insertions, say, in any given source string correspond to deletions from any target word (Ackroyd, 1980; Navarro, 2001).

Calculating any specific cell depends on letters at respective positions i in the source word x and j in the target, y. Should those two characters be identical, the algorithm in Figure 3.2 assigns the value from the upper-left neighbour of cell $C_{i,j}$. That reflects the lack of any need to transform letter x_i into y_j. Conversely, when those two letters differ, the algorithm assigns $C_{i,j}$ by adding 1 to the smallest value from neighbouring cells. That value reflects a minimum cost for transforming letter x_i into y_j. A final value for L appears in the bottom-right cell, $C_{i,j}$, which reflects the minimum cost, in terms of the number of operations, of turning string x into y (Navarro, 2001).

Steps in Calculating L

The matrix that follows as Figure 3.2 shows the first steps in comparing two common words from GRiST mind maps: 'abused' and 'abusive'. Letters from those two words are shown here just for information, in the outer row and column; although the algorithm considers individual letters, they not required in the

matrix. The first mandatory row comprises initial values of $0..|x|$ for the source string 'abused', while the first column is set to $0..|y|$ for the target string 'abusive'. Those initial settings, shown in italics, do not depend on neighbouring cells; rather, they are assigned directly.

Conversely, values for individual cells of the second row, shown shaded, arise from comparing the first letter of x with all those in y, that is, cells $C_{1,1..|y|}$. It must be stressed that those cells are assigned as the algorithm progresses, even though they are presented as if completed. Beneath that nascent matrix come four rows of computations for individual cells. The first three of those rows give the edit operations that were considered, with the selected one highlighted. The last row assigns any derived values to the corresponding shaded cells, $C_{1,1..|y|}$:

		a	b	u	s	i	v	e	
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	
a	*1*	0	1	2	3	4	5	6	$C_{1,\,1..\lvert y\rvert}$

	$x_i = y_j$	$\uparrow 1+2$	$\uparrow 1+3$	$\uparrow 1+4$	$\uparrow 1+5$	$\uparrow 1+6$	$\uparrow 1+7$
min of:	= 'a'	$\leftarrow 1+0$	$\leftarrow 1+1$	$\leftarrow 1+2$	$\leftarrow 1+3$	$\leftarrow 1+4$	$\leftarrow 1+5$
	$\nwarrow = 0$	$\nwarrow 1+1$	$\nwarrow 1+2$	$\nwarrow 1+3$	$\nwarrow 1+4$	$\nwarrow 1+5$	$\nwarrow 1+6$
$C_{i,j}$	$C_{1,1} = 0$	$C_{1,2} = 1$	$C_{1,3} = 2$	$C_{1,4} = 3$	$C_{1,5} = 4$	$C_{1,6} = 5$	$C_{1,7} = 6$

$i = 1,$
$j = 1..|y|$

Figure 3.3: Steps for $i = 1$, $j = 1..|y|$ in calculating L

Steps from Figure 3.3 compare the first letter, 'a', of the source word with all of those from the target word. Using $i = 1$ and $j = 1$ in the first step means that characters x_i and y_j are identically 'a'. In that special case, the algorithm ignores the left and upper neighbours, and selects the upper-left cell. The value in that cell gives a cost of 0 for transforming 'a' into 'a', that is, nothing. Because this is the first row, that cost comes from an initialised value rather than from one derived by the algorithm.

Subsequent steps in Figure 3.3 deal with differing source and target letters. When, for example, $i = 1$ and $j = 2$, those letters x_i and y_j are 'a' and 'b' respectively. In such cases, neighbouring cells reflect the lowest cost of changing x_i into y_j. In that way, cell $C_{1,2}$ has upper neighbour $C_{0,2}$, which holds the value 2; adding 1 gives a total cost of 3. The value 1 in upper-left neighbour $C_{0,1}$ in turn gives a total cost of $1 + 1 = 2$. The lowest cost, though, comes from the value 0 in left-hand neighbour $C_{1,1}$. The resulting total of $1 + 0 = 1$, then, is the lowest cost of transforming 'a' into 'b'. Coming from a left-hand neighbour, that lowest cost reflects that 'a' was substituted by 'b' in the target word.

The Dynamic Nature of Calculating L

As Navarro (2001) points out, the computation of L is called dynamic because any matrix grows as the algorithm progresses. In that way, cells having $i = 1$ in Figure 3.3 constitute upper neighbours of cells in the subsequent row, where $i = 2$. Rather than the initial values used when $i = 1$, that second iteration depends on values from the preceding step; that is shown in Figure 3.4, which compares the

source word's second letter, 'b', against all those in the target word 'abusive'. Cells on row $i = 2$ are derived in succession as the algorithm increments j. When $i = 2$ and $j = 2$, the letters x_i and y_j are both 'b'; in that case, the second cell of calculations assigns $C_{2,2}$ directly from the upper-left neighbour:

		a	b	u	s	i	v	e	
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	
a	*1*	0	1	2	3	4	5	6	
b	*2*	1	0	1	2	3	4	5	$C_{2,\ j..\lvert x\rvert}$
		$\uparrow 1+0$	$x_i = y_j$	$\uparrow 1+2$	$\uparrow 1+3$	$\uparrow 1+4$	$\uparrow 1+5$	$\uparrow 1+6$	
min		$\leftarrow 1+2$	= 'b'	$\leftarrow 1+0$	$\leftarrow 1+1$	$\leftarrow 1+2$	$\leftarrow 1+3$	$\leftarrow 1+4$	$i = 2,$
of:		$\nwarrow 1+1$	$\nwarrow = 0$	$\nwarrow 1+1$	$\nwarrow 1+2$	$\nwarrow 1+3$	$\nwarrow 1+4$	$\nwarrow 1+5$	$j = 1..\lvert y\rvert$
$C_{i,j}$		$C_{2,1} = 1$	$C_{2,2} = 0$	$C_{2,3} = 1$	$C_{2,4} = 2$	$C_{2,5} = 3$	$C_{2,6} = 4$	$C_{2,7} = 5$	

Figure 3.4: Steps for $i = 2$, $j = 1..|y|$ in calculating L

Steps from Figure 3.4, then, compare the 'b' in 'abused' with successive letters of the word 'abusive'. Because cell $C_{2,2}$ was set just by reference to upper-left neighbour $C_{1,1}$, cell $C_{2,2}$ inherits a value of 0 that was set in a similar way for comparing 'a' with 'a' to give cell $C_{1,1}$. In other words, the cost of 0 for changing 'b' into 'b' reflects a preceding zero-cost transformation.

Steps for $i = 1$ from Figure 3.3 selected a uniform edit operation when source and target letters differed. The lowest cost in all such cases came from left-hand neighbours. That is not so when $i = 2$; Figure 3.4 obtained a lowest cost from upper neighbour $C_{1,1}$ when $j = 1$, rather than from left-hand neighbours when $j = 3..|y|$. The cost entered into cell $C_{1,1}$ arose, in fact, from comparing 'a' in 'abuse' with 'b' in 'abusive' during the preceding step for $i = 1$. In the current step, where $i = 2$, the comparison is between 'b' in 'abuse' and 'a' in 'abusive'. In that way, the cost of changing 'b' into 'a' is the inherited cost of transforming 'a' into 'b'. Repeated steps, then, give a matrix of costs; such a matrix is presented next.

The Matrix Arising from Calculating L

Importantly, inverse transformations such as 'a' to 'b' and 'b' to 'a' cancel one another, emphasising the symmetry of matrices for L noted by Ackroyd (1980) and by Navarro (2001). Repeating the processes described for the first two rows until $i = |x|$ and $j = |y|$ gives the finished matrix in Figure 3.5, where a final value of $L = 3$ appears in $C_{6,7}$, the bottom-right cell:

		a	b	u	s	i	v	e
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
a	*1*	0	1	2	3	4	5	6
b	*2*	1	0	1	2	3	4	5
u	*3*	2	1	0	1	2	3	4
s	*4*	3	2	1	0	1	2	3
e	*5*	4	3	2	1	1	2	2
d	*6*	5	4	3	2	2	2	3

Figure 3.5: Steps for $i = 1..|x|$, $j = 1..|y|$ in calculating L

The value of $L = 3$ Figure 3.5, then, is the minimum number of insertions, deletions or substitutions needed to transform 'abused' into 'abusive'.

In fact, no asymmetrical insertion, deletion or substitution occurs in Figure 3.5 until both i and j reach 5. Whereas preceding cases of $i = j$ were cost-free because letters x_i and y_j were identical, the value for cell $C_{5,5}$ arose from comparing 'e' in 'abuse' with 'i' in 'abusive'. Rather than defaulting to the upper-left neighbour, those different letters forced the algorithm to determine the lowest cost from three competing neighbours. Now, the dynamic programming algorithm from Figure 3.2 added the constant 1 in such cases, rather than simply taking any neighbouring cell's value. Just as for comparisons between identical letters, an upper-left neighbour yielded the lowest cost. In this case, though, further adding 1 to the 0 in $C_{4,4}$ gave an overall cost of 1.

That an upper-left neighbour in Figure 3.2 gave the lowest cost of transforming 'e' into 'i' meant that those letters were substituted. In a similar way, cell $C_{6,6}$ arose from comparing 'd' in 'abused' with 'v' in 'abusive'. In that case, the lowest cost of 1 came from cell $C_{5,5}$; adding 1 gave a total cost of 2. Cell $C_{5,5}$, in turn, reflected the prior cost of 1 incurred by substituting 'e' with 'i'.

The final value of L in cell $C_{6,7}$ arises from comparing 'd' with 'e'. All of the three neighbours consulted bore a value of 2, giving a cost of $2 + 1 = 3$. That would best be interpreted as appending the letter 'e' with an insertion. That, as Navarro (2001) points out, is unimportant should just a value of L be required[1]. The nature of such changes are more obvious to humans; L emulates that ability, allowing

[1]Indeed, the implementation by Gilleland (2003) introduced in Part III defaults to 'deletion'.

machines to determine what transformations separate any two words. That, in turn, reflects the degree of similarity between words.

A Drawback of the Standard Version of L

In light of comparing 'abused' and 'abusive', it is tempting to use L directly as a criterion of similarity. Smaller values of L would indicate relatively little difference between character strings, be they candidate spelling corrections or words related by a common stem. An important problem remains, though: any specific value of L indicates more overall variation between short words than between longer ones. It follows that any algorithm for L might account for the lengths of strings under comparison. All the same, various studies have found the standard algorithm for L sufficient. Attention now turns to such approaches, before examining adjusted forms of the Edit Distance.

3.3.2 Using and Refining the Levenshtein Distance

The Edit Distance, then, provides a means of filtering candidate suggestions for spelling mistakes. Indeed, Brill and Moore (2008) applied L to that task when creating a patented spelling checker. In addition, L has been successfully applied to a process known, if clumsily, as lemmatisation, by which Lyras, Sgarbas, and Fakotakis (2007) identified so-called lemmas in Greek texts. Whereas Brill and Moore (2008) modified the algorithm for L, Lyras et al. (2007) used the standard version to good effect.

Applying the Standard Version of L

Selecting L as tool reflected that algorithm's successful application in areas such as genomics, speech and handwriting recognition, spell-checkers, and Internet search engines. In genomics, for example, strings of interest comprise the initials of the four base pairs that make up DNA, from the alphabet $\{A, T, C, G\}$. In contrast, approaches to spelling correction discussed in Section 3.2 involved alphabets used by humans. Whatever the application or alphabet, L measures the relative similarity of strings. For the process of lemmatisation, L offered a means of determining likely lemmas, from which users might choose (Lyras et al., 2007).

A lemma for any given word arose as follows. First of all, such words were researched in a database of 30,000 Greek lemmas, compiled in advance. Searches that yielded several candidate lemmas were refined by comparing suggestions against whatever word was entered. That involved computing the standard L separating any input word from each associated lemma. Having done that, lemmas having the minimum L within the returned list were chosen. That, though, was the default selection criterion. Should users so wish, lemmas of a specified additional distance were returned for consideration. Entering, say, 2 as the desired approximation returned lemmas having a distance of $L \leqslant (min + 2)$ from the source word entered, where min was the minimum L for the group concerned (Lyras et al., 2007).

Accounting for String Lengths when Calculating L

Whatever alphabet is used, then, higher values of L reflect greater variation between any two strings. Indeed, L might better be described as a measure of *dis*-similarity. The impact of any edit operations, though, depends on the lengths of strings being compared; even a single transformation between short strings is more noteworthy than it would be between longer ones. Overcoming that problem demands that raw values of L be adjusted to account for words' lengths. Such a relative edit distance would better reflect the extent to which edit operations affect any given word (Paleo, 2007).

A review by Navarro (2001) reveals considerable research into refining and extending the Edit Distance algorithm. Because using L to compare large texts involves numerous calculations, improving computational efficiency is a common aim. That is not the motivation here, though. Rather, it is that the basic form of L is, by itself, too coarse a measure. Indeed, Brill and Moore (2008) point out most spelling checkers apply some form of weighting to L when comparing words. Fortunately, Navarro (2001) sees L as easily adapted. Attention turns next, then, to studies that have made such adaptations.

Adaptations to the Levenshtein Distance

The study of Greek text by Lyras et al. (2007) used L to support decisions taken by humans. Although highlighting the benefit of applying L to stemming, human intervention and a dependence on existing stems limited the utility of that approach. In contrast, Ackroyd (1980) made an adjustment to L in search of a more automated process. That study modelled the transmission of speech signals by using simulated telephone lines, aiming to aid machines in identifying spoken words. To that end, a process of character-string encoding transformed utterances into what were called symbol-strings. The result was a dataset of five symbol-strings for each of ten spoken words (Ackroyd, 1980).

That dataset, though, showed wide variation in the length of symbol-strings, due to how quickly people spoke. A Weighted Levenshtein Distance (WLD) solved that problem by disregarding minor variations in word duration, while remaining sensitive to large differences. The starting point for the WLD was the standard dynamic programming algorithm, which was expressed in a slightly different way than was presented for Navarro (2001) in Figure 3.2. In that respect, Figure 3.6 shows a cost of $d(i, j)$ applied to any insertion, deletion or substitution at respective positions i and j in the source and target strings:

$$g(i,j) = min \begin{bmatrix} (g(i-1,j) & + & d(i,j) \\ (g(i-1,j-1) & + & d(i,j) \\ (g(i,j-1) & + & d(i,j) \end{bmatrix}$$

insertions: comparison is $\uparrow$
substitutions: comparison is $\nwarrow$
deletions: comparison is $\leftarrow$

Figure 3.6: An alternative form of L, from Ackroyd (1980).

Figure 3.6 shows the familiar operations of insertion, substitution and deletion noted by Navarro (2001), which correspond respectively to upper, upper-left and left neighbouring cells. Whereas that version of the algorithm added a standard cost of 1 to values from neighbouring cells, the alternative version in Figure 3.6 shows that just $d(i, j)$ is considered. For two given strings, though, identical values of L arise from both versions. A cost of 1 for all operations in the alternative achieves the same end (Ackroyd, 1980).

Standard L was adapted by means of a weighting d_{ID} that applied just to insertions and deletions. The cost of substitutions, $d(i, j)$, is retained from the standard version, as shown in Figure 3.7:

$$g(i,j) = min \begin{bmatrix} (g(i-1,j) & + & \mathbf{d_{ID}} \\ (g(i-1,j-1) & + & d(i,j) \\ (g(i,j-1) & + & \mathbf{d_{ID}} \end{bmatrix}$$

insertions: weighted by constant d_{ID}

deletions: weighted by constant d_{ID}

Figure 3.7: A weighted form of L, from Ackroyd (1980).

The weighting d_{ID} in Figure 3.7 carried a value of 2, affecting insertions and deletions equally. WLD for transforming any part of symbol-string A into B, was the same as for turning B into A (Ackroyd, 1980).

More importantly than the weighting d_{ID} itself was the adjustment made to the WLD for symbol-strings of differing lengths. For any two symbol-strings of respective lengths I and J, that involved first multiplying the difference in those lengths by the weighting d_{ID}. The resulting value was subtracted from the standard version of L, as the following expression shows:

$$weightedWLD = WLD - |I - J|d_{ID} \qquad \text{(Ackroyd, 1980).}$$

That adjustment allowed the WLD to compensate for the lengths of symbol-strings, eliminating any penalty on words of differing lengths. Subtracting a function of string lengths from L reduced the impact of values arising from widely differing lengths. The larger any such difference became, the lower the resulting L. That is not to say that large edit distances were ignored completely; rather, L arising from differences in string lengths were less influential than those due to substituted symbols. In that way, words that closely resembled one another apart from the speed of speech were treated as identical. Experiments showed that spoken English digits were correctly recognised in about 95% cases. For words of more than one syllable, that rose to nearly 99%, which was deemed as good as might be hoped (Ackroyd, 1980).

Applying Weighted Forms of L to Analysing Text

Work by Ackroyd (1980) well demonstrated the utility of adjusting L for the relative lengths of participating strings. The symbol-strings examined, though, comprised electronic representations of utterances

rather of printable English words. Attention now, then, turns to ways that weighted forms of L assist in analysing text. One such example comes from a study by Paleo (2007) of so-called gazetteers, software programmes that search for words in texts. On detecting words that exist in a pre-prepared list, gazetteers highlight matching regions in any text under analysis. For example, regions that refer to cities are annotated with the locations of those cities (Paleo, 2007).

Absolute edit distances, though, were not seen as useful, a separation of one character being more noteworthy in short words than in longer ones. Accordingly, an Approximate Gazetteer dealt with relative distances, derived as follows. The gazetteer first computed L between a given word from the supplied list and any region of text. Subsequently, the length of such regions was multiplied by a relative threshold of between 0.0 and 1.0, giving an absolute threshold against which to compare values of L. Text regions that yielded values of L below any associated absolute threshold were deemed to be worthy of annotation by the gazetteer. In that study, the Levenshtein algorithm was seen as a good compromise between flexibility in handling edit operations, and of computational efficiency. (Paleo, 2007).

Amending the Underlying Algorithm of L

Whereas Paleo (2007) adjusted values of L after they had been calculated, Higuera and Micó (2008) altered the underlying algorithm. The resulting contextual edit distance accounted for word-lengths in a Spanish dictionary of over 80,000 words. In fact, differing flavours of L stressed specific aspects of any comparison. Figure 3.8 shows the first of those distances, d_{sum}, between strings x and y of respective lengths $|x|$ and $|y|$. The component $d_E(x, y)$ represents the standard edit distance between those strings:

Figur 

Micó (2008).

Although Higuera and Micó (2008) offered little further explanation of the adjusted distance from Figure 3.8, it is important here due to considering the lengths of strings x and y, $|x|$ and $|y|$ respectively. Dividing the standard edit distance d_E by the summed lengths $|x| + |y|$ would reduce L when comparing longer words. In that respect, standard L would not differentiate between comparing a short word and a very long one, and comparing two medium-sized words having the same summed length.

In contrast, two further distances d_{min} and d_{max} respectively divide d_E by the length of the shorter word, $min(|x|, |y|)$, or of the longer word, $max(|x|, |y|)$. Whatever particular measure is applied, though, the general rule is that the edit distance $d(x, y)$ is inversely proportional to any word lengths involved, that is, $d \propto \frac{1}{l}$. In that expression, l might constitute summed lengths, or just the shorter or the longer

one. In all of those cases, though, values of d tend to diminish as l grows. Any given value of L, then, would indicate a greater distance between short strings than between longer ones.

3.3.3 A Proposed L' that Accounts for Word Length

Referring to any refined version of L as L', the common rule is that $L' \propto \frac{L}{d}$. A given number of deletions, insertions or replacements between longer strings would produce a smaller value of L' than would shorter words. The approach taken here relies just on the simple version of L from Figure 3.2 that uses costs of 0 and 1. Word lengths, though, will be considered only after first calculating simple L between two strings. Exactly what that length adjustment entails is addressed next.

The term d was introduced to express various combinations of two words' lengths. Possibilities included the sum of lengths, the longer length, the shorter one, and the ratio of the two. Ackroyd (1980) provided the sole example of using absolute differences in lengths. Even so, that was but part of an expression involving the additional weighting d_{ID}. In fact, that absolute difference in lengths constitutes the adjustment applied by this thesis.

In illustration, the example of 'abuse' versus 'abusive' involved both replacements and insertions. All the same, the value of $L = 3$ suggests that removing the last three letters would leave the common sub-string 'abus'. Unfortunately, words that are not related often yield the same result of $L = 3$, for example, 'articulate' compared with 'particular'. Clearly, a more discerning measure must be sought Rather than dividing L by some combination of word lengths, the L' algorithm employed here involves subtraction. Further, that adjustment will be applied after calculating simple L, in contrast with studies reviewed in Section 3.3.2 that adjusted the underlying algorithm itself. The adjustment employed here subtracts any difference in word lengths, d, from L. For two words of respective lengths i and j, then, L' is calculated as follows:

$$\begin{aligned} &\text{absolute difference in word lengths:} && d = |i - j| \\ &\text{adjusted } L\text{:} && L' = L - d. \end{aligned}$$

Although that might seem over-simple in comparison to adjustments made by other approaches, the efficacy of L' is proved by experiments in improving spelling correction that now follow.

3.4 Experiments in Improving Spelling Correction

The proposed means of checking spelling corrections were assessed by implementing the pseudo code from Figures 3.9 and 3.10. The resulting Java class `MindmapSpellChecker` was run against the collection of GRiST mind maps to assess the overall effectiveness of those measures. Specific attention was paid to values of $maxL$ that led to accepting appropriate corrections, while rejecting others.

Method

The first step was to compile a list of unique words from GRiST mind maps. To that end, all nodes were retrieved from the database that held those mind maps[1]. Text from each node was transformed to lower case, then split into separate words by the Java method `split()` from the `String` class. Characters other that alphanumeric ones served as delimiters, and were discarded during that process. For a particular node, that yielded one or more words comprised of just the letters $a - z$ and the numerals $0 - 9$. Words containing numerals were subsequently discarded, and did not participate in the analysis. The following command, then, splits text in a Java String object called `nodeText` into an array of single words:

```
String[] words = nodeText.toLowerCase().split('\\W');
```

Words resulting from splitting nodes were loaded into a Java `TreeSet` object. By storing any particular value just once, such objects omit duplicated entries. The ensuing list served as an extra dictionary for checking spelling corrections, although the mind map containing any non-word was not consulted. Suspected errors absent from that list of GRiST words were passed to the Jazzy spelling checker from Idzelis (2005). Jazzy provides several dictionaries as separate files, of which just two were used: the general and the U.K. dictionaries; words from the U.S. dictionary did not contribute spelling corrections.

Candidate corrections from Jazzy were further processed by a bespoke Java object, `MindmapSpellChecker`, which calculated L and L' between errors and corresponding suggestions. Although matrices introduced in Section 3.3.2 revealed shared sub-string between words, that facility will not be used to validate spelling corrections. Appropriate corrections might be rejected needlessly if, say, a letter must be inserted in order to correct a misspelling, splitting any shared sub-string to either side of that insertion. The very nature of spelling correction suggests makes that likely. For that reason, just the Edit Distance from such matrices will be used to refine suggested corrections.

[1] Chapter 9 deals in full with retrieving information from GRiST mind maps.

Proposed refinements for spelling corrections are summarised next as pseudo code in Figure 3.9, which shows how a given word is researched, and any suggestions evaluated:

Pseudo Code for L' spelling correction	Comments
get corrections for word	Retrieve suggestions for the current word
for each correction until one accepted	Find the first acceptable suggestion
if correction in GRiST	Correction is a word from GRiST mind maps
accept correction	Accept proposed correction for the current word
else	
if `correctionOk()`	Check acceptability with L'
accept correction	Accept proposed correction for the current word
end-if	
end-if	
end-for	

Figure 3.9: Pseudo Code for checking spelling corrections by means of L'.

An empty list from the first step from Figure 3.9 indicates a valid word; otherwise, suggested corrections are checked in GRiST mind maps, which in that way constitute a dictionary. The loop that processes spelling corrections will be entered, then, should any suggestions arise. Words that receive no support from those mind maps are checked further by the function `correctionOk()`[1]. Additional pseudo code for that function appears next as Figure 3.10, which applies L' to proposed corrections:

Pseudo Code for `correctionOk()`	Comments
get L between word & correction	Compute L and $L' = L - d$ between word & suggestion
if $L' = 0$	Length differences account for variation
accept correction	Accept proposed correction for the current word
else	
if ($d = 0$ AND $L < maxL$)	Equal lengths, and reasonably similar words
accept correction	Accept proposed correction for the current word
end-if	
end-if	

Figure 3.10: Pseudo code for the `correctionOk()` function called from Figure 3.9.

Pseudo code from Figure 3.10 first checks for corrections having $L' = 0$ in comparison with spelling mistakes; such suggestions are accepted. The second check, applied when $L' > 0$, is performed on words that yield corrections having the same length. In such instances, L' is checked against a constant given as $maxL$. Suggestions are rejected should L exceed that constant, indicating excessive variation.

[1]Please note the font used to depict functions such as `correctionOk()`, and subsequently, all Java classes.

The first candidate separated by $L' = 0$ from any offending word was accepted automatically. Otherwise, simple L assessed corrections that were the same length as the word that produced them. Values from the inclusive range 0..2 were applied as the constant $maxL$, which defined successively more lenient criteria for accepting corrections. Suggestions were dismissed should comparisons with offending words yield $L > 2$. Such unrecognised words were subsequently researched in the list from GRiST, in search of smaller words that yielded $L' = 0$. That, in turn, would reveal valid words embedded in misspellings.

Non-words arising from conjoined words were split into separate, valid words. That was done by a combination of L' and researching words from GRiST mind maps. Non-words were compared against each word from the list compiled from those mind maps. Cases having $L' = 0$ were added to a `TreeSet` from an array of such objects; each element of that array represented a particular non-word. On completion, `MindmapSpellChecker` examined the resulting list of `TreeSet`s. Should any such object contain more than just a single word, the lengths of those words were summed. Non-words having lengths equal to the summed length from the associated `TreeSet` were split into those corresponding words.

Results

GRiST mind maps contained 4447 unique words, of which 4253 were wholly alphabetic. Jazzy failed to recognise 374 of those 4253 words, and offered suggestions for 337. Of those suggestions, 212 appeared in GRiST mind maps, and were accepted as novel words. Of the remaining 125 unrecognised words assessed by L', 59 yielded corrections that gave $L' = 0$. A further 30 corrections of identical length to the word producing them were accepted by means of the standard Edit Distance, where $L = 1$; an additional 11 corrections were accepted by $L = 2$. Suggestions for the remaining 37 words were rejected as having L greater than the ceiling of $maxL = 2$. A summary of results presented so far follows as Table 3.1:

Note	Test Performed on Non-Word vs. Jazzy Suggestion	Counts
Rejected words	No suggestions from Jazzy	37
Accepted suggestions	Found as a word in GRiST mind maps	212
	$L' = 0$	59
	$d = 0$ and $L = 1$	30
	$d = 0$ and $L = 2$	11
Rejected suggestions	All suggestions failed the above tests	25
Totals		374

Table 3.1: Summary of Spelling Correction Results.

Details of results from Table 3.1 are presented next in four parts. First of all come corrections that were accepted solely by consulting Jazzy, to show how it performed unaided. After those come corrections that found support from GRiST mind maps. The third set of results shows suggestions that were accepted

due to checking L and L'. After that come results from Jazzy alone that improved on consulting GRiST mind maps, or on checking L and L'. For each of those first three sets of results, acceptable corrections are shown first. Separate tables of inappropriate corrections, where needed, appear immediately after any suitable ones. The fourth main set shows results that corrected conjoined words by splitting them into separate, valid words. Following the main four sets of results from spelling correction, a supplementary set highlights corrections that will subsequently affect stemming.

Results I for Jazzy corrections Alone

Taking the first suggestion from Jazzy for any dubious word sometimes yielded suitable corrections. A few examples are presented next as Table 3.2:

Non-Word	Suggestion	Non-Word	Suggestion	Non-Word	Suggestion
aquire	acquire	flucky	fluky	survelience	surveillance
begining	beginning	likliehood	likelihood	tattooes	tattoos
colabarating	collaborating	openess	openness	welbeing	wellbeing

Table 3.2: Appropriate spelling corrections from Jazzy alone.

In contrast to corrections from Table 3.2, Jazzy accepted various candidate corrections that, to a human, would be blatantly misleading. Examples of such inappropriate suggestions follow as Table 3.3:

Non-Word	Suggestion	Non-Word	Suggestion	Non-Word	Suggestion
agrophoia	agraphia	keyworker	coworker	sytem	stem
benzoes	bonzes	parasuicidal	presystole	untreatable	intratubal
clorazil	gloriously	premorbid	pyromorphite	whinging	winging

Table 3.3: Inappropriate spelling corrections from Jazzy alone.

Discussion I of Jazzy corrections Alone

The first set of results, then, determined the efficacy of simply consulting Jazzy. As Table 3.2 showed, appropriate suggestions arose for misspellings of various lengths. Those cases needed no further analysis, any first suggestion from Jazzy proving adequate. Conversely, Table 3.3 showed various unacceptable suggestions from Jazzy. Indeed, certain of those words were not mistakes at all, but valid words missing from Jazzy's dictionaries. That echoes the distinction made by Kukich (1993) between non-words which constitute true errors, and those that, while unrecognised, are actually valid. Examples included 'para-suicidal' and 'clorazil', having respective corrections 'presystole' and 'gloriously'. To a human eye, such suggestions bear little, if any, resemblance to corresponding suspect words.

Specialised medical terminology, though, might legitimately be excluded from general-purpose checkers

such as Jazzy. Even so, less arcane words such as 'keyworker', 'untreatable' and 'whinging' failed to find dictionary entries. Corrections for truly misspelt words suffered a similar problem. Although 'agrophoia' was an error, the suggestion of 'agraphia' would be misleading. The intended word 'agrophobia' was not offered as a suggestion, revealing the limitations of dictionary files searched by Jazzy. Perhaps one or more additional dictionaries might yield a majority decision in such cases; even so, nothing short of a full list of English words could be seen as complete. Any such list would not be current for very long, due to the dynamic nature of English noted by Kukich (1993).

Results II for Jazzy Corrections Researched in GRiST Mind Maps

Several misspellings yielded just a sole suggestion. In 148 such cases, suggestions existed in one or more GRiST mind maps. Table 3.4 gives examples of sole corrections accepted in that way:

Non-Word	Suggestion	Non-Word	Suggestion	Non-Word	Suggestion
abusice	abusive	generalisied	generalised	oppossed	opposed
breavement	bereavement	helplesness	helplessness	pyschotic	psychotic
childhhood	childhood	ilness	illness	referal	referral
depresed	depressed	litrature	literature	shittiy	shitty
elderely	elderly	mpulsive	impulsive	trearful	tearful
finacial	financial	nihlistic	nihilistic	uncomon	uncommon

Table 3.4: Appropriate sole suggestions accepted by reference to GRiST mind maps.

In addition, various unrecognised words yielded several suggestions. In 41 such cases, the first suggestion was accepted on finding support in one or more GRiST mind maps. A selection of those corrections appears next as Table 3.5. The number of suggestions for any word from the Non-Word column is given under column n, followed by the first correction from any proffered list. Examples are ordered alphabetically by the number of suggestions:

Non-Word	Suggestions		Non-Word	Suggestions		Non-Word	Suggestions	
	n	1^{st}		n	1^{st}		n	1^{st}
absense	2	absence	neglectt	2	neglect	reqire	5	require
diference	2	difference	severly	2	severely	worring	5	worrying
familiy	2	family	diferent	3	different			

Table 3.5: Appropriate first suggestions accepted by referring to GRiST.

In cases of multiple suggestions, the first correction in any such list did not, sometimes, exist in GRiST's mind maps. The second suggestion from Jazzy, though, was found in those mind maps in 14 such cases, a selection of which appear next as Table 3.6. Preferred second suggestions in that table appear in bold type; examples are again ordered alphabetically by number of suggestions:

Non-Word	Suggestions			Non-Word	Suggestions		
	n	1^{st}	2^{nd}		n	1^{st}	2^{nd}
especialy	2	especial	**especially**	illnesss	3	illness's	**illnesses**
sayng	2	sang	**saying**	tenous	5	tenus	**tenuous**
seperate	2	sperate	**separate**	ligher	6	liger	**lighter**

Table 3.6: Appropriate second suggestions accepted by referring to GRiST mind maps.

Note that all second suggestions from Table 3.6 were longer than corresponding first entries; that was due entirely to Jazzy. Further, second suggestions were better than those offered before them.

Discussion II of Jazzy Corrections Researched in GRiST Mind Maps

Suggestions from Jazzy, then, were researched in a list of words from GRiST mind maps. Cases from Table 3.4 where Jazzy offered just a single correction showed various acceptable suggestions. Corrections such as 'psychotic' for 'pyschotic' proved Jazzy to contain certain mental-health terms, at least; such words, though, enjoy relatively common usage. That is further true of corrections such as 'referral', 'depressed', and 'nihilistic'. In jazzy's favour is the suggestion 'shitty' for 'shittiy' that shows a coverage of slang terms. Note, though, that the difference between, say, 'psychotic' and 'pyschotic' arises from transposing the letters 'sy' for 'ys'; in terms of edit distance, two substitutions constitutes a relatively small difference that should not challenge Jazzy. Support from GRiST mind maps, though, further boosts confidence in such corrections.

A similar approach yielded results from Table 3.5, except that any corrections appeared as the first entries in longer lists of suggestions. Finding corrections in GRiST mind maps terminated such searches immediately, ignoring any further suggestions: the first one proved acceptable. Suggestions were, though, relatively similar to any offending words; the word 'reqire', for example, needs just a single insertion of 'u' to make 'require'. Because such suggestions were but competing candidates, support from GRiST mind maps was all the more important in selecting the best correction.

Table 3.6 further presented corrections appearing second in any list from Jazzy. Interestingly, Jazzy presented shorter corrections before longer ones; take, for example, the misspelling 'ligher' that yielded six suggestions. The first one, 'liger', is a cross between a male lion and a tigress, whereas the longer subsequent entry 'lighter' was a far more likely replacement. For longer lists of suggestions, then, researching GRiST mind maps avoided inappropriate corrections in favour of longer, apposite ones.

Ignoring whatever mind map contained any non-word avoided taking support from the author responsible. Cutting and pasting non-words in FreeMind would lend false evidence of such words' true existence, whereas usage by a further author is more convincing. In fact, such research yielded no inappropriate corrections, regardless of position in any list of suggestions. That is not to say, though, that all spelling

errors were rectified in that way. Corrections that failed to find support from GRiST mind maps, then, were further checked by means of L'; results from that procedure are presented next.

Results IIIa for Spelling Corrections Checked by $L' = 0$

In the absence of any support from GRiST mind maps, then, proposed spelling corrections were analysed by means of L'. Suggestions at an adjusted distance of $L' = 0$ from corresponding unrecognised words were accepted without further checks; a selection of such corrections appears next in Table 3.7. Column L lists simple edit distances between words and any suggested corrections. Differences in length between words and suggestions appear under column d, with the resulting adjusted distance under L':

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
1	1	0	acurately	accurately	1	1	0	occurence	occurrence
			develomental	developmental				poteniate	potentiate
			embarassment	embarrassment				stupororus	stuporous
			forefully	forcefully				targetting	targeting
			hetrosexual	heterosexual	2	2		monsylabic	monosyllabic

Table 3.7: Appropriate spelling corrections accepted by $L' = 0$.

All but one of the suggestions from that table were the first or only one offered by Jazzy, as was the case for subsequent results in this section[1]. All of the inappropriate spelling corrections accepted by applying a criterion of $L' = 0$ are presented next as Table 3.8:

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
1	1	0	paradoxicaly	paradoxical	1	1	0	parkinsons	parkinson
			spliff	spiff				whinging	winging
			factly	fatly	2	2		ptsd	ptosed

Table 3.8: Inappropriate spelling corrections accepted by $L' = 0$.

Discussion IIIa of Spelling Corrections Checked by $L' = 0$

Suggestions from Jazzy, then, might not exist in any GRiST mind map. In such cases, corrections were compared with any word they sought to replace. To that end, L' quantified any similarity between non-words and candidate corrections. For example, the value of $L = 1$ between the replacement 'heterosexual' for 'hetrosexual' from Table 3.7 indicated just a single insertion, deletion, or substitution. The difference in those words' lengths, $d = 1$, gives $L' = L - d = 1 - 1 = 0$. That lengths differed by 1, then, accounts for all the variation noted by $L = 1$. In other words, a letter must have been inserted, rather than deleted or substituted; that is the letter 'e' inserted between 'het' and 'rosexual'. In a similar way, the correction 'monosyllabic' was accepted for 'monsylabic'. Although that case gave $d = 2$ and $l = 2$, the value of $L' = L - d = 0$ indicated two insertions: the letter 'o' after 'mon', and a second 'l' in 'sylabic'.

[1]The sole exception was the second suggestion 'wouldst' that replaced 'wouldnt'.

Table 3.7 showed the majority of suggestions accepted in that way to be suitable, with exceptions listed in Table 3.8. Of those inappropriate substitutions, 'ptsd' was an acronym for post-traumatic stress disorder rather than a real word. For the misspelling 'paradoxicaly', accepting the first suggestion that had $L' = 0$ led to abandoning the search in favour of 'paradoxical'. The more suitable word 'paradoxically' came second in the list, due to Jazzy's tendency to offer shorter suggestions first. Checking might be allowed to continue, stopping on the first unlikely suggestion rather than on the first reasonable one.

Further problems arose from accepting suggestions from single-entry lists, when any suggestion similar enough to the corresponding non-word was applied automatically. For example, lack of support for the word 'spliff' elsewhere in GRiST mind maps led to $L' = 0$ accepting 'spiff' as a replacement. In fact, the node `[patient x .half a spliff]` was a child of `[cannabis]` that in turn branched from `[substance misuse]` by way of `[drugs]`. Context, as Kukich (1993) and Wilcox-O'Hearn et al. (1998) note, might contribute knowledge that leads machines to better choices. That, though, would mean considering semantic distance, in order to link 'spliff', which is actually a cannabis cigarette, to related words that concern drug abuse.

Results IIIb for Spelling Corrections Checked by $d = 0$ and $L' = 1$

Whereas $L' = 0$ in Table 3.8 guaranteed acceptance, cases having $L' > 0$ were subjected to further checking by means of L, the standard Edit Distance. That metric was applied to non-words having suggestions of identical length, that is, when $d = 0$. An acceptance threshold of $L = 1$ in such cases gave the results in Table 3.9; the redundant L' column is retained to highlight that $L' = L$ when $d = 0$:

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
1	0	1	antecedant	antecedent	1	0	1	genisis	genesis
			collabaration	collaboration				perepheral	peripheral
			dysthimia	dysthymia				reduntant	redundant
			fundemental	fundamental				tranqyillisers	tranquillisers

Table 3.9: Spelling Corrections Accepted by $d = 0$ and $L = 1$.

While corrections from Table 3.8 were appropriate, others were less so. Such inappropriate corrections accepted by $d = 0$ and $L = 1$ appear next as Table 3.10:

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
1	0	1	aspergers	aspersers	1	0	1	detox	detor
			bugetting	begetting				rejecton	rejector
			cotard	dotard				wouldnt	wouldst

Table 3.10: Inappropriate Corrections Accepted by $d = 0$ and $L = 1$.

Discussion IIIb of Spelling Corrections Checked by $d = 0$ and $L' = 1$

Further to cases having $L' = 0$, Table 3.9 presented largely suitable corrections separated by $L' = 1$ from any suspect word. In those cases, suggestions were accepted should $d = 0$, indicating a corresponding non-word of identical length. In that way, the error 'tranqyillisers' was correctly changed to 'tranquillisers'. Identical lengths of 14 letters gave $d = 0$; a value of $L = 1$ further yielded $L' = 1$, indicating the sole substitution of 'u' for 'y'. In fact, the majority of corrections from Table 3.9 involved replacing one vowel with another, such as in 'peripheral' for 'perepheral, 'dysthymia' for 'dysthimia', and 'fundamental' for 'fundemental'[1]; that suggests a rule to be researched further. For now, the correction 'tranquillisers' in particular encourages links between forms of drugs, whether obtained on prescription or off the street.

The few exceptions that arose were listed in Table 3.10. Of those, the word 'aspergers' was a medical term missing from Jazzy's dictionaries, as was the missing word 'detox', a slang term for detoxification therapy undergone by drug addicts. Surprisingly, 'rejector' was the sole suggestion offered for 'rejecton'; 'rejection' would have been preferable, emphasising that suggestions can be checked only if they actually arise.

Results IIIc for Spelling Corrections Checked by $d = 0$ and $L' = 2$

While continuing to insist on identical lengths, allowing slightly more variation between words and suggested corrections produced the results Table 3.11, for cases where $d = 0$ and $L = 2$:

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
2	0	2	assualt	assault	2	0	2	sequelea	sequelae
			decieve	deceive				talior	tailor
			exagerrated	exaggerated				wierdly	weirdly
			heirarchy	hierarchy					

Table 3.11: Appropriate Spelling Corrections Accepted by $d = 0$ and $L = 2$.

The next set of results in Table 3.12 show inappropriate suggestions accepted by $d = 0$ and $L = 2$:

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
2	0	2	naomi	nomoi	2	0	2	respitory	raspatory
								underactive	interactive

Table 3.12: Inappropriate Spelling Corrections Accepted by $d = 0$ and $L = 2$.

Discussion IIIc of Spelling Corrections Checked by $d = 0$ and $L' = 2$

Results for $L' = 1$ between non-words and candidate corrections raised confidence in accepting such suggestions; results arising for $d = 0$ and $L' = 2$ from Table 3.11 further proved that approach. Having

[1]Treating 'y' as a vowel, as indicated by Phonics on the Web at `[http://www.phonicsontheweb.com/y-roles.php`.

adjusted for word length, allowing two edit operations identified and corrected spelling errors. Replacements for 'decieve', 'wierdly' and 'heirarchy' reflected authors' tendency to transpose the vowel pairs 'ie' and 'ei', which accounts for two substitutions found by $L = 2$. The first two examples were corrected by replacing 'ie' by 'ei', while the reverse worked for the non-word 'heirarchy'. The required two substitutions for 'talior', though, involved a consonant and a vowel, transforming 'li' to 'il' in the correction 'tailor'. Those letters, though, were contiguous, unlike those in 'exagerrated'. In that case, $L = 2$ reflected deleting a 'g' and inserting the letter 'r' to give 'exaggerated'.

Results IIId for Spelling Corrections Checked by $L' > 0$

So far, suggestions have been accepted for comparisons having $L \leqslant 2$ combined with $d = 0$. Non-words having suggestions of differing length, though, had $d > 0$. In such cases, corresponding values of L and L' differed accordingly. Suggestions for non-words were rejected should L' exceed zero, as Table 3.13 shows. Those results are sorted by L', which ranges from 1 to 7; acceptable suggestions that were wrongly refused appear in bold type:

L	d	L'	Non-Word	Suggestion	L	d	L'	Non-Word	Suggestion
2	1	1	agrophoia	agraphia	6	3	3	premorbid	pyromorphite
2	1	1	**colabarating**	**collaborating**	4	0	4	likelyto	ligulate
3	1	2	keyworker	coworker	6	2	4	clorazil	gloriously
3	1	2	**survelience**	**surveillance**	6	2	4	asbergers	icebreakers
3	0	3	seroxat	Xeroxed	9	2	7	parasuicidal	presystole

Table 3.13: Spelling corrections rejected by $d = 0$ and $L > 2$, or by $d > 0$ and $L' > 0$.

Discussion IIId of Spelling Corrections Checked by $L' > 0$

Raising the ceiling $maxL$ to 2, then, yielded the erroneous corrections from Table 3.12. Perhaps that 'Naomi' is a name might prevent it being replaced, although that would entail external sources of knowledge such as those studied by Sure et al. (2002) and Lin et al. (2005). Because the error 'respitory' arose from the node `[agitate respitory conditions]`, it seems reasonable to assume 'respiratory' as the intended word, though Jazzy did not offer that suggestion. Adding 'r' to that error to give 'respi*r*tory' forced the appropriate correction from a command-line check. Accomplishing that by reducing L from 2 to 1 indicates an underlying consideration of edit distance by Jazzy. A further unsuitable correction accepted by $L' = 2$ and $d = 0$ was 'interactive' for 'underactive', which should be hyphenated[1].

Results discussed so far may seem obvious to humans; suggestions from Table 3.13 that failed the test for $d = 0$ and $L - d = 2$, though, reveal the value of refining spelling corrections. Table 3.3, which showed unrefined Jazzy corrections, showed 'agraphia' replacing non-word 'agrophoia'. That correction,

[1] Jazzy duly accepts 'under-active' as valid.

though, was rejected by means of checking L'. Unfortunately, that does not much help the search for concepts in GRiST mind maps; rather than introducing the word 'agraphia' into the information base those mind maps comprise, 'agrophoia' was retained as a novel word. The desired word 'agrophobia' was not uncovered. On the other hand, rejecting the correction 'pyromorphite' for the medical term 'premorbid' meant retaining a valid and important word.

Two results from Table 3.13 had $d = 0$, indicating non-words and corrections of identical length. The first example, 'seroxat' was rejected as a replacement for 'Xeroxed' due to $L' > 2$. That value arose from $L = 3$, which gave $L' = L - d = 3$. A difference of three edit operations in words of identical length, then, was deemed too great. In a similar way, the suggestion 'ligulate' for 'likelyto' was rejected because of $L' = 4$. Although leaving 'likelyto' intact, that non-word is preferable to 'ligulate'; the sub-string 'likely' might yet match related words from GRiST mind map nodes. Regrettably, valid suggestions 'collaborating' and 'surveillance' were rejected. That they are long words suggests that values of $d = 1$ more acceptable in such cases, rather than $d = 0$.

Results IV for Corrections that Split Words

Lastly come results for composite words resolved by research in GRiST mind maps. All misspellings under the Non-Word column of Table 3.14 gave $L' = 0$ when compared with smaller words from those mind maps. Corresponding shorter words appear under the columns headed Separated. Three words from Table 3.14, 'anemia', 'hypo' and 'thoria', did not exist in any GRiST mind map, and appear in bold type. Non-words that failed to be split are likewise shown in bold:

Non-Word	Separated	Non-Word	Separated	Non-Word	Separated
alcoholicsdrug	alcoholic	courtproceedings	court	majorityclients	clients
	drug		proceedings		majority
causesanemia	causes	disinhibited	dis	otherrisk	other
	anemia		inhibited		risk
childcare	care	**hypothoria**	**hypo**	otherservices	other
	child		**thoria**		services
confusedmoving	confused	killthemselves	kill	preferedmethod	method
	moving		themselves		prefer

Table 3.14: Corrections that changed when further checks were applied.

One of the emboldened words from Table 3.14, 'anemia', was deemed valid on having Jazzy research the supplied U.S. dictionary. Conversely, 'hypo' and 'thoria' did not appear as separate words in any GRiST mind map, and was left intact.

Discussion IV of Corrections that Split Words

Results from Table 3.14 showed corrections that split misspellings into separate words. That was accomplished by comparing such misspellings with all other words from GRiST mind maps. Values of $L' = 0$ between composite words and those from mind maps meant that corresponding values of L arose entirely from insertions at the beginning or at the end of such smaller words. That, in turn, indicated a shorter word embedded in a longer one. Although one such sorter word, 'anemia', did not exist in any GRiST mind map, the U.S. dictionary researched by Jazzy reported it as valid. Researching that U.S. word in the U.K. dictionary yielded the more appropriate correction 'anaemia'.

Of all the conjoined words from Table 3.14, just one was left intact. The word 'hypothoria', though, is a medical term that well fits the information base of GRiST mind maps. Indeed, 'hypo' is actually a prefix that is best ignored as an actual word. Remaining composite words were successfully split by means of applying the criterion $L' = 0$. Take, for example, the words 'majority' and 'clients' that comprise 'majorityclients', or 'alcoholicsdrug' that yields 'alcoholics' and 'drug'. Separating such important words allows them to participate in the tuples $nodeID \rightarrow concept$ and $concept \rightarrow concept$. In that way, knowledge has arisen from nonsense, demonstrating the advantage of L' over L.

In contrast, standard L would have missed that aspect of words such as 'alcoholicsdrug'. Respective values of 4 and 10 between 'alcoholicsdrug' and the shorter words 'alcoholics' and 'drug' would reflect little similarity, particularly in the latter case. Those values of L, though, fail to reflect that 'drug', say, was appended to 'alcoholics' to make 'alcoholicsdrug'. Subtracting from L the difference in length between any two strings d, though, cancels out insertions to reveal smaller, valid words.

Supplementary Results for Corrections that Affect Stemming

The need for accurate atomic concepts was raised in Chapter 2; that, in turn, relies on stems of optimal length that neither over- nor under-stem words from GRiST mind maps. With that in mind, Table 3.15 shows corrections that will affect the length of any shared sub-strings between related words. Three stems are considered; for each stem, pairs of columns give groups of non-words to the left that, corrected, yield those on the right:

Stem: 'agress'		Stem: 'assess'		Stem: 'differ'	
Non-Word	Correction	Non-Word	Correction	Non-Word	Correction
aggession	aggression	assessing	assessing	diference	difference
aggresion	aggression	assesment	assessment	diferent	different
aggresive	aggressive	assessmenr	assessment	difering	differing
aggresively	aggressively			differeniate	differentiate

Table 3.15: Corrections that changed when further checks were applied.

Discussion of Supplementary Results that Affect Stemming

Various results from the four main sets were re-presented in Table 3.15 to show spelling corrections that affect stemming. Words from the first pair of columns expressed variations on the concept of aggression. Corrections for all four of those misspellings would add an extra 's' to the best stem for that group of words, giving 'agress' rather than 'agres'[1]. In the middle pair of columns, 's' is inserted into 'assessing' and 'assesment' to give 'assessing' and 'assessment' respectively. Otherwise, the stem 'asses' might be taken as the plural of 'ass'. In addition, the non-word 'assessmenr' is replaced by 'assessment', which already exists in the group. In that way, the number of words to consider while stemming is reduced. For the right-hand pair of columns, inserting the letter 'f' into words such as 'difering' and 'diferent' extends the stem 'difer' to 'differ'[2].

General Discussion of Spelling Correction Results

The approach presented in this section aimed to accept suitable corrections from Jazzy, while rejecting inappropriate suggestions. Using GRiST mind maps as an additional dictionary proved to be the single most successful way of achieving that end. All the same, taking into consideration L and L' successfully differentiated likely suggestions from those best rejected. The proposed approach to refining spelling corrections makes that of Petkovic, Kostanjcar, and Pale (2005) seem overly simplistic. In that study, suggestions separated from any non-word by 2, or sometimes 3, edit operations were accepted as corrections. The measure L' offered here proved more discerning in handling unrecognised words from GRiST mind maps.

As Navarro (2001) notes, though, L in any form is best applied to shorter texts; dynamic programming algorithms take unacceptably long to derive L between larger strings, due to comparing each source letter with every letter in any target string. Although all words from GRiST mind maps were researched when splitting composite non-words, L was applied to single pairs of words, and worked quickly. The relatively small amount of words in GRiST helped, there. Should larger repositories be examined, the `MindmapSpellChecker` Java programme might take unacceptably long to process words in that way.

Allowing corrections that exist as words in one or more mind maps yielded reliable results. No inappropriate corrections were accepted, regardless of where in any list of suggestions they occurred. Appropriate suggestions often appeared first in a list, sometimes by themselves. The first correction from longer lists was sometimes rejected in preference for the second suggestion. In those cases, researching a dictionary derived from the problem domain avoided inappropriate corrections. That all such corrections improved on Jazzy supports viewing of GRiST mind maps as a valuable body of knowledge.

[1] Agres is, in fact, a town in Valencia, Spain - see `http://en.wikipedia.org/wiki/Agres`
[2] Difer is actually a yacht manufacturer - see `http://www.difer-yachts.com/`

3.5 Qualitative & Quantitative Comparison with Jazzy

To illustrate what improvements accrue from using L for spelling correction, rather than algorithms such as Jazzy, the earlier summary of results is reproduced below as Table 3.16. Of the 374 non-words detected by Jazzy, 37 found no correction at all. Of the remainder, 212 were used in separate experts' mind maps, and accepted as valid words. Jazzy, then, was asked to supply just 125 suggestions for supposed non-words. Of those, 100 corrections were accepted because they were close enough to the corresponding spelling mistake. The remaining 25, though, were rejected, and the offending word left unchanged:

Note	Test Performed on Non-Word vs. Jazzy Suggestion	Counts
Rejected words	No suggestions from Jazzy	37
Accepted suggestions	Found as a word in GRiST mind maps	212
	$L' = 0$	59
	$d = 0$ and $L = 1$	30
	$d = 0$ and $L = 2$	11
Rejected suggestions	All suggestions failed the above tests	25
Totals		374

Table 3.16: Summary of Spelling Correction Results (Reprise).

Table 3.16 shows that considerably less of Jazzy's suggestions were accepted when subjected to additional processing. After further checking, what Jazzy deemed to be non-words were found more likely to be unknown yet valid words. That is the main improvement made by this thesis: instead of blindly accepting Jazzy's first suggestion, novel words were accepted as important concepts for the knowledge base of GRiST mind maps. In that sense, rejecting inappropriate suggestions was as important as any actual corrections made.

3.6 Chapter Discussion

In fact, two discussions follow now. The first compares the approach assessed here by experiments against contrasting techniques reviewed in Section 3.2. The second discussion highlights ways in which applying L' helped to refine suggested spelling corrections for non-words from GRiST mind maps. To begin, then, by considering that approach in relation to other work.

Spelling Correction by Means of L' in Contrast To Reviewed Approaches

Automated spelling corrections, while often inappropriate, are the best that any spelling checker might find. Any information base arising from GRiST mind maps, though, would suffer should misleading replacements be accepted; a more discerning approach, therefore, was brought to bear on correcting spelling errors automatically. That, in turn, demanded addressing the three challenges raised by Kukich (1993): detecting non-words, correcting isolated spelling errors, and considering the context of any misspelled words.

The first of those challenges was met by means of the Jazzy spelling checker; any word from GRiST that generated spelling suggestions was, it follows, a non-word. Rather than automatically taking the first suggestion for any non-word, though, further analysis determined the most acceptable, if any. Should that analysis reveal excessive variation between a given non-word and any suggested corrections, what might have been spelling errors were retained as valid novel words. To that end, refinement by means of L and L' effectively undid spelling errors by determining the number of inserted, deleted or substituted letters needed to transform a given non-word into any proposed correction. That process was illustrated on a superficial level by means of a word-ladder puzzle. In a similar way, though, the Edit Distance L quantified variation between text strings that might be seen as a far more challenging word ladder.

Indeed, Brill and Moore (2008) note that most spelling checkers use the Edit Distance to compare words. Rather than isolated words, though, Kernighan et al. (1990) addressed triples of words, comprising a spelling mistake and two candidate replacements. Variation between words in any triple, though, was limited to an Edit Distance of one: words in any triple differed by just a single inserted, deleted or substituted letter. In addition, human judges assessed such candidate corrections in advance, in order to provide a training dataset. In an observation similar to that made by Buckingham and Adams (2006) about pruning GRiST mind maps, that task proved more difficult and time consuming. In contrast, the approach taken here was fully automated, and required no training. Further, the measure used by Kernighan et al. (1990) was specific to English. In contrast, my approach based on L' is able to process whatever alphabet, even non-linguistic ones.

Working at a finer granularity than whole words, Kukich (1993) used n-grams of letters to generate reliable spelling corrections. More likely combinations of letters were treated as valid, even though they might not be recognised by any spelling checker. The approach described here, though, treated words primarily as strings of characters from a given alphabet; comparisons were limited to just pairs of letters from corresponding source and target words. All the same, whole words were important in studies by Wilcox-O'Hearn et al. (1998), by Kernighan et al. (1990), and by Mcnamee et al. (2001), which researched unrecognised words in the very texts under investigation. Those studies treated as valid any frequently

occurring non-words. A similar approach was used here, though in absolute terms, rather than by word frequencies. Non-words used by one or more separate GRiST authors were left intact, which admitted various specialised terms associated with mental health risks.

Although those studies highlighted the efficacy of using any body of text as a dictionary in itself, certain shortcomings were evident. Notably, Wilcox-O'Hearn et al. (1998) handled just those sentences containing a sole spelling error; in contrast, the approach described here copes with numerous spelling errors in any mind map node. In addition, Kernighan et al. (1990) underestimated the severity of spelling errors, allowing candidate corrections separated by just a single edit operation from any misspelling. That was further the case in a patent application made by Brill and Moore (2008). Indeed, they stress that approach's facility with strings of arbitrary size. That, though, is actually inherent in L, and true of any approach that uses it.

In a broader sense, context might cover the meanings of words; such approaches, though were rejected both by Kukich (1993) and by Wilcox-O'Hearn et al. (1998) as over-complex, and no better than purely text-based techniques. Concurring with that view, the proposed approach based on L and L' described in this chapter successfully corrected misspelled words, while leaving intact novel, though valid, words. Having, then, discussed the proposed approach to spelling correction in relation to other work, particular contributions to GRiST are addressed next.

Checking Spelling in Mind Maps by Means of L and L'

The overall aim of this chapter lay in identifying, and possibly correcting, non-words from GRiST mind maps. Misspelled words detract from the accuracy and coverage of any information base arising from those mind maps; inappropriate corrections, though, would be similarly damaging. Indeed, words such as 'parasuicidal' and 'clorazil' were not mistakes at all, but valid words unknown to Jazzy. Specialised medical terminology, though, might legitimately be excluded from general-purpose checkers; even so, the less arcane words 'keyworker' and 'untreatable' were likewise absent from Jazzy. Respective suggestions 'presystole' and 'gloriously', though, were rejected, and those novel words retained. Despite allowing for word-length, L' detected excessive variation between corrections and corresponding non-words.

In certain cases, suggestions such as 'psychotic' for 'pyschotic' were acceptable, proving Jazzy to contain at least some mental-health terms; such words, though, enjoy relatively common usage. Further, the correction 'shitty' for 'shittiy' showed a degree of coverage for slang terms. Longer lists of suggestions, though, benefited from researching candidate corrections in GRiST mind maps. Rather than taking the shortest suggestion that normally takes precedence in Jazzy, such research revealed better options. In that way, the non-word 'worring' was corrected to 'worrying' found elsewhere in GRiST mind maps, rather than the shorter suggestion 'warring'. Further, context in respect of those mind maps might

involve hierarchical clues. Although 'spliff' appeared just once in the entire collection, a parent node of `[drugs]` might discourage taking Jazzy's offer of 'spiff' as a replacement. That, though, would demand a consideration of semantic distance in addition to the Edit Distance.

Rejecting inappropriate suggestions, then, was of equal importance as retaining valid, though unknown, ones. A further advantage of applying spelling correction prior to the main task of extracting knowledge was in splitting long strings into separate words. In addition to removing invalid words, novel ones were generated in that way, for example, 'alcoholics' and 'drug' replacing 'alcoholicsdrug'. Whereas standard L between such words would indicate great variation, L' adjusted for word-lengths to reveal underlying shared sub-strings. Although inappropriate corrections continued to arise, refinements applied here reduced that number dramatically. As a result, non-words that might have been excluded in fact enriched the information base, and were allowed to participate in further analyses.

3.7 Chapter Summary

After introducing the need for spelling correction before fully analysing GRiST mind maps, an overview of automated spelling correction revealed three challenges: detecting non-words, correcting isolated misspellings, and accounting for the context of misspellings. Having shown ways in which spelling mistakes arise, and approaches to resolving such errors, this chapter proceeded with an approach to refining suggestions for non-words from GRiST mind maps. The popular Levenshtein algorithm, L, was introduced as a measure of Edit Distance between non-words and any alternatives offered.

Various studies, though, have refined that metric to allow for the lengths of any strings under comparison. Following that lead, the proposed L', accounted for word-lengths, ensuring that suggestions resembled any corresponding non-word. Subsequently, experiments assessed improvements arising from that approach to spelling correction, and discussed any results. This summary now closes the chapter, and turns attention to the role of L' in stemming.

4

Stemming for GRiST Mind Maps

4.1 Introduction

Chapter 2 showed that intensional knowledge might be extracted from the extensional variety held in GRiST mind maps. The dual processes of classification and normalisation yielded the tuple $nodeID \rightarrow concept$, records from which reflect particular ideas from a specified node. Such records allow GRiST mind maps to act as an information base of ideas about mental health risks. Rather than holding actual words, though, records from that tuple store shorter strings that match portions of related words. Within any group of such words, members will contain that sub-string, which is called a stem, and any process that identifies them, stemming. After spelling correction, then, stemming is the second challenge facing this automated analysis of GRiST mind maps. To that end, the Edit Distance that helped to refine suggested spelling corrections will further aid stemming.

Before addressing mind maps specifically, though, an overview describes current approaches to stemming in general. Existing studies will be shown to derive stems from isolated words, and further to account for context during stemming. Following that comes an exposition on extracting stems from GRiST mind maps by means of adjusted Edit Distance, L'. Subsequent experiments assess the proposed approach to stemming mind map concepts. Although results from each experiment are discussed as they arise, an overall chapter discussion assesses the value of applying L' to stemming in light of those experiments, before a summary closes the chapter. To start, then, with the promised overview of stemming.

4.2 An Overview of Stemming

In a review of approaches to automated text analysis, Aas and Eikvil (1999) note the ever-increasing amount of information on the Internet, and see a growing need for helping users to manage and search such resources; the time and money required to categorise information by hand makes automation increasingly attractive. To that end, a process of feature extraction might transform documents into a more manageable state. In that sense, feature extraction is a form of pre-processing that feeds any main algorithm under investigation (Aas & Eikvil, 1999).

An important part of such pre-processing is stemming, which removes suffixes from words to reveal so-called stems. Any specific stem identifies a group of words that express a particular underlying concept (Aas & Eikvil, 1999). Put another way, stemming is the search for what Mayfield and McNamee (2003) call the morphologically invariant portion of any word, while Xu and Croft (1998) refer to a process of reducing variant word forms to common roots. Similarly, Porter (1980) notes that Information Retrieval (IR) might improve should so-called term groups be conflated, that is, merged into a single term.

By whatever definition, stemming produces strings of letters that are not in themselves words. That end-users might find such stems difficult to interpret is unimportant. Rather, stems serve to facilitate IR by expanding words in any query with related forms. In that way, queries might retrieve a larger set of relevant documents. Now, although stemming might improve the recall of information, there is a risk of inappropriate stems retrieving non-relevant documents. The challenge lies in achieving the best trade-off between those opposing tendencies (Xu & Croft, 1998).

Approaches to stemming reflect the two categories that were seen earlier for spelling correction: processing words in isolation as opposed to considering context. As to the former, basic stemming might simply involve removing suffixes from plurals; researching tables of common word-endings is one way to do that (Xu & Croft, 1998). In a similar vein, a stem dictionary might be used (Porter, 1980). More advanced approaches to stemming isolated words employ so-called conflation algorithms. While some researchers embed linguistic knowledge in such algorithms, others treat words as just n-grams of letters. Approaches that consider context, on the other hand, use whatever text surrounds any stemmed word to assess the likelihood of being correct. Having introduced the two main approaches to stemming, attention now turns to various studies that employed them.

4.2.1 Deriving Stems from Isolated Words

Stemming, then, identifies groups of related words. For example, Aas and Eikvil (1999) derived the stem 'walk' for 'walker', 'walked', and 'walking' by removing the respective suffixes '-er', '-ed', and '-ing'. That the stem 'walk' is a word in its own right is of no particular importance; Xu and Croft (1998) point out that stems are better seen as n-grams that are embedded in related words. In fact, the algorithm that Aas and Eikvil (1999) used was what they call the popular Porter stemmer, which is discussed next.

The Porter Stemmer

Porter (1980) devised an ingenious approach of splitting words into n-grams that contained just vowels or consonants. N-grams composed of vowels were termed V, while C denoted n-grams of consonants. The length of any particular n-gram, then, depended solely on the number of contiguous letters of a specific type. Having identified V and C n-grams within any word, the Porter stemmer goes on to analyse repeating sequences of VC. In that way, words are reduced to recurring blocks that comprise one or more vowels followed by one or more consonants. The symbols $[C]$ and $[V]$ represented the first and last such sequences, should they exist. Between those extremes might be m repetitions of VC. The following expression, then, sums up the construction of any English word:

$$[C](VC)m[V] \qquad \text{(Porter, 1980)}.$$

Cases where $m = 0$ mean that words comprise just the $[C]$ and $[V]$ components. Words that do contain VC groups have $m > 0$; in those cases, the $[C]$ and $[V]$ components are allowed to be null (Porter, 1980).

Although Porter (1980) provides examples, they are not well discussed; what follows is an interpretation of some of those examples. Table 4.1 gives shows the construction of words having m values from 0 to 2. The middle columns of that table show $[C]$, VC and $[V]$ as described by Porter (1980), while the final column is my comment:

m	Word	$[C]$	VC	$[V]$	Comment
0	tree	tr	-	ee	Just $[C]$ and $[V]$.
1	trouble	tr	(ou bl)	e	V_1 = 'ou' C_1 = 'bl'.
2	troubles	tr	(ou bl) (e s)	-	V_1 = 'ou' C_1 = 'bl', V_2 = 'e' C_2 = 's'.

Table 4.1: Stems from the Porter stemmer, adapted from Porter (1980)

.

All of the rows from Table 4.1 have an identical $[C]$ component, 'tr'. In the first row, the word 'tree' was accounted for by just the $[C]$ and $[V]$ components, each having two letters. For $m = 1$, the word 'trouble' starts with 'tr' in $[C]$, while the $[V]$ component contains the final 'e'. In between comes a single VC group, with 'ou' and 'bl' in V and C respectively. The final entry contains the word 'troubles', for which $m = 2$. Note that taking the plural of 'trouble' raises m from 1 to 2; adding the consonant 's' to the vowel 'e' yields an additional VC group. Because of that, the terminating $[V]$ is left empty.

Having discovered a way of sub-dividing words, Porter (1980) goes on to specify five steps in his algorithm. Each of those steps bears a transformation rule that results in a shortened suffix. In addition, an optional condition stipulates when any particular rule should be applied. Those conditions used a shorthand notation based on the symbol '$*$'. For example, $*v*$ meant that any stem resulting from removing a suffix must contain a vowel, while $*o$ stood for a stem that ended with the pattern consonant-vowel-consonant (CVC), where the second consonant is not W, X or Y. Table 4.2, then, lists those five steps in stemming:

Step	Suffix Rule		Condition	Example	
	From	To		From	To
1	-sses	-ss	-	caresses	caress
	-ing	-	$(*v*)$	motoring	motor
2	-ational	-tion	$(m > 0)$	conditional	condition
3	-alize	-al	$(m > 0)$	formalize	formal
4	-ance	-	$(m > 1)$	allowance	allow
5	-e	-	$(m = 1 \ \& \ !*o)$	cease	ceas

Table 4.2: Step that comprise the Porter stemmer, adapted from Porter (1980)

.

The lack of a condition from the first entry from Table 4.2 means that '-ing' can always be removed. Conversely, the second example for Step 1 stipulates that any stem must contain a vowel; that was the case for 'motor', which actually contains two 'o's. The next step, number 2, specifies that '-ational' may be shortened to '-tion' just for words having one or more VC group. That same condition applies to Step 3, which adjusts, for example, '-alize' to '-al'. In a similar way, Step 4 applied just to words having at least two VC groups. Lastly, the '!' symbol in Step 5 negates the '$*o$' requirement for any remaining stem to end in CVC; that 'ceas' ends with the pattern VCV satisfies that condition (Porter, 1980).

Porter (1980) notes that the first step deals just with plurals and past participles. That step in fact had three discrete parts, although subsequent steps were seen as more straightforward. Importantly, the algorithm avoids removing suffixes that would result in too short a stem. The value of m was seen as a useful guide in that respect. For example, removing the suffix '-ate' from 'relate' having $m = 1$ would render the short stem 'rel'. The value of $m = 2$ for 'activate', though, leaves the adequate stem 'activ'. In practice, applying suffix-stripping steps to 10,000 words yielded 6,370 distinct stems. Stemming reduced the number of terms by about one third (Porter, 1980).

The Dual Problems of Under- and Over-Stemming

Over-aggressive stemming, though, risks obtaining stems that are too short to be useful. That risk recurs in research by Orengo and Huyck (2001), who note that what appears to be a suffix might actually part of any desired stem. Removing such false suffixes would produce stems that conflate unrelated words. That problem was overcome for the Portuguese language by specifying a minimum length for any stem; even so, avoiding over-stemming was difficult. For example, although the suffix '-inho' might indicates diminutive forms, 'golfinho' means 'dolphin', rather than a smaller form of 'golf'. In that case, it would be wrong to treat '-inho' as a suffix (Orengo & Huyck, 2001).

In order to alleviate that problem, lists of exceptions were created to specify words that are not, in fact, directly related by any given stem. The following entry for '-inho' stipulates such exceptions:

inho, 3, {caminho, carinho, cominho, golfinho, padrinho, sobrinho, vizinho}.

The first part of the entry is the suffix '-inho', while the number 3 dictates the minimum number of letters allowed in any stem resulting from removing that suffix. Words enclosed within braces {...} were treated separately from words reliably suffixed by '-inho'. Employing such exceptions reduced over-stemming mistakes by 5%. Conversely, under-stemming occurs when a true suffix is not removed, meaning that related words will not be fully conflated (Orengo & Huyck, 2001).

Shortcomings of Language-Specific Stemmers

Algorithms such as the Porter stemmer, though, suffer the serious drawback of being language-specific. Stemming any novel language will require new linguistic rules, which in turn demands a detailed knowledge of any language under investigation. In contrast, concentrating on n-grams of letters overcomes any reliance on embedded linguistic knowledge, and allows a language-neutral stemmer. That approach should work over a wide variety of languages (Mayfield & McNamee, 2003).

To that end, words were parsed to see if they contained n-grams that occur naturally. Those known n-grams were compared with sub-strings from words in question; each letter within any word was treated as the start of a new sub-string. Any words that contained a particular n-gram were conflated in that way. Such derived stems were further checked for over-stemming by means of a measure called the inverse document frequency (IDF). That reflected how many words would be conflated by any particular stem; those having high IDF were discarded as too general (Mayfield & McNamee, 2003).

That approach based on n-grams proved viable for eight European languages, of which no knowledge was built into the algorithm. The Wilkinson test of significance showed it to perform equally well as a language-specific stemmer, for some languages. The sole adjustment for any given language involved selecting a suitable length for the number of letters, that is, of 'n' for any n-gram. There was, though, an important prerequisite: a pre-compiled list of n-gram frequencies. All the same, calculating such frequencies was seen as a straightforward task. A more serious drawback concerned the performance penalty incurred by the high number of string comparisons required (Mayfield & McNamee, 2003).

4.2.2 Accounting for Context during Stemming

The Porter stemmer, then, is a rule-based algorithm that removes suffixes from words to reveal underlying stems; problems, though, arise from employing such programmes. Notably, what seems to be a suffix might in fact be part of a stem. Such over-aggressive stemming yields stems that conflate words having little shared meaning, such as 'pol' for 'policy' and 'police'. Conversely, lenient stemming risks the opposite effect, under-stemming, which fails to conflate truly related words such as 'matrix' and 'matrices'. Such under- and over-stemming might lead to serious failures in information retrieval (Xu & Croft, 1998).

Corpus-Based Stemming

A noted problem with the Porter stemmer concerns a failure to reflect actual language usage. Although rules in that algorithm are specific to English, that is not seen as a built-in linguistic model. Rather, such rules are designed to handle specific aspects of isolated words within a body of text, while neglecting any wider knowledge existing in such texts. An alternative approach of corpus-based stemming might rectify that shortcoming, by using what is called the co-occurrence of word variants. Put another way,

word forms that should be conflated for a given corpus will occur together in the very documents from that corpus. Unrelated words, on the other hand, should co-occur rarely (Xu & Croft, 1998).

To that end, a metric called EM, a variation of the Expected Mutual Information Measure (EMIM), measures the degree to which any word a co-occurs with word b within any corpus. Although the actual meaning of EM was left unclear, the following expression shows how it was calculated. Variables n_a and n_b are the respective occurrences of words a and b, and n_{ab} is the frequency with which a and b appear together. That actual frequency is compared with the expected number of co-occurrences, $En(a, b)$. The function $max()$ avoids negative results arising from the subtraction term:

$$em(a, b) = max(n_{ab} - \left(\frac{En(a, b)}{n_a + n_b}\right), 0) \qquad \text{(Xu \& Croft, 1998).}$$

My interpretation of that metric is that the expected co-occurrence of two words is divided by the sum of independent frequencies from the corpus. That ratio will be lower should two words' isolated occurrences exceed any expected co-occurrence. Conversely, higher ratios reflect isolated occurrences that are less frequent than are predicted co-occurrences. Subtracting the resulting ratio from actual co-occurrences gives a value for EM. In that way, higher ratios diminish any value for EM. Overall, relatively high EM values arise when two words often co-occur. Any given EM value, though, is reduced should such words appear by themselves relatively more than they do together.

In fact, the corpus in that study comprised text windows on a desktop machine. In that restricted environment, co-occurring word variants were seen to enhance the performance of stemming algorithms without a need for expert linguistic knowledge. Notably, that approach successfully avoided over-stemming the words 'company' and 'computer' to yield 'com'. That result, though, did not arise from applying the EM measure; rather, 'company' and 'computer' were assigned a value of $em = 0$ from the outset. Although 'company' and 'computer' share the same prefix, adjacent n-grams 'pan' and 'put' differ. Because of that, those words were deemed to be unrelated (Xu & Croft, 1998).

Statistical Approaches to Stemming

The Porter stemmer, then, might be criticised for its reliance on rules. Even so, such a rule dictated permissible n-grams in the co-occurrence approach of Xu and Croft (1998). In a similar way, Larkey, Ballesteros, and Connell (2002) note that designers of stemmers often build linguistic expertise into algorithms. In contrast, statistical methods promote conflation without resort to linguistic rules. Related words might be grouped purely by measuring the similarity of n-grams. So-called equivalence classes can be formed solely from words that share a particular n-gram of letters; the challenge lies in setting an appropriate threshold for the proportion of any related words that such n-grams must comprise (Larkey et al., 2002).

Although removing suffixes works well for English, that approach is less effective for languages such as Arabic. The problem there is that suffixing is just one so-called inflectional process that results in related word forms. Arabic words are additionally subject to in-fixing, which alters particular n-grams within words. For example, the word 'kitaab' means 'book', while 'kutub', 'katabaand' and 'naktubu' respectively mean 'books', 'he wrote' and 'we write'. The lack of any common sub-string in those words argues against n-gram analyses of Arabic text (Larkey et al., 2002).

Further difference between Arabic and English arose from a distributional analysis of newspaper text, in that more words occurred just once in the former. In addition, more distinct Arabic words appeared than did English words in a comparable sample. A dedicated Arabic stemmer removed just non-letters, weak vowels, and a few prefixes and suffixes. Such light initial stemming produced stems that conflated words into broad classes. Such preliminary stems were refined by means of the co-occurrence metric, EM, devised by Xu and Croft (1998). The resulting statistical stemmer depended on co-occurrence rather than on patterns of n-grams (Larkey et al., 2002).

4.3 Extracting Stems from GRiST Mind Maps

The Porter Stemmer, then, embodies linguistic rules about English. Although such rules aid stemming, Porter (1980) notes that rules for improving stemming in one area of a vocabulary might cause equal degradation elsewhere, and that successive layers of rules make analyses unnecessarily complex. In fact, the word 'rules' might overstate the role of linguistic knowledge in stemming. Rather, Xu and Croft (1998) prefer the word 'heuristics': looser common-sense rules that raise the probability of success. Determining such heuristics for stemming GRiST mind maps is the aim of this chapter.

Keeping Apart Unrelated Words

Rules in the Porter algorithm, then, drive a process of removing suffixes; prefixes, though, are not removed, in order to avoid inappropriate stems (Porter, 1980); although precisely how was left unclear, leaving prefixes attached to stems was seen as an advantage. The opposite view is vigorously taken here: prefixes and suffixes alike are seen as strings that do not belong in any stem. Should the prefix 'un', say, be the sole variation between any two words, they should be conflated by exactly the same stem. Indeed, whole words sometimes act as fixes on otherwise unrelated words, For example, the word 'fully' suffixes both 'carefully' and 'forcefully', despite 'care' and 'force' being unrelated. Any information base of GRiST mind maps must keep such words separate, while continuing to reflect that more distant relationship.

In fact, Orengo and Huyck (2001) overcame that problem by maintaining lists of exclusions that kept separate, for example, unrelated form of words suffixed '-inho'. Such exclusion lists will similarly separate

words from GRiST mind maps; that will involve records based on tuples of intensional knowledge, held apart from those mind maps. Those lists will, though, be compiled automatically, in contrast to the manually imposed exclusions specified by Orengo and Huyck (2001). In turn, that will facilitate queries made to any information base; as Xu and Croft (1998) showed, noted, users might subsequently choose what word variants to use in any particular query.

All the same, Xu and Croft (1998) disagree with the Porter Stemmer's conflation of 'addition' and 'additive'. Such Porter-like behaviour, though, is desired of stems for GRiST words, in that 'abuse' and 'abusive', like 'addition' and 'additive', are closely related. That the first word in each pair is a noun while second ones are adjectives is irrelevant, at least in terms of stemming. In that respect, analyses will be restricted to word-forms rather than accounting for any roles that words fill.

Degrees of Automation in Approaches to Stemming

Additional concern with reviewed approaches to stemming arises over degrees of automation, both in preparing to extract stems and in the process itself. A prerequisite for Xu and Croft (1998), for example, was the expected number of co-occurrences $En(a, b)$ calculated in advance by researchers. In a similar way, work by Mayfield and McNamee (2003) depended on a pre-compiled list of n-gram frequencies. In contrast, the approach here is to derive from scratch any important stems for GRiST concepts, in a fully automated way.

Accounting for Context during Stemming

A further key aspect of stemming considered by Xu and Croft (1998) and Larkey et al. (2002) was context, in the form of co-occurring word variants. In a similar way, GRiST mind maps will be researched to evaluate the coverage of stems. Rather than frequency of co-occurrence, though, any similarity between stems and conflated words will be assessed; should that indicate wide variation, stems will be rejected in favour of longer ones that conflate fewer, more closely related, word-forms. Indeed, Xu and Croft (1998) note that such sets might be huge: 165 stems in that study conflated more than 100 members; such stems are so general as to be useless, and will be avoided by the means just described.

A Linguistically Neutral Approach to Stemming

The proposed approach to stemming for GRiST Mind Maps, then, follows the linguistically neutral approach to spelling correction taken in Chapter 3. There seems little advantage in such additional complexity, given that Xu and Croft (1998) consider n-gram approaches to perform equally well as do stemmers imbued with linguistic rules. Indeed, Larkey et al. (2002) and Mayfield and McNamee (2003) note that stemmers are commonly specific to particular languages. Although studies reviewed in Section 4.2 covered various languages, they were treated in isolation according to differing syntactical rules. In contrast, the stemmer used here aims to be more generic, in considering words as strings of characters

rather than as linguistic entities. From that minimalist approach, though, will arise linguistic atoms such as prefixes and suffixes, regardless of any particular alphabet.

Rather than the n-grams employed by Porter (1980), Xu and Croft (1998), and Larkey et al. (2002), the Edit Distance will measure any similarity between strings. That, in turn, treats letters as n-grams just one character long in calculating the cost of turning one word into another. Words from GRiST mind maps will first be grouped by means of the adjusted Edit Distance L'; such words will be similar, having allowing for inserted letters. In fact, stems arise from inspecting matrices from calculating L, by means of the diagonal runs that Chapter 3 showed to reveal shared sub-strings between words.

The best stem for any group of words arises from checking for excessive variation between conflated words. Stems that conflate too wide a variety of words will be rejected, so as to avoid over-stemming. The opposing problem of under-stemming will be less troublesome, though; using L to group together words in the first place will ensure that just similar words are conflated. The real problem lies in avoiding stems that are too short. Having described how stems are to be retrieved from GRiST mind maps, then, attention now turns to experiments that were performed to test the approach.

4.4 Experiments in Extracting Stems

This section describes five experiments that assess L and L' as a means of stemming. First of all comes a general method that applied to all of those experiments, and a correspondingly generic overview of results. Following that come specific experiments that identify and refine stems from GRiST mind maps.

Method

Unique words extracted from GRiST mind maps in Chapter 3 further form the basis of stemming. Here, though, any spelling corrections from earlier experiments were applied during retrieval. Each unique word from that spell-checked list was compared by means of L against remaining members. Pairs of words were processed further should matrices from such comparisons yield a shared sub-string; comparisons giving no such sub-string were ignored, regardless of L or L'.

Any selected word-pairs were held as Java objects of the bespoke class `SortedL`, which implemented the `Comparable` interface to render them sortable. Such objects held pairs of words under comparison, any shared sub-string, and corresponding values of Edit Distance L, difference in word lengths d, and adjusted distance L'. In that way, `SortedL` objects constituted tuples of inputs to, and outputs from, calculating L. The actual sort order, then, was:

Sort field 1: l the length of any shared sub-string,
2: L' the refined Edit Distance,
3: d any difference in word lengths,
4: L the standard Edit Distance.

After sorting by means of the Java method `Array.sort()`, tuples were ordered primarily by ascending length of any shared sub-string. Within such groups of potential stems, successive tuples reflected increasing values of L'. Tuples having identical sub-string lengths and L' were further ordered by d, reflecting growing differences in length between source and target words. Further sorting by the standard L made absolute Edit Distance the most minor sub-division, for tuples having source and target words of identical length.

Categories of Tuples to be Processed

Tuples might occur by themselves, or along with further tuples for related stems. In fact, such numbers of tuples will fall into three categories, according to the numbers of words and candidate stems detected. Those categories are as follows:

Category 1: singleton tuples, reflecting solitary pairs of potentially related words,
2: groups of tuples that share a common stem, and
3: groups of tuples that suggest multiple stems.

Cases from category 1 reflect solitary pairs of possibly related words. Such cases were refined by means of L', the Edit Distance adjusted for string lengths. Acceptance criteria of $L' \leqslant maxL' \leqslant 2$ were applied to source and target words from sole tuples, in successive increments. Note was made of both appropriate and inappropriate stems accepted or rejected in that way, as was the case for subsequent categories. Lists of tuples from category 2 reflect multiple comparisons that yield a common stem. Such cases were refined by means of L'; potential stems were checked for $L' \leqslant maxL' \leqslant 2$ that applied to singleton tuples.

Whereas categories 1 and 2 comprise sole candidates, category 3 reflects competing stems. In such cases, a sole representative tuple was taken for each stem. Breaks in the main ordering, sub-string length, grouped together tuples bearing any given stem. Choosing the first member of such groups reduced lists of tuples to representative comparisons, which yielded any candidate stem. Lists of such representatives were processed sequentially until excessive L' arose between consecutive tuples. Should processing halt prematurely, stems were taken either from the arresting tuple, or from the one preceding. That depended on further checking $\overline{L_S}$.

The Average Length of Stems across Tuples: $\overline{L_S}$

In fact, $\overline{L_S}$ had two applications; the first considered absolute values of $\overline{L_S}$ for any tuple that might halt processing. The second application involved differences in $\overline{L_S}$ between consecutive representative tuples. Tests that consider aspects of L appear in the pseudo-code in Figure 4.1. Input comprises a list of one or more tuples, which are processed sequentially until reaching a tuple with excessive L' between given source and target words, as shown by pseudo-code in Figure 4.1 for a function called `getBestStem()`:

Pseudo-code for `getBestStem(i, tuples)`	Comments
int index = 0	Index of the selected tuple

for i from 1 to tuple count AND index < 0
  if tuple[i].$L' > maxL'$
  index = `getIndex(i, tuples)`
  end-if
end-for
	Process tuples for a given source word s
	Excessive variation between source & target:
	  get index of tuple offering the best stem

return tuple[index].stem	Take the stem from the selected tuple

Figure 4.1: Pseudo-code for the `getBestStem()` function used by Java class `MindmapStemmer`.

By default, the `getBestStem()` function from Figure 4.1 takes stems from the first tuple in any list, unless a better one emerges subsequently. For that reason, the loop commences at the second tuple, at index 1 in a zero-based array. If started at all, looping continues until reaching a tuple with excessive L' between given source and target words. Permitted distance is specified by the constant $maxL'$, which was varied from 0 to 2. The index of the tuple offering the best stem depends on a function called `bestStem()`.

Accepting and Rejecting Stems by means of L

Measures related to L and L' determine whether stems should be accepted, and if so, from what tuple. Tuples having L' within acceptable limits, specified by the constant $maxL'$, are not processed further; rather, the loop seeks to refine instances that suggest termination. The first stem to fail those tests causes remaining tuples to be ignored. The aim is to determine how far down a list of tuples should the process continue, before stems become too vague. To that end, `MindmapStemmer` calls the `getIndex()` function to determine what tuple holds the best stem.

The function `getIndex()` called by `bestStem()` from Figure 4.1 appears next as Figure 4.2, which refines stems arising from tuples that halted processing. Three checks applied there; the first involves checking standard L between source and target words from the current tuple. The second test checks for excessive variation between a particular stem and any conflated words, $\overline{L_S}$, while the third test compares that value on consecutive tuples. In those cases, stems are taken from preceding tuples at index $i-1$, instead of any arresting tuple at index i:

Pseudo-code for `getIndex(i, tuples)`	Comments
int index = i	Default to the row that halted processing
if $L \leqslant maxL$ index = $i-1$	Actual L for the current tuple
else if $\overline{L_S} \leqslant max\_avg\_L_S$ index = $i-1$	Average L between stemmed words
else if $\lvert L_i - L_{i-1} \rvert \leqslant max\_\Delta\_L$ index = $i-1$	Difference in L between ajdacent tuples
return index	Exit with the appropriate index

Figure 4.2: Pseudo-code for the `getIndex()` function called by `bestStem()` from Figure 4.1.

Any stems identified by implementing the pseudo-code from Figures 4.1 and 4.2 subsequently populated records from the tuple $nodeID \rightarrow concept$, associating specified nodes with stemmed concepts. Further, records from tuple $concept \rightarrow concept$ kept separate distantly any insufficiently related words arising from over-stemming. Attention turns next. then, to results from experiments in assessing successive stages from that pseudo-code, implemented in the `MindmapStemmer` Java class.

Results

In subsequent sections, the method, results and discussion for each experiment are presented together, before an overall discussion at the end of this chapter. All of those experiments involved running the Java `MindmapStemmer` class that implemented pseudo-code from Figures 4.1 and 4.2. Results from six experiments, then, are to be reported:

Part I : applying L' to singleton tuples,

Part II : applying L' to multiple tuples for a sole stem,

Part III : refining stems by standard L,

Part IVa : refining stems by absolute values of $\overline{L_S}$,

Part IVb : refining stems by $\Delta\overline{L_S}$ between consecutive tuples.

Supplementary results for prefixes and suffixes revealed by L'.

Subsequent results tables will follow a standard format. Columns headed Source Word list words that drive any comparisons, being words against which ostensibly related ones are judged. Those latter words, which share sub-strings with corresponding source words, appear under Target Word columns. Three further columns respectively show standard L between pairs of source and target words, differences in word lengths d, and adjusted distance L', that is, $L - d$. Following those numerical columns comes one for any sub-strings arising from comparisons. To preserve space, groups of such results may appear side by side, separated by a double vertical line. Additional columns, when needed, will show averages that reflect variation between stems and words, $\overline{L_S}$.

Method I for Applying L' to Singleton Tuples

Sole tuples, then, reflected solitary pairs of potentially related words. Stems from such tuples were refined by means of L', the Edit Distance adjusted for string lengths. Stems were rejected should L' between source and target words from any sole tuple exceed the constant $maxL'$. Conditions of $L' \leqslant maxL' \leqslant 2$ were applied in successive increments. Because looping starts at 1 in the function `getIndex()`, sole tuples would fail to enter that loop. Rather, such tuples were specifically checked for L'.

Results Ia for Applying $maxL' = 0$ to Singleton Tuples

Separate results arose from runs having the constant $maxL'$ set to 0, 1 and 2, and are presented in that order. In each case, both appropriate and inappropriate stems arose from sole tuples. To start, then, with examples of acceptable stems arising from tuples having $L' = maxL' = 0$, in Table 4.3:

Source	Target	L	d	L'	Stem	Source	Target	L	d	L'	Stem
amphet-amines	amphet-amine	1	1		amphet-amine	hopeless-ness	hopeless	4	4		hopeless
police	policeman	3	3	0	police	marital	extra-marital	5	5	0	marital
before	before-hand	4	4		before	hetero-sexual	sexual	6	6		sexual

Table 4.3: Appropriate longer stems from sole tuples, by means of $L' = 0$.

Although L varies from $1 - 6$ in Table 4.3, adjusted distances under column L' are identically zero; all of those cases yielded $L' = L - d = 0$. In addition, stems varied in length from 6 for 'sexual' and 'police' to 11 for 'amphetamine'.

Discussion Ia of Applying $maxL' = 0$ to Singleton Tuples

Sole tuples in this section, then, reflected stems from solitary pairs of words: any such candidate stem conflated just two morphological variations. Lacking competitors, such stems had just to be accepted or rejected. Cases having $L' = 0$ were special, though, in indicating word variants arising from just inserted characters. Examples from Table 4.3 showed such stems to be generally suitable. For example, 'before' and 'beforehand' are adequately related as to be conflated by 'before', as is 'hopelessness' by 'hopeless' and 'policeman' by 'police'.

Various prefixes arose from applying $maxL' = 0$ to negate insertions. For example, 'extramarital' yields the prefix 'extra-', while 'hetero-' arose from 'heterosexual'. In a similar way came the plural suffix '-s' from 'amphetamines', in addition to '-ness' from 'hopelessness'. Further, fixes identified by means of $L' = 0$ revealed shorter words that expressed more basic forms of related ideas.

Results Ib for Applying $maxL' = 1$ and $maxL' = 2$ to Singleton Tuples

Raising $maxL'$ from 0 to 1 gave further acceptable stems, which are listed on the left-hand side of Table 4.4; the right side of that table shows additional stems arising from incrementing $maxL'$ to 2:

Source	Target	L	d	L'	Stem	Source	Target	L	d	L'	Stem
adolesc-ence	adolesc-ent	2	1		adolesc-en	disabling	disability	3	1		disab
negative	negativity	3	2	1	negativ	literature	literacy	4	2	2	litera
bereave-ment	bereaved	4	3		bereave	retarded	retard-ation	5	3		retard

Table 4.4: Appropriate longer stems from sole tuples, by means of $L' = 1$ and $L' = 2$.

Tuples from the left side of Table 4.4 have $L' = 1$ as a result of d being always one less than corresponding values of L. In a similar way, tuples from the right have $L' = 2$. In contrast, standard L from that table ranged from 2 to 5. Further, stems varied in length from 5 for 'disab' to 9 for 'adolescen'. Such stems, though, did not arise from using, say, L and d to indicate any shared sub-string; rather, stems comprise diagonal 'runs' in matrices that resulted from calculating L. In that way, the stem 'negativ' arose from the shaded run of zero-costs shown in Figure 4.3:

		n	e	g	a	t	i	v	i	t	y
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
n	*1*	0	1	2	3	4	5	6	7	8	9
e	*2*	1	0	1	2	3	4	5	6	7	8
g	*3*	2	1	0	1	2	3	4	5	6	7
a	*4*	3	2	1	0	1	2	3	4	5	6
t	*5*	4	3	2	1	0	1	2	3	4	5
i	*6*	5	4	3	2	1	0	1	2	3	4
v	*7*	6	5	4	3	2	1	0	1	2	3
e	*8*	7	6	5	4	3	2	2	1	2	3

Figure 4.3: A run of zero-costs in the matrix for L between 'negative' and 'negativity'.

The matrix from Figure 4.3 describes a path of zero-cost for the first seven letters, 'negativ'. That cost, though, changes to 1 in cell $C_{8,8}$ on replacing 'e' by 'i', and again to 2 in $C_{8,9}$ to reflect the insertion of 't'. Cell $C_{8,10}$ subsequently detects the inserted 'y'; being the bottom right-most cell, $C_{8,10}$ further carries the ultimate value of $L = 3$.

Discussion Ib of Applying $maxL' = 1$ and $maxL' = 2$ to Singleton Tuples

Moving on to instances having $L' = 1$ revealed words having a further deleted or substituted letter, in addition to any inserted characters; that led to differences in lengths of $d = 1$. In that way, stems arose such as 'bereave' for 'bereaved' and 'bereavment', where 'e' was deleted before appending 'ment'. That deletion accounted for an extra edit operation, in addition to those that inserted a suffix. Rather than an insertion and a deletion, transforming 'adolescence' into 'adolescent' involved replacing 'c' by 't', before deleting the terminating 'e', accounting for $L = 2$. Similar reasoning explains converting 'negative' into 'negativity' by replacing the trailing 'e' with 'i', before adding 'ty'; those operations add up to $L = 3$.

By means of the matrix for comparing 'negative' and 'negativity' from Figure 4.3, a machine determined the shared sub-string 'negativ' of seven letters, even though comparisons were restricted to individual characters. In fact, of those examples of stems from sole tuples having $L' = 1$, just 'bereave' comprised a valid word. Otherwise, stems were strings of letters that were not words in themselves, a tendency noted by Xu and Croft (1998). Stems of any composition, though, serve to fill the *concept* field of tuples identified by abstraction, and do not need to be recognisable words. It is, rather, words that are conflated by such tuples that will be seen by humans.

Applying a More Lenient Limit for L'

Incrementing $maxL'$ to 2 covered words having two deleted or substituted characters, after allowing for insertions. For example, changing 'disabling' into 'disability' required changing the terminating letters 'ng' into 'ty', at a cost of two substitutions. Now, although those suffixes in full are '-ing' and '-ity', $L' = 2$ arose because the first letter of both, 'i', required no change. Note further that $d = 1$ reflects the insertion of 'i' after 'disab'. Together, $L' = 2$ and $d = 1$ account for the standard Edit Distance, $L = 3$. In a similar sense to 'disab', the stem 'litera' for 'literature and 'literacy' is not a real word; conversely, the stem for 'retarded' and 'retardation' reveals a novel word, 'retard'.

The suffix '-ity' that arose from $L' = 1$ between 'negative' and 'negativity' recurs on comparing 'disabling' and 'disability', which gave $L' = 2$. The latter, though, required inserting an 'i', that is, 'disab-i-lity', reflecting an additional edit operation. In addition, novel suffixes from comparisons yielding $L' = 2$ comprised '-ing', '-ed' and '-ation', respectively from 'disabling', 'retarded' and 'retardation'.

Results Ic for Inappropriate Stems from Applying $maxL'$ to Singleton Tuples

While sole tuples have offered suitable stems so far, various less apposite ones arose from lower values of L'. A selection of less appropriate stems follow as Table 4.5, all but one arising from $maxL' = 0$:

Source	Target	L	d	L'	Stem	Source	Target	L	d	L'	Stem
relying	lying	2	2		lying	wellbeing	being	4	4	0	being
sources	resources	2	2	0	sources	taking	under-taking	5	5	0	taking
towards	wards	2	2		wards	omni-potence	potent	6	5	1	poten

Table 4.5: Inappropriate longer stems from sole tuples, by means of L'.

Just a sole stem from Table 4.5 was due to L' greater than zero; that was, 'potent', with $L' = 1$. In contrast, values of L itself from Table 4.5 ranged from 2 to 6, which in all but that last case equalled corresponding values of d, giving $L' = 0$. Resulting stems varied in length from 5 for 'wards' and 'lying' to 7 for 'sources'.

Discussion Ic of Inappropriate Stems from Applying $maxL'$ to Singleton Tuples

Stems from source and target words separated by $L' \leqslant 2$, then, successfully conflated words relevant to mental health. Exceptions from Table 4.5, though, show the difficulty of stemming in the absence of 'real world' knowledge. All the same, just 6 inappropriate stems arose from considering $L' \leqslant 2$ for sole tuples. Of those, it could be argued that 'omnipotence' and 'potent' are related etymologically; the same might be said of 'sources' and 'resources', and to a lesser extent of 'being' and 'wellbeing'. Such is the difficulty in automating stemming: L' considers just morphological, rather than semantic, distance

between words. Even were L' so equipped, automated decisions would continue to require an appropriate acceptance threshold.

In fact, all but one unacceptable stem came from applying $maxL' = 0$. Those cases revealed whole words and corresponding pre- and suffixes; the problem remained of what constitutes a true fix. The word 'relying', for example, does not mean 're-lying', as in repeating untruths, or returning to bed; neither does 'towards' mean 'in the direction of wards'. On the other hand, 'sources' and 'resources' might share a relatively distant meaning, as might 'taking' and 'undertaking'. The point, though, is in making mind maps make sense; in that respect, under-stemming raises a real danger of conflating less related words, which would detract from GRiST mind maps acting as an information base.

Method II for applying L' to Multiple Tuples for a Sole Stem

Comparisons revealed so far yielded a sole tuple between solitary pairs of words. In contrast, comparing any source word against more than one target word yielded accordingly more tuples. Such lists reflect the latter categories introduced in the main method. Specifically, category 2 comprises multiple comparisons that yield a common stem; handling such cases, though, depended on the length of any stem. Short stems of 4 characters are considered shortly; for now, just longer stems are addressed. Potential stems of 5 or more characters, then, were subjected to the criterion $L' \leqslant maxL' \leqslant 2$ that was applied to singletons.

Results IIa for applying L' to Multiple Tuples for a Sole Stem

Results presented here, then, reflected several tuples that bore a common stem. In fact, few such cases arose; multiple tuples for longer stems more commonly yielded competing candidates, which are addressed in subsequent parts of these results. Of stems that had just to be accepted or rejected, then, examples of appropriate choices from applying $maxL' = 0$ appear next as Table 4.6:

Source	Target	L	d	L'	Stem	Source	Target	L	d	L'	Stem
hospital	hospitals	1	1		hospital	schizo	schizophrenia	7	7	0	schizo
	hospitalise	3	3	0			schizophrenic	7	7		
	hospitalisation	7	7				schizophrenics	8	8		

Table 4.6: Appropriate longer stems from multiple tuples for a single stem.

The stem 'hospital' from Table 4.6 was 8 characters long, and arose from comparisons having Edit Distances from $L = 1$ to $L = 7$. Corresponding differences in lengths between source and target words gave $L' = 0$ in all three cases. The six-letter stem 'schizo' was derived in a similar way: values of $L = 7$ to $L = 8$ were negated by respective corresponding values of $d = 7$ to $d = 8$. Note further that the source word and stem in that latter case are identically 'schizo'.

Certain stems derived by checking L' for multiple tuples, though, were less acceptable. Comparisons

that led to such inappropriate sole stems follow as Table 4.6, which shows that the stem 'place' arose from $L' = 0$, and further, 'appro' from $L' = 2$:

Source	Target	L	d	L'	Stem	Source	Target	L	d	L'	Stem
place	placed	1	1	0	place	approach	appropriate	5	3	2	appro
	places	1	1				appropriately	7	5		
	replaced	3	3				inappropriate	7	5		
	complacent	5	5				inappropriately	9	7		

Table 4.7: Inappropriate longer stems from multiple tuples for a single stem.

The two stems from Table 4.6 were both, as it happens, 5 characters long, and further conflated 4 words each, though respectively by means of $L' = 0$ and $L' = 2$. L itself ranged from 1 to 9 inclusively.

Discussion IIa of applying L' to Multiple Tuples for a Sole Stem

The reason for using L' rather than standard L is evident from Table 4.6. Considerably higher values of L were permitted than the maximum of 3 that Petkovic et al. (2005) took to indicate word variants. Indeed, the suitable stem 'hospital' was accepted despite having $L = 7$ when compared to 'hospitalisation'. The difference in those words' lengths, $d = 7$, explained any variation to be solely due to insertions. Appending the letters 'isation' to 'hospital' gave an adjusted distance of $L' = 0$, which was well within $maxL' \leqslant 1$.

Similarly large values of L were, in turn, negated by d to give the stem 'schizo'. That stem was, in addition, the source word on corresponding tuples, and therefore appeared in GRiST mind maps as an actual word. In contrast to the suffixes '-s', '-ise' and '-isation' appended to 'hospital', the whole word 'schizo' appeared to act as a prefix. That, though, was not truly the case; strings remaining after stripping that prefix, such as 'phrenic', are obscure words[1]. Although valid, such words would not reflect the core idea of schizophrenia. That problem will be revisited when discussing supplementary results later in this chapter.

Clearly undesirable stems, though, were accepted by applying $L' = 0$. Note, though, that while three target words were compared in Table 4.6 to give reliable stems, those from four comparisons in Table 4.6 were better rejected. Such increasing conflation might warn against judging any stem as optimal.

[1]The phrenic nerve is in the head, where schizophrenia arises: see `http://en.wikipedia.org/wiki/Phrenic_nerve`.

Results IIb for applying L' to Multiple Stems

A further key concept of 'suicide' recurs throughout GRiST mind maps. Two stems, though, arose from words expressing morphological variations on that idea: 'suicid' and 'suicide', as Table 4.8 shows:

Source	Target	L	d	L'	Stem	Target	L	d	L'	Stem
suicide	suicides	1	1	0	suicide	suicidal	2	1	1	suicid
	suicidecan	3	3			suicidality	5	4		
	parasuicide	4	4			suicidally	4	3		
						parasuicidal	6	5		

Table 4.8: The stem 'suicid' from an acceptably low $L' = 1$.

Competing stems 'suicid' and 'suicide' from Table 4.6 were respectively 6 and 7 characters long, and arose from comparisons having Edit Distances from $L = 1$ to $L = 6$. Corresponding differences in lengths between source and target words gave $L' = 0$ for the three cases on the left-hand side, and $L' = 1$ for a further four to the right.

Discussion IIb of applying L' to Multiple Stems

To a human eye, words from tuples in Table 4.8 were clearly related, although 'suicidecan' was a spelling error that escaped correction; that was because it rendered two words. Allowing a slightly higher Edit Distance of $L' = 1$ gave the better stem 'suicid', which further conflated related words from the longer stem 'suicide'. That was allowed by removing the letter 'e' to allow for suffixes such as '-al' that start with vowels. Indeed, that ideally depicts the advantage of L' over L, in that a relatively high L obscured the underlying stem. Allowing for differences in word-lengths by means of L' revealed various forms of an extremely important concept. Having shown success in applying L' to multiple stems, attention turns now to overcoming under-stemming, whereby candidate stems conflated too wide a range of words.

Method III for Considering Standard L

Tuples that yielded the stem 'suicide' had either $L' = 0$ or $L' = 1$. In contrast, multiple stems might carry varying values of L' that required further refinement. In such cases, a sole representative tuple was taken for each candidate stem. After sorting, breaks in the main ordering of sub-string length grouped together tuples bearing any given stem. Subsequently, finding the best stem from lists of representative tuples involved detecting the first unacceptable comparison between source and target words.

Representative tuples, then, were processed sequentially until encountering a tuple having $L' > 1$. Further checking values of standard L, as in the pseudo-code from Figure 4.2, showed whether processing should stop, or continue past any offending tuple. To that end, tuples showing $L > maxL = 4$ halted the loop, a stem being taken from the preceding tuple. Conversely, processing was allowed to continue should

$L < maxL$.

Results III for Considering Standard L

First of all, then, comparisons between a given source word and any similar target words yielded a list of tuples. As an illustration, Table 4.9 presents those from comparisons with the word 'abusing':

Source	Target	L	d	L'	Stem	Target	L	d	L'	Stem
abusing	using	2	2	0	using	abuse	3	2	1	abus
	abusive	2	0	2	abusi	abused	3	1	2	
						abuser	3	1	2	

Table 4.9: The stem 'abus' from an acceptably low $L = 2$.

The 5 tuples from Table 4.9 were condensed into the 3 representatives that follow as Table 4.10. In that table, and in following ones, values of L' that suspended processing are highlighted in italics; corresponding values of L that were consulted appear in bold type, as does any subsequently accepted stem. Using that notation, Table 4.10 depicts a value of $L = 2$ between tuples for 'abusing' and 'abusive', where the tuple at index 1 suspended the loop:

Source Word	Target Word	L	d	L'	Stem
abusing	using	2	2	0	using
	abusive	**2**	0	*2*	abusi
	abuse	3	2	1	**abus**

Table 4.10: Tolerate $L' = 2$, due to $L = 2$ between 'using' and 'abusive'.

The value of $L' = 0$ between 'abusing' and 'using' in Table 4.10 allowed consideration of the second tuple, between 'abusing' and 'abusive'. Although $L' = 2$ on that tuple exceeded $maxL' = 1$, the corresponding $L = 2$ fell below the constant $maxL = 4$, which allowed processing to continue. The stem 'abus', then, came from the last tuple in the list; that was the shortest stem on offer, due to sorting the list.

In contrast, results that follow in Table 4.10 have $L > maxL$. The given example shows words that shared a sub-string with 'unpredictable' to yield 11 tuples:

Source	Target	L	d	L'	Stem	Target	L	d	L'	Stem
unpredictable	predict	6	6	0	predict	predictable	2	2	0	predictable
	predicts	6	5	1		unpredictability	4	3	1	unpredictab
	predictive	5	3	2		tablet	9	7	2	table
	predictor	6	4	2		dictates	8	5	3	dicta
	prediction	6	3	3						
	predictors	6	3	3						

Table 4.11: Tuples arising from comparisons with 'unpredictable'.

The 10 entries from Table 4.11 were refined into representative tuples that were sorted into the order described in the method. That yielded the 5 tuples presented as Table 4.12:

Source Word	Target Word	L	d	L'	Stem	l
unpredictable	predictable	2	2	0	predictable	11
	unpredictability	4	3	1	unpredictab	11
	predict	6	6	0	**predict**	7
	tablet	**9**	7	*2*	table	5
	dictates	8	5	3	dicta	5

Table 4.12: Best stem 'predict' for 'unpredictable' selected, due to excessive $L = 9$ for 'tablet'.

In a similar way as for words expressing 'abuse', the 5 representative tuples from Table 4.12 were processed sequentially until excessive L' arose between 'unpredictable' and 'tablet'. Because $L = 9$ on that tuple so greatly exceeded $maxL = 4$, the stem 'predict' arose from the preceding tuple; in addition, the target word 'predict' was, in itself, a valid word. Further, the prefix 'un-' was detected, as were the suffixes '-able' and '-ability'.

Discussion III of Considering Standard L

By further checking comparisons having excessive L', standard L sufficed to select the appropriate stems 'abus' and 'predict'. Having identified candidates by means of L', which allowed for word-lengths, standard L succeeded in refining such stems. Given that $L' = L - d$, the difference in word-lengths d, rather than the Edit Distance L, might have been used to indicate excessive variation. Either measure would identify stems that conflate less target words in the corresponding group than would any preferred stem.

Halting on tuples having excessive L rejected stems arising from unacceptably dissimilar source and target words. Selecting the stem 'abus' involved choosing between three such alternatives. Competing stems for 'abus' were 'abusi', and 'using' that was in addition a mind-map word. In fact, 'using' appeared

as a stem on just a sole tuple, which might further detract from taking it as a stem. The reliable stem 'abus' was last in the corresponding list of tuples due to sorting representatives, ensuring that shortest acceptable stem was taken. By optimising stems in that way, `MindmapStemmer` found a balance between the twin dangers of under- and over-stemming.

The stem 'predict', though, arose from a more challenging six possibilities. All target words other than 'tablet' and 'dictates', though, contained the stem 'predict', making that the best choice. In addition, a representative tuple for 'predict' offered '-un' as a prefix, and '-able' and '-ability' as suffixes. Words from the full list of tuples showed further suffixes '-'s, '-ive', '-or', '-ors' and '-ion'; words having those suffixes would be conflated by the resulting stem 'predict', despite being absent from the analysis which yielded that stem. Having presented stems refined by means of L, then, attention turns to cases requiring a more discerning technique.

Method IVa for Considering Absolute $\overline{L_S}$

In further cases of excessive L' between source and target words, L could not differentiate between stems from the arresting tuple and the one preceding. Instead, `MindmapStemmer` checked any variation between words conflated by candidate stems. Should such variation exceed the maximum allowed, a stem was taken from the tuple preceding the one that halted processing.

To that end, `MindmapStemmer` calculated average Edit Distance in the following way. The first step was to combine source and target words from any tuple into single set. Using that set of n words, the Edit Distance L_S was calculated between stem S and any word w from list index i as follows:

$$L_S = L(S, w_i).$$

For any list of n words, dividing summed L_S values by n gives the average distance between a given stem and any words that it conflated. Accordingly, that average distance $\overline{L_S}$ is expressed as:

$$\overline{L_S} = \frac{\sum_{i=0}^{n} L(S, w_i)}{n}.$$

As a result of that calculation, individual tuples carried $\overline{L_S}$ values that measured variation between a given candidate stem and any conflated words. Subsequent sorting gave representative tuples that, by means of that extra field, reflected variation between such a tuple's stem and any corresponding full list of conflated words. Should any tuple arrest processing because of $L' > maxL' = 1$, corresponding values of $\overline{L_S}$ were compared against the constant $max\overline{L_S} = 4.0$; stems from any such tuple at index i having $\overline{L_S} \geqslant max\overline{L_S}$ were rejected in favour of stems offered by the immediately preceding tuple $i - 1$.

Results IVa for Considering Absolute $\overline{L_S}$

The following results reflect the second test from pseudo-code in Figure 4.2, invoked for tuples that, although having possibly excessive L', showed acceptable values of L. In such cases, $\overline{L_S}$ between stems and any source and target words was considered instead. An example of applying that measure comes from the word 'assess', for which Table 4.13 presents initial comparisons. Tables that follow have an additional column for $\overline{L_s}$, with values from arresting tuples highlighted in bold:

Source	Target	L	d	L'	Stem	$\overline{L_S}$	Target	L	d	L'	Stem	$\overline{L_S}$
assess	assessed	2	2	0	assess	4.25	classes	3	1	*2*	asses	**4.50**
	assessor	2	2				eyeglasses	6	4	2		
	assessors	3	3				assertions	5	4	1	asse	5.33
	assessing	3	3				assertive	5	3	2		
	assessment	4	4				harassed	5	2	3		
	assessments	5	5				embarrassed	8	5	3		

Table 4.13: All tuples arising from comparisons with 'assess'.

12 initial tuples from Table 4.13 gave rise to the 3 representative tuples in Table 4.14. The value of $L' = 2$ between 'assess' and 'classes' that halted processing appears in italics; further important values are highlighted in bold. A value of $L = 3 < maxL$ on the tuple for 'classes' did not justify rejecting the stem 'asses'; subsequently, $\overline{L_S} = 4.50$ on that tuple meant taking a stem from the preceding representative tuple, for the target word 'assessed':

Source Word	Target Word	L	d	L'	Stem	$\overline{L_S}$
assess	assessed	2	2	0	**assess**	4.25
	classes	**3**	1	*2*	asses	**4.50**
	assertions	5	4	1	asse	5.33

Table 4.14: Whole-word stem 'assess' selected, due to excessive $\overline{L_S} = 4.5$ for stem 'asses'.

Having suspended processing due to L' between 'assess' and 'classes', the first extra check for $L < maxL$ failed to indicate excessive variation. Further inspection of the offending tuple from Table 4.14 revealed that $\overline{L_S} = 4.50$ exceeded the constant $max\overline{L_S} = 4.0$. Consequently, the preceding representative tuple for comparing 'assess' and 'assessed' yielded the stem 'assess', in preference to 'asses'.

Discussion IVa of Considering Absolute $\overline{L_S}$

Excessive L', then, indicated that stems were becoming too vague, and processing was duly halted, leaving a machine to decide whether to take a stem from any arresting tuple, or from the immediately preceding representative. Although standard L has been shown to work in some cases, comparisons for words deemed similar to 'assess' showed no such glaring difference. Indeed, the tuple from Table 4.13 that

arrested processing due to excessive L' failed to exceed the maximum L allowed. That first supplementary check clearly needed refining.

In cases where L could not determine an appropriate stem, further analysis by $\overline{L_S}$ revealed what tuple bore the best one. Larger values of $\overline{L_S}$ reflected stems that conflated too wide a variety of words, leading to under-stemming; instead, an immediately preceding stem was taken in preference. In that way, obtaining the stem 'assess' involved reducing 12 initial tuples from Table 4.13 to the 3 tuples of Table 4.14. That well demonstrates the efficacy of using representative tuples to emphasise boundaries between competing stems.

While processing representatives from Table 4.14, values of L as large as 5 were allowed to pass because they reflected solely insertions, indicated by $L' = 0$. The first tuple with $L' = 2$ arose from comparing 'assess' with 'classes'. Although that tuple had an acceptable $L = 3$, the stem 'asses' was rejected due to excessive $\overline{L_S} = 4.50$. Put another way, the average distance between 'asses' and any words it conflated was too great. Though $\overline{L_S} = 4.25$ on the preceding tuple, for 'assess', was not that much smaller, the corresponding value of $L' = 0$ on that tuple meant accepting it without reference to $\overline{L_S}$.

Considering absolute values of $\overline{L_S}$, then, successfully determined optimal stems such as 'assess'. That relied on detecting excessive variation between a particular stem and any conflated words. Further cases, though, passed that test, and threatened to promote over-stemming by conflating too widely. The next experiment overcame that problem by way of differences in $\overline{L_S}$, rather than individual values.

Method IVb for Considering Differences in $\overline{L_S}$

Should no excessive value of $\overline{L_S}$ appear in any list of representative tuples, those that halted processing by having excessive L' were subjected to a further check based on $\overline{L_S}$. Rather than values from individual tuples, `MindmapStemmer` compared the spread of words conflated by stems from consecutive representatives. In short, $\overline{L_S}$ for a stem from tuple i was compared to that from preceding tuple, $i - 1$. Absolute differences in $\overline{L_S}$, then, are represented by $\Delta\overline{L_S}$, calculated as:

$$\Delta\overline{L_S} = |\overline{L_S}_i - \overline{L_S}_{i-1}|$$

Stems from tuples having $\overline{L_S} > max\overline{L_S} = 1.0$ were rejected in favour of a stem from any preceding tuple.

Results IVb for Considering Differences in $\overline{L_S}$

The application of $\Delta\overline{L_S}$ is well depicted by comparisons involving 'depressants'. In fact, 10 tuples carried target words such as 'depression', 'pressure' and 'present'; to preserve space, though, just the 4 resulting representatives are shown in Table 4.15. A value of $L' = 2$ on the arresting tuple for comparing 'depressants' with 'depressed' appears in italics; further relevant cells are shown in bold type. The value $L = 4$ matched, but failed to exceed, the allowed maximum, which led to taking the difference between $\overline{L_S} = 4.00$ in the arresting tuple and $\overline{L_S} = 3.72$ in the preceding row:

Source Word	Target Word	L	d	L'	Stem	$\overline{L_S}$
depressants	depressant	1	1	0	depressant	**3.72**
	depressed	**4**	2	*2*	**depress**	**4.00**
	press	6	6	0	press	3.81
	presence	6	3	3	pres	4.64

Table 4.15: Tolerate $L' = 2$ and $\Delta L = 3$, due to $\Delta\overline{L_S} = 0.28$ between stems 'depressant' and 'depress'.

A difference of 0.28 between highlighted values of $\overline{L_S}$ from Table 4.15 was much smaller than the limit of 1.0 specified by $max\_Delta\_L_S$. Consequently, the stem 'depress' was taken from the tuple that halted the loop, rather than from the one preceding it. Although full results were not presented, one unrefined tuple was worthy of note: that for the stem 'depress', between 'depressants' and 'depressive'. That tuple bore $L = 4$, $d = 1$ and $L' = 3$, which exceeded the maximum of $maxL' = 2$ for singleton tuples. A final point from these results concerns the novel suffix '-ant' arising from 'depressant', further to the recurring '-ed' from 'depressed'.

An example of correctly rejecting a stem by the means just described comes from comparisons for 'emotions'. In that case, 5 tuples yielded just the 2 representatives in Table 4.16, which highlights cells in the way described for Table 4.15:

Source Word	Target Word	L	d	L'	Stem	$\overline{L_S}$
emotions	emotion	1	1	0	**emotion**	**2.33**
	motive	**4**	2	*2*	moti	**4.00**

Table 4.16: Best stem 'emotion' for 'emotions' selected, due to difference in $\overline{L_S} = 1.67$.

In a similar way as for the stem 'depress', $L' = 2$ from comparing 'emotions' against 'motive' from Table 4.16 was unacceptable; $L = 4$ on that tuple, though, failed to exceed the maximum specified by the constant $maxL = 4$. Highlighted values of $\overline{L_S}$, though, gave a difference of 1.67 that exceeded the limit specified by $max\Delta\overline{L_S} = 1.0$. As a result, the stem 'emotion' arose from the preceding comparison between 'emotions' and that whole word, 'emotion'. In addition, the suffix '-s' recurred in 'emotions'.

Discussion IVb of Considering Differences in $\overline{L_S}$

The measure termed $\overline{L_S}$, then, reflects the degree to which stems conflate words. Stems were rejected should they differ too greatly, on average, to any words that were conflated. Conversely, low $\overline{L_S}$ indicates such variation to be limited. Rather than relying on discrete comparisons reflected by L and $L\prime$, considering averages encouraged more appropriate stemming.

The first example of considering $\Delta\overline{L_S}$ involved 'depressants'. A tuple with $L' = 2$ from a comparison with 'depressed' suggested excessive distance between those words. Taking the stem 'depressant' from the preceding tuple, though, would have missed the better stem 'depress'. Indeed, that epitomises understemming, which was avoided by a relatively low value of $\overline{L_S} = 0.28$. That showed the stems 'depressant' and 'depress' to conflate word variants having overall low edit distances. Further, had tuples been treated in isolation, $L' = 3$ between 'depressants' and 'depressive' would have been excessive. Considering average variation across the corresponding group, though, made 'depressive' suitably conflated by 'depress'.

Comparisons involving 'emotions' resembled those for 'depress', up to a point. As for that earlier set, a value of $L' - 2$ halted processing, in this case, for the stem 'moti' arising from comparing 'emotions' and 'motive'; $L = 4$ from that tuple was further accepted. A difference in $\overline{L_S}$ of 1.67, though, meant that the stem 'moti' conflated considerably less similar words than did the preferred stem 'emotion'. In combination with adjusting for word-lengths by means of $L' = 2$, stems were successfully refined by means of the average edit distance, $\overline{L_S}$, separating them from any conflated words.

Throughout this section, various prefixes and suffixes have arisen from applying measures based on the Edit Distance, L. The following supplementary results present a fuller review of such fixes, which brings together results from preceding experiments.

Supplementary Method for Fixes Arising from Calculating L'

Stems presented so far arose from shared sub-strings revealed by diagonal runs in matrices. Letters represented by cells preceding any such run were, in that way, potential prefixes; conversely, cells further down diagonals might have constituted suffixes. Instances of such candidate fixes were counted during the experiments described here. Frequencies in double figures, exceeding 9, indicated likely fixes.

Supplementary Results for Fixes Arising from Calculating L'

Preceding experiments yielded the prefixes that appear next as Table 4.17. The first row gives prefixes themselves; those shown in italics failed to qualify as common fixes. Frequencies on which that decision depended appear in the second row. Further, a double-bar column separator shows the division between rejected and accepted prefixes:

Prefix	*ac*	*ad*	*wi*	*as*	*be*	*im*	*ir*	*no*	*pr*	di	re	dis	de	ex	un	in
Count	4	4	4	5	5	5	5	6	7	10	12	13	14	15	25	36

Table 4.17: Frequencies of prefixes arising from stemming.

Further prefixes arose from whole words that attached to longer words from GRiST mind maps. Examples follow as Table 4.18, with whole-word stems in the first column, and conflated words in the second one:

Stem	Contained in
after	afterlife, aftermath and afterwards
fully	carefully, forcefully and successfully
where	nowhere, anywhere, elsewhere and somewhere

Table 4.18: Whole-word prefixes arising from stemming.

Those same stemming experiments yielded the suffixes in Table 4.19. Rows appear as two pairs; the first pair comprises suffixes that were rejected, as does the first entry in the second pair. Remaining entries on the second row, though, were accepted as true suffixes:

Suffix	*ally*	*atic*	*cy*	*ors*	*st*	*ty*	*nd*	*or*	*tic*	*us*	*ent*	*es*	*ity*	*rs*
Count	4	4	4	4	4	4	5	5	5	5	6	7	7	9

Suffix	*te*	ic	ness	able	er	ies	ce	ive	ion	al	ly	ed	ing	s
Count	9	10	10	11	12	12	17	19	27	32	49	58	76	117

Table 4.19: Frequencies of suffixes arising from stemming.

Accepted suffixes from Table 4.19 show high frequencies of occurrence, particularly '-ing' and '-s', found respectively on 76 and 117 different words.

Supplementary Discussion of Fixes Arising from Calculating L'

Insisting on double-figure frequencies before accepting fixes, then, separated incidental ones from those that regularly appeared in GRiST mind maps. That said, 'di-' is questionable; remaining prefixes, though, were appropriate. Note further the relatively low counts for prefixes compared to those for suffixes that follow shortly. Although prefixes were generally less frequent than were suffixes, reliable prefixes such as 'un-' emerged. The most frequent, 'in', arose from words such as 'inability', 'inevitability' and 'informally'. For 'inevitability', though, that is inappropriate; people do not commonly speak of 'evitability'.

Whole-word prefixes pose a further problem, in that they might conflate words that are too distantly related. Take, for example, the stem 'fully' that would identify 'carefully', 'forcefully' and 'successfully'. Without a semantic component to stemming, such groups of words will appear similar to machines. All the same, the benefit arises of having revealed such words. Indeed, tuples of the form *concept* $\rightarrow$ *concept* might serve both to bring related concepts together or to keep them separate. That would entail a *type* field to indicate that relationship: a minor addition. In that way, users of the GRiST mind-map information base might at least be offered a choice in the treatment of such words.

Suffixes, on the other hand, reached higher frequencies than did prefixes. The highest count of all was for the single-letter suffix 's', although 'ed', 'ing' and 'ly' were similarly reliable due to high frequencies of occurrence. In contrast, less frequent suffixes such as 'ally', 'atic' and 'ors' actually reflected the shorter suffixes '-ly', '-ic' and '-s' appended to words ending respectively in 'al', 'at' and 'or'. Those longer strings, then, are not true suffixes, and were correctly rejected. That positive note concludes experiments that derived stems by means of L in various forms. There now follows an overall discussion of those experiments, and what they suggest in respect of identifying atomic concepts from GRiST mind maps.

4.5 Qualitative & Quantitative Comparison with Porter

What improvements, then, accrue from using L for stemming, rather than the Porter stemmer? To determine the answer to that question, both stemmers were run against a list of 3,989 unique words extracted from GRiST mind maps. Whereas the Porter stemmer extracted 2,834 unique stems, the L-based approach developed in this thesis found just 1,504.

In addition, secondary outputs from that latter purely textual approach included a list of English prefixes and suffixes, as Section 4.4 has just shown. The Porter stemmer, on the other hand, left prefixes intact to yield distinct stems such as 'unpredict' and 'unsafe'. In contrast, the new technique developed here gave the stems 'distinct' and 'safe' that conflated related words that were prefixed with 'un'. A similar result arose for the Porter stems 'appropri' and 'inappropri' that carried the prefix 'in', with the

new L-based technique giving just 'appropriate'.

Table 4.20 gives examples of differences in stems from those two approaches; on the left hand side are those from the Porter stemmer, with those arising from variants of L to the right:

Porter Stems	L Stems
alcohol, alcoholicsdrug	alcohol, drug
aphetaminescannabi	aphetamine, cannabis
medic, medical, medicin	medic
parasuicid, suicid	suicid
parano, paranoia, paranoiac, paranoid	paranoi
violenc, violent	violen
withdraw, withdrawn	withdraw
young, younger, youngster	young

Table 4.20: Porter stems compared with those arising from L and L'

.

In all but the first two results from Table 4.20, several stems were extracted by the Porter stemmer for words that, in fact, were directly related. In contrast, single stems arose from applying L and L' that conflated the various forms found by the Porter stemmer.

The novel stemmer produced here, then, showed both qualitative and quantitative improvements over the Porter stemmer. Fewer stems arose from applying L than from the stemmer provided by Porter (2006), and those that were produced proved more accurate in conflating related words. By offering fewer stems, the approach based on L helped to avoid over-stemming, where too narrow a set of words is conflated. Indeed, the Porter stemmer did just that, surprisingly giving distinct stems that were morphologically very close, such as 'young', 'younger', and 'youngster'.

The first entry from Table 4.20, though, was a non-word caused by a missing space character. While the Porter stemmer took 'alcoholicsdrug' as a separate stem to 'alcohol', the new approach derived two distinct words, 'alcohol' and 'drug', that conflated related words such as 'alcoholic' and 'drugs' in further GRiST mind maps.

Note too how the Porter stemmer assumed that the 's' on the end of 'aphetaminescannabi***s***' indicated the plural of 'aphetaminescannabi'. In such cases, linguistic knowledge embedded in the Porter stemmer actually had an adverse effect. In contrast, L successfully separated those conjoined words to generate the desired stems. Having split the original non-word, the missing 'm' from 'aphetamines' gave a spelling error; spell-checking any resulting words ensured the correct stem 'a***m***phetamine', which conflated that word itself, as well as the plural 'amphetamines'.

4.6 Chapter Discussion

There are, in fact, two discussions that follow now. The first compares the approach assessed here by experiments against contrasting techniques reviewed in Section 4.2. The second discussion highlights stronger and weaker aspects of using L'. To begin, then, by considering that approach in relation to other work.

Stemming by Means of L' in Contrast To Reviewed Approaches

Extracting stems from GRiST mind maps reflected a process called feature extraction, whereby Aas and Eikvil (1999) transformed knowledge into a more manageable state. To that end, stemming revealed groups of words that expressed a particular underlying concept. That stemming produced strings of letters that were not themselves words was unimportant; rather, stems serve to improve access to knowledge from GRiST mind maps. The challenge, as Xu and Croft (1998) pointed out, lay balancing opposing tendencies of under- and over-stemming.

To an extent, applying L' resembled the inverse document frequency (IDF) described by Mayfield and McNamee (2003). That measure reflected the numbers of words that would be conflated by any particular stem; those having high IDF were discarded as too general. That was achieved here by means of the average distance between stems and any conflated words, $\overline{L_S}$. Excessive values of that average variation led to rejecting stems that conflated too wide a range of words. Further, the performance penalty said to arise for IDF from high numbers of string comparisons was not a problem; experiments processed an entire list of unique words from GRiST mind maps in matters of seconds.

The Porter Stemmer used by Aas and Eikvil (1999) relied largely on removing suffixes, and further embedded knowledge about English. In contrast, the approach used here measured variation between words by means of the Edit Distance, L. Having rejected removing suffixes as a means of stemming, such fixes actually arose from that process, in addition to several common prefixes. Identifying such fixes, then, required no linguistic rules. In turn, reliable fixes might further seed a second pass of `MindmapStemmer` armed with knowledge about fixes arising from this first pass. Fixes would be valuable in assessing stems by indicating sub-strings that might well be inserted. Further, combining stems with fixes would generate words which, checked by Jazzy, would provide further evidence about any stem's coverage.

Further divergence from the stemmer invented by Porter (1980) lay in treating individual letters, rather than recurring n-grams of several characters. On the other hand, context in the sense of co-occurring word variants used by Xu and Croft (1998) was not considered, here. Even so, measures based on L avoided the over-stemming that a consideration of context sought to avert. Notably, that approach avoided

over-stemming the words 'company' and 'computer' to 'com', due to the differing respective n-grams 'pan' and 'put'. Rather using, say, co-occurring word variants, measures based on L used here avoided over-stemming in such cases as 'asses' which, while stemming words such as 'assessor' and 'assessment', further included 'harassed'. The slightly longer stem 'assess', selected by means of L, proved ideal.

As Orengo and Huyck (2001) point out, over-aggressive stemming might remove strings that, although treated as suffixes, are actually parts of words. That was seen for, say, the suffix 'ally', of which the letters 'al' were, in fact, an integral part of words such as 'formal'. Such longer false suffixes were successfully rejected due to low rates of occurrence. Deriving fixes rather than relying on dictionaries of such strings, then, proved more accurate, and therefore more useful. In just what ways that might be for GRiST mind maps is considered next.

Stemming Mind Map Concepts by Means of L and L'

The overall aim of this chapter was to identify and organise concepts from GRiST mind maps, which were seen as belonging to a wider set of related representations known as semantic networks. Mind maps, though, stand apart from other types of semantic network by lacking any mechanism for controlling what is recorded; put another way, no intensional knowledge exists that might dictates specific forms to which knowledge must conform. Intensional knowledge, in turn, results from a process of normalisation, which usually precedes creating any actual data. In contrast, the approach taken here was to derive intensional knowledge from existing GRiST mind maps, in order to map any concepts from that collection. Just as for other types of semantic network, that intensional knowledge was stores separately, leaving any original mind maps unchanged.

Intensional knowledge from stems identified in this chapter comprise a first step towards a more formal network. Indeed, GRiST mind maps served that purpose for Buckingham and Adams (2006), who extracted concepts manually for a more formal Galatean semantic network. Although mind mapping is seen as little more than a brainstorming tool, though, those mind maps themselves constitute an information base. Intensional knowledge from normalisation makes mind maps a valuable body of knowledge that machines might research. Further, techniques demonstrated here could readily be applied to mind mapping in general. Providing intensional knowledge would help users to organise mind maps from whatever domain was represented. By reducing related words to a common stem, nodes expressing whatever form of a particular concept will be grouped together.

Adjusted versions of L successfully identified three such important GRiST concepts: abuse, depression, and suicide, by means of respective stems 'abus', 'depress', and 'suicid'. Tuples made up of a unique node identifier and a stem plot instances of those concepts within GRiST mind maps. The standard form of L could not have achieved such success; differences in word lengths raised relatively high values of L

between, say, 'suicide' and 'parasuicidal'. Allowing for word length, though, revealed that what might have constituted excessive variation was due, in fact, to adding fixes. Having automatically determined frequently occurring fixes, future research might consider a second pass that further refines stems in light of those fixes. Although automated approaches are unlikely ever to be wholly accurate, L provided relatively few unsuitable stems. Even those might improve on making more discerning adjustments to L.

4.7 Chapter Summary

After introducing the idea of stems, various approaches to that process were reviewed before presenting a technique for stemming the forty six mind maps that Buckingham and Adams (2006) created for the GRiST project. By means of L', the adjusted Edit Distance, key concepts of abuse, depression and suicide resulted from cross-comparing words found in GRiST mind maps. By holding concepts as stems, related words might be mapped between mind map nodes. In that sense, stems contributed towards a repository of intensional knowledge by reflecting what Mayfield and McNamee (2003) call the morphologically invariant portions of words. Having introduced that approach based on L', experiments followed that identified and refined stems by allowing for words of differing lengths. Subsequently, qualitative and quantitative improvements over the Porter stemmer were presented. Following an overall discussion of those experiments in relation to an information base of GRiST mind maps, this summary closed the chapter.

Part II

Applying Knowledge to GRiST Mind Maps

5

Resolving Ambiguity in GRiST Mind Maps

5.1 Introduction

Chapter 1 showed GRiST mind maps to diverge from the ideal proposed by Buzan (1996), particularly in regard to the rule demanding single-word nodes that was dismissed as excessive. Reducing ideas to such a Bag-Of-Words (BOW) would require humans and machines alike to reconstruct any original meaning, during which nodes and, and hence ideas, might be recombined incorrectly. Permitting more expressive nodes avoids that problem of interpretation for humans, and faithfully represents intended ideas. In fact, approaches based solely on classes discard many useful words, destroying semantic relations between any that remain (Bekkerman & Allan, 2005).

The richness of mind maps from GRiST, then, contrasts with other semantic networks such as concept maps, which employ tangible constructs, as did ontologies created in OntoEdit. Nodes from those latter networks were derived from nouns that represent concrete things. In a similar way, Knowledge Bases formulated in Description Logics are based on unary predicates, or classes. Subsumption is a further important aspect of such formal knowledge structures, as in the given example of a concept D that subsumed a further concept C, indicating that all C are specific types of D (Nardi & Brachman, 2002).

In contrast, galateas from GRiST do not reflect strict subsumption hierarchies. Rather, GRiST introduced Membership Grades (MGs) to reflect degrees of association between concepts. That allowed the galatean model to treat concepts as members of sets, rather than as a strict subsumption hierarchy. In that way, galatean hierarchies decompose concepts such as self-neglect into constituent elements (Buckingham et al., 2004). It is important to note that high-level concepts could share constituents: this would not be allowed in a subsumption model. Further, less formal hierarchies allow complex concepts, using several words. That tendency is far more marked in GRiST mind maps. As nodes branch from a central idea, concepts become increasingly more specific and detailed (Hegazy & Buckingham, 2008).

That more relaxed approach to galatean and mind map structures yields nodes that do not stand for discrete objects. For example, prepositions and conjunctions indicate relationships between more specific concepts. The need arises, then, for machines to distinguish between words that express concepts, and other words that tie them together. That process builds on the stemming described in Chapter 4, which identified the word forms taken by core concepts. This chapter focuses on what mind map authors meant when using such words to compose nodes that constitute sentences. To that end, a popular tool called WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1990) is employed here. After an overview of WordNet, two challenges are shown to arise in using it. Overcoming those challenges will involve processing prepositions, in particular. Specifically, patterns of preposition usage will provide valuable clues as to the meaning of ambiguous words. First, though, here is an overview of WordNet

5.2 An Overview of WordNet

Chapter 2 introduced the Jazzy spelling checker as a useful store of lexical data, that is, of information about words. The creators of WordNet, though, see algorithms such as Jazzy as mere rapid page-turners, which ignore ways in which humans organise lexical knowledge. WordNet, in response to that shortcoming, arose from psycholinguistic studies that revealed a mental lexicon comprising four Parts Of Speech (POS): nouns, verbs, adjectives and adverbs. Those four POS are content words, while remaining POS-types are function words. WordNet, then, emulates lexical memory by using separate stores (Miller et al., 1990). Because WordNet's design so influences any application, justification follows for fashioning it in that way.

The Organisation of WordNet

Justification for isolating nouns in WordNet came from patients suffering from anomic aphasia. Strokes in the left-hemisphere of the brain had left sufferers unable to name objects, suggesting a discrete physical location for nouns (Miller, 1990). Further, that discrete lexical memory for nouns describes a semantic network, in which nodes represent superordinate terms, or hypernyms, shown by the symbol @→ (Collins & Quillian, 1969). In that way, a branch concerning canaries was represented as:

> canary @→ finch @→ passerine @→ bird @→ vertebrate @→ animal.
>
> (adapted from Collins & Quillian, 1969).

Those hypernym relationships suggest that all canaries are finches, and ultimately, animals. Rather than a BOW, then, memory for nouns forms a hierarchy of progressively more specific hypernyms. A selected branch from that network forms Figure 5.1 overleaf, which further depicts attributes such as 'Has Skin' and 'Can Fly' as arrows emerging from nouns at each level:

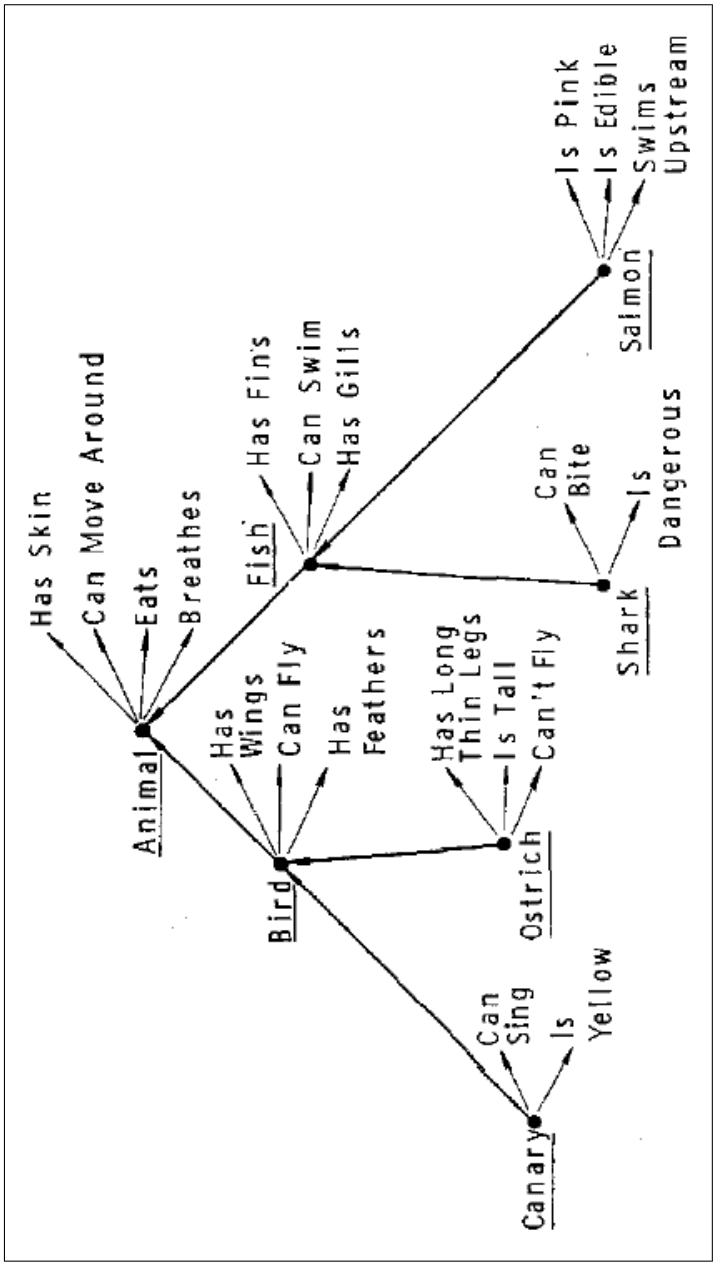

Figure 5.1: A semantic network of nouns and attributes, adapted from Collins and Quillian (1969)

By means of the semantic network from Figure 5.1, responding to assertions such as 'a canary can sing' involves descending the network as far as 'canary', then taking the attached attribute 'Can Sing'. However, deciding that 'a canary can fly' involves moving up one level from 'canary', having identified that concept, to 'bird' which holds the attribute 'can fly'. Indeed, experiments showed response times for answering 'true' or 'false' to such assertions increased with the number of levels traversed, supporting the view of nouns as a semantic network (Collins & Quillian, 1969).

In fact, nouns are the sole POS that characterise discrete objects; conversely, verbs have no concrete existence. In further contrast to nouns, verbs exhibit lexical entailment rather than inheritance. Entailment means that one verb necessarily involves another. For example, 'snoring' entails 'sleeping': to truly snore, one must be asleep, although the opposite is not true. That difference in lexical relationships led to holding verbs separately in WordNet (Fellbaum, 1990; Miller et al., 1990).

WordNet collectively treats remaining POS, adjectives and adverbs, as modifiers that qualify nominal and verbal concepts expressed, respectively, by nouns and verbs. Unlike nouns, modifiers have no synonyms, that is, words of equivalent meaning; such differences in usage justify holding apart modifiers from other POS (Fellbaum, Gross, & Miller, 1990). Further support comes from the bi-polar nature of modifiers, that is, how they might express opposed extremes such as 'hot' and 'cold' (Fellbaum, 1990).

Whether partitioning POS in that way was justified is not an issue, here; rather, it is the various theoretical influences on WordNet's designers that is of interest. While Jazzy uses distinct dictionaries just for convenience, WordNet applies psycholinguistic principles to justify separating POS. Although Jazzy adequately corrected spelling errors from GRiST mind maps, WordNet will build on that by eliciting connections between concepts. To that end, attention turns next to the mechanism by which WordNet holds related words.

WordNet as a Semantic Network

Having portrayed mind maps as semantic networks in Chapter 2, then, the WordNet tool for further refining GRiST mind maps holds lexical knowledge in a similar way. In fact, WordNet covers both the forms and the meanings of words, respectively known as morphology and semantics. The relationship between form and meaning is many-to-many: specific words might have multiple meanings, while several words might express a sole meaning. The word 'board', say, might mean a plank of wood or a group of executives, a phenomenon called polysemy. Further, so-called senses reflect differing meanings for any given word (Miller et al., 1990).

In contrast to polysemy, synonymy arises from varying word forms having similar meanings; exchanging such words does not alter so-called truth values. Changing 'plank' to 'board', though, retains truth

value, albeit in the specific context of carpentry. Further, substituting one POS for another would change the meaning of a sentence; that verbs, say, cannot be synonyms of nouns further justifies considering those two POS as discrete networks (Miller et al., 1990).

In practice, WordNet represents synonymy by means of synonym sets, or synsets, that enclose words in braces. By that means, synsets {*board, plank*} and {*board, committee*} depict alternate interpretations of 'board' (Miller et al., 1990). As semantic networks, though, synsets might further be depicted as nodes connected by lines, as Figure 5.2 shows for detail from WordNet's semantic network of nouns. Individual synsets appear as ellipses, which are connected by arrows that link related synsets; in that way, pointers between synsets result in a highly interconnected network (Miller, 1990). Here, then, is detail from that semantic network, which depicts human body parts and inter-relationships:

Figure 5.2: Detail of WordNet's semantic network of nouns, from Miller (1990)

Solid arrows from Figure 5.2 act as pointers to hyponyms, which express more specific forms of any given concept; for example, 'brother' is a particular type of 'relative'. Conversely, 'relative' is a hypernym of both 'brother' and 'sister'. Meronyms, on the other hand, reflect part-whole relations. For example, the meronym 'relative' is part of a 'family', which makes 'family' a holonym of 'relative'. The final semantic relationship, antonymy, addresses opposing concepts such as 'brother' and 'sister'. In the absence of such relationships, though, synsets hold a short gloss in parentheses; in that way, a further sense of 'board' appears in the synset {board, (a person's meals, provided regularly for money)} (Miller et al., 1990).

Note, though, that pointers from Figure 5.2 depicting hyponyms might be double-headed to reflect a reflexive relationship with hypernyms, as might meronyms further dictate holonyms. In addition, arrows from 'bone' point to both 'arm' and 'leg'; although such sideways links are forbidden in mind mapping, the semantic networks underpinning WordNet are far more interconnected. Aspects of WordNet's network, though, will be reflected in records based on intensional knowledge from GRiST mind maps. In turn, that intensional knowledge will enrich extensional knowledge already available from hierarchical relationships in those mind maps. That will depend on WordNet's coverage of word usage, addressed next.

The Coverage of WordNet

WordNet offers wide coverage of the four POS that it does consider. In illustration, table 5.1 presents statistics from references cited in this overview. The column headed Count gives the number of words held for each POS covered from the preceding column. The column headed Synsets indicates the corresponding number of synsets, while the last column reveals the source of those statistics[1]:

POS	Count	Synsets	Source
All POS	95,600	70,100	Miller et al. (1990)
Nouns	57,000	48,800	Miller (1990)
Adjectives	19,500	10,000	Fellbaum et al. (1990)
Verbs	21,000	8,400	Fellbaum (1990)
Adverbs	-	-	Miller et al. (1990)
Actual Totals	97,500	67,200	

Table 5.1: Statistics provided by the WordNet team.

Note, though, the lack of statistics for adverbs in Table 5.1, which Miller et al. (1990) admit but do not explain. Another point to heed is that specific POS have fewer synsets than words. That is taken as due to synsets that contain several variations on a particular word, for example, plurals of nouns. Verbs show the greatest difference, likely due to holding infinitives along with conjugated forms.

Similar reasoning might explain why overall published figures from first row of Table 5.1 disagree with the totals in the last row, obtained by summing those for specific POS. That might reflect the evolving content of WordNet, or could be due to polysemy, whereby different senses of a particular word might be counted more than once. For example, nouns sometimes act as modifiers (Fellbaum et al., 1990). Such ambiguity is but one of the drawbacks of WordNet to be discussed next.

[1]References from The WordNet team at Princeton University (2005) appear together as 'Five Papers about WordNet'.

5.3 Challenges Raised by WordNet

Although stemming identifies nodes that express morphological variants of words, determining POS might uncover additional knowledge in GRiST mind maps. Nouns, for example, suggest possible subjects and objects for phrases, while verbs are actions performed by or on those nouns. Such an analysis would establish relationships between atomic concepts within any given node, and between separate nodes. That would turn to advantage the relative verbosity of those nodes in comparison with the norm proposed by Buzan (1974, 1996, 2003).

Nodes such as `[abusing substances]` might, in that way, reveal to machines that something called 'substances' is the object of the action 'abusing'. In a similar way, the node `[abuse to client]` draws together two concepts, 'client' and 'abuse', that are both nouns. While the stem 'abus' from Chapter 2 relates morphological variants from those nodes, machines might further differentiate between words as actions and as concrete objects. That process is hampered, though, by ambiguity within WordNet that raise difficulties in determining any most likely POS for words.

Further, WordNet covers just content words, choosing to omit stop words. One particular type of stop word though, prepositions, hint at what might constitute a subject of any action, and what an object. For example, the preposition 'to' in `[abuse to client]`, though, shows 'client' to be the object of that phrase. By means of the word 'to', machines might render the equivalent phrase 'persons unknown abuse the client', where 'abuse' has been transformed into a verb. In that way, subjects, actions and objects of phrases might be held in a standard form by intensional knowledge. That absence of stop words, though, will be addressed shortly; before that comes the problem of ambiguity in WordNet.

Ambiguity in WordNet

GRiST mind maps comprise a modest corpus of 4242 unique words, of which 4112 are content words. Table 5.2 presents statistics gathered about those content words from WordNet, for unambiguous to the left, and ambiguous ones to the right. For each group, a column lists combinations of POS reported by WordNet, the total words so classified, n_w , and those totals as a percentage the 4112 content words:

POS Combinations	n_w	%	POS Combinations	n_w	%
Adverb only	188	4.6	Adverb + other	17	0.4
Adjective only	447	10.9	Not in WordNet	184	4.5
Verb only	507	12.3	Adjective + other	834	20.3
Noun only	1029	25.0	Noun + verb	906	22.0
Subtotals (of 4112)	2171	52.8	-	1941	47.2

Table 5.2: WordNet statistics for content words from GRiST mind maps

Left-hand subtotals from the last row of Table 5.2 reveal that around half of the words researched were reliably categorised, while remaining words had at least two possible POS. Most noticeable is the large number of words appearing both as nouns and as verbs. Of further note is the number of adjectives that could be taken for various other POS. Further, 4.5% of content words from GRiST failed to find an entry in WordNet; Chapter 3 took those as novel words rather than as spelling errors.

Percentages for words unknown to WordNet, or that might be construed as nouns or verbs, are reproduced next as a pie chart in Figure 5.3. Although many words are definitely nouns or verbs, a considerable proportion might be either, and are ambiguous:

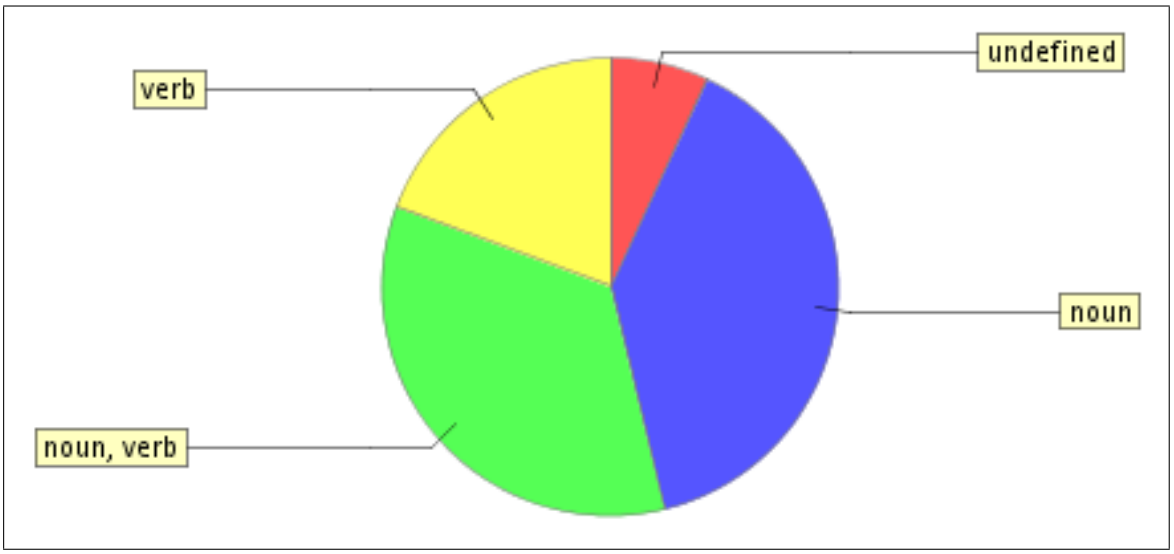

Figure 5.3: WordNet POS statistics I for words in GRiST mind maps.

Figure 5.3 re-emphasises the 4.47% of words from GRiST mind maps that were unknown to WordNet. The main feature of that graph, though, is the degree of confusion between nouns and verbs. To complement that chart, Figure 5.4 presents statistics for combinations involving adjectives and adverbs:

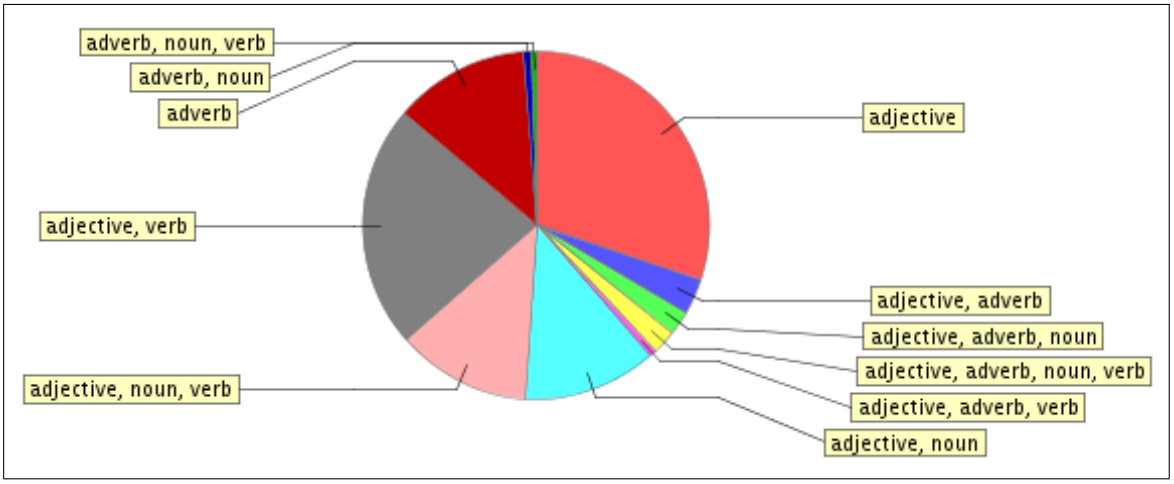

Figure 5.4: WordNet POS statistics II for words in GRiST mind maps.

The lack of distinction between adjectives and adverbs in Figure 5.4 is of no consequence: those POS will be treated collectively as modifiers. Of greater impact is that modifiers may be confused with nouns and verbs. In extreme cases, words could be used as any of the four POS supported by WordNet. The overlap between POS demonstrated by Figures 5.3 and 5.4 emphasises the need to resolve ambiguity.

Extracting reliable knowledge from GRiST mind maps demands finding intended meanings for words used by mental health experts. In fact, WordNet itself helps to a degree by means of a measure called familiarity, which reflects frequency of use. Indeed, WordNet employs polysemy - the number of senses for a given word - because polysemy and word frequency are correlated: more frequent words tend to have more distinct meanings that, all the same, are easily read, understood, and recalled. In that way, noun hypernyms were refined by removing less popular words. Take, for example, the hierarchy for 'bronco'; the words 'ungulate' and 'cordate' were removed due to having just one sense, while the more familiar words 'horse' and 'pony' were retained (Beckwith, Miller, & Tengi, 1993).

Familiarity, then, will help to find likely interpretation for words from GRiST mind maps. For example, the word 'house' has twelve noun senses but just two as a verb; the more familiar noun proves appropriate most of the time (Beckwith et al., 1993). That approach, though, fails for words such as 'abuse', which has three senses as a noun and three as a verb. In such ambiguous cases, familiarity alone cannot determine a most likely intended meaning. Overcoming such ambiguity has been the aim of various studies that, in a similar way to spelling correction, consider the context in which words are used. One approach to accounting for context employs word n-grams, as is shown next.

Using N-grams to Reflect Context

The process of disambiguation, then, might improve by considering words' context. That means treating ambiguous words in relation to other words, rather than in isolation (Brants, 2003). In that respect, the term n-gram describes any group of words to be taken together, with unigrams, bigrams and trigrams respectively holding 1, 2 and 3 words (Basili, Marziali, Pazienza, & Velardi, 1996; Bekkerman & Allan, 2005). There now follow examples of work that captured context in terms of n-grams.

Bigrams have helped to determine appropriate interpretations of nouns. Doctors, nurses and lawyers, for example, are all professional people; lawyers, though, work in a different profession than do doctors and nurses. That is evident from the hypernym relationships in Figure 5.5, where the '@→' symbol introduced in Section 5.2 shows successive levels in a noun hierarchy, with relevant entries in bold type:

person	@→	professional	@→	**lawyer**		
	@→	professional	@→	health professional	@→	**doctor**
		PhD.	@→	**doctor**		
	@→	professional	@→	health professional	@→	**nurse**
		nanny	@→	**nurse**		

Figure 5.5: Hypernym relationships for 'lawyer', 'doctor' and 'nurse' (Resnik, 1995)

Figure 5.5 shows that doctors and nurses share the general concept of 'professional' with lawyers. Doctors and nurses, though, are more closely related via 'health professional'. Further ambiguity arose because 'doctor' could mean someone with a PhD, while 'nurse' might refer to a nanny.

Appropriate senses of 'doctor' and 'nurse', when appearing together, were determined by a measure called the most informative subsumer. That measure involved researching words from bigrams in WordNet, whose noun hierarchies denote successively more specific concepts. At some point, any two such hierarchies will diverge; until that point, words from a bigram share a common ancestry. The most informative subsumer, then, was the lowest-level common ancestor, which denoted the most specific parent concept of words held in any bigram.

In that way, the most informative subsumer between 'doctor' and 'nurse' was 'health professional', which appeared lower down the two hierarchies than 'professional' (Resnik, 1995). To clarify further, the words 'PhD.' and 'nanny' defined different concepts, and subsumed just 'doctor' *or* 'nurse'. Conversely, the less informative subsumer, 'person', appeared higher in the hierarchy than did 'health professional'. Using bigrams rather than individual words, then, helped to avoid less appropriate senses (Resnik, 1995).

Clustering as a Means of Overcoming Ambiguity

A further example of using bigrams examined text from 20 categories of news reports. Although that study did not specifically address ambiguity, it showed the importance of considering context. The aim was to identify words that characterised particular news categories. To that end, lists were compiled of words that appeared both discretely as unigrams, and together as bigrams. The distribution of those n-grams across the 20 news categories was analysed by a technique called distributional clustering (Bekkerman & Allan, 2005). Because the next chapter deals with clustering in detail, attention here focuses on the benefits of analysing bigrams in that way.

In fact, the input to distributional clustering comprised the frequencies for 20,000 bigrams. By drawing together bigrams having similar frequencies, resulting clusters reflected the distribution of bigrams across types of news reports. That, in turn, revealed bigrams comprised of words more likely to be found together than apart in any particular text category (Bekkerman & Allan, 2005).

Importantly, bigrams proved especially good at distinguishing between news categories, due to a lack of overlap between sets of bigrams from each category. Conversely, unigrams were less discriminating, being more likely to occur across categories. Further, bigrams reflected associated words such as 'human rights' and 'operating system', showing tight interconnection between bigram components. In contrast, a BOW approach would separate such words, even should they appear together in stable phrases. Pairs of nouns, for example, would be reduced to isolated concepts, and any special meaning lost (Bekkerman & Allan, 2005). That is the view taken here: emergent knowledge from word-pairs in GRiST mind maps is not evident when such words are taken separately.

Applying Trigrams to Resolving Ambiguity

An example of overcoming ambiguity by means of trigrams comes from work on the 500,000-word LD corpus. That study employed dependency relations such as noun-preposition-noun to specify POS for words in trigrams. An initial learning phase processed trigrams from just unambiguous WordNet results, yielding trigrams that reflected reliable patterns of POS around any given preposition. Subsequently, a testing phase compared ambiguous cases, which had earlier been ignored, against reliable trigrams from the learning phase. Distributional clustering of dependency relations arranged such trigrams into reliable patterns of POS co-occurring with any preposition. In that way, ambiguous POS were assigned the type from the equivalent position of such a reliable trigram (Basili et al., 1996). The best interpretation of any ambiguous word, then, arose from the context of POS for words from well-founded trigrams.

Issues Arising from Studies of N-grams

The approaches just described, then, used n-grams to reflect the context of ambiguous words. In particular, Basili et al. (1996) applied trigrams to resolving ambiguity; unambiguous trigram components helped

to determine POS for ambiguous parts of further trigrams. In that sense, trigrams acted as templates that indicated dependable patterns of word usage, suggesting POS for ambiguous words from unambiguous ones used in similar contexts. A like approach will shortly be introduced for handling ambiguous words from GRiST mind maps. For now, though, attention turns to issues with studies of n-grams.

The first issue concerns an emphasis on nouns, and on subsumption relationships between them. Indeed, Basili et al. (1996) specifically restricted attention to WordNet's noun hypernyms that represent subsumption. Take further the word 'nurse' from work by Resnik (1995). In fact, 'nurse' is more likely to be a verb: WordNet reports five verb senses, compared with just two as a noun. Despite that higher familiarity, 'nurse' was not treated as a verb. Indeed, neither was 'doctor'; that word, though, is less well differentiated in WordNet, having four verb and three noun senses. While Basili et al. (1996) considered 'nurse' and 'doctor' as nouns when they appeared together, both might just as well have been verbs..

A second issue from studies of n-grams concerns degrees of human intervention. The most informative subsumers of Resnik (1995), for example, relied on human judges that chose most likely WordNet senses for over a hundred ambiguous words. Resulting confidence values between 0 and 4 identified instances having low confidence levels of 1 or 0; such cases were excluded from ensuing analyses. Human intervention was further evident in the approach of Basili et al. (1996) that, although said to be unsupervised, in fact entailed training the system on manually compiled trigrams. Further, both of those studies addressed just words having entries in WordNet. Having showed in Table 5.2 that WordNet lacked 184 content words from GRiST mind maps, such an approach would risk missing important concepts.

Ambiguity, then, might be overcome by applying distributional clustering to word n-grams that contain prepositions. Chapter 6 will deal with the particular clustering algorithm proposed for resolving ambiguity in GRiST mind maps; attention now, though, turns to a further shortcoming of WordNet: a lack of the very stop words, such as prepositions, that are seen as vital to that effort.

WordNet's Lack of Stop Words

While constituting an important source of knowledge about the form and meaning of words, WordNet coverage is limited to just content words: nouns, verbs, adjectives and adverbs. Various other important POS are missing from WordNet. Indeed, function words, often called stop words, are commonly ignored, or removed from texts under investigation (Yang & Pedersen, 1997; Brants, 2003). Even Yang and Pedersen (1997), who claimed to examine all degrees of feature selection, give the exception of removing stop words. That, though, risks discarding useful information (Bekkerman & Allan, 2005).

That view, though, contrasts sharply with those encountered in approaches to spelling correction: words that appeared in pre-compiled lists of stop words were removed from text, on the basis that they carry no

information (Aas & Eikvil, 1999; Dolamic & Savoy, 2008). While that might be justified in the context of refining non-words, it is not the case concerning ambiguity. A further reason for ignoring stop words is to expedite IR; response times in such applications generally improve should stop words be removed prior to processing documents (Yang & Pedersen, 1997; Brants, 2003). Performance reasons aside, semantic networks comprise concepts that denote 'things'; DL, in particular, stress subsumption relationships between unary predicates, or classes, suggested by nouns (Horrocks, Sattler, & Tobies, 2000; Nardi & Brachman, 2002; Tsarkov et al., 2007). That arises from the importance placed on hypernym-based subsumption relationships, which are a property of nouns alone (Miller, 1990).

The view of Bekkerman and Allan (2005) is heartily endorsed, here: ignoring stop words discards useful information. While atomic concepts might form class hierarchies, stop words such as prepositions reflect relationships between those concepts. In that respect, neglecting stop words would deny the richness of GRiST mind maps, the combination version of which, after all, Buckingham and Adams (2006) saw as an optimal representation of knowledge about mental health risks. To illustrate words that WordNet would ignore, Table 5.3 presents stop words taken from those mind maps:

POS	Stop Words from GRiST mind maps
Articles	a, an, the
Prepositions	about, at, by, for, in, like, near, now, of, on, over, to, with
Pronouns	anybody, he, her, himself, me, she, some, them, this, us, what, you

Table 5.3: Non-WordNet parts of speech from GRiST mind maps.

Prepositions are the most important POS from Table 5.3, in that they govern relationships between concrete concepts. Pronouns and articles might further represent such specific classes; indeed, implicit references to 'things' will shortly elevate certain stop words to that status. One particular type of stop word from Table 5.3, prepositions, will play important roles in exposing relationships suggested by mind map nodes. Before that, though, come examples of studies that recognised the importance of stop words.

Recognising the Importance of Stop Words

WordNet, then, distinguishes between content words, comprising nouns, verbs, adjectives and adverbs, and stop words, which include prepositions. Automated analyses of text, though, commonly ignore stop words as a source of knowledge. A notable exception comes from work on Recognising Textual Entailment (RTE). Two similar sentences with opposing meanings demonstrate the difficulty that machines have with RTE: 'Slow down so that you do not hit riders' compared with 'Do not slow down so that you hit riders'. The same words express very different outcomes associated with slowing down. The key word for expressing entailment in those sentences was the preposition 'so' (Blake, 2007).

Difficulty with RTE was overcome by having machines identify roles for words in sentences. Such knowledge was couched in the dependency grammar of Figure 5.6, which reflects the wrapping of the German Reichstag building in aluminised fabric by Christo, the artist. Labelled arrows in that diagram describe dependencies between words:

Figure 5.6: An example of dependency-grammar, after Blake (2007)

The left side of Figure 5.6 shows a dependency labelled 'det' for 'determinant', that is, the relationship between 'the' and 'Reichstag'. More important dependencies exist between 'by' and 'the Bulgarian artist Christo', and between 'in' and '1995'. The former denotes that Christo did the wrapping, while the latter indicates when (Blake, 2007). The core idea from Figure 5.6, then, was summarised as in Figure 5.7:

Figure 5.7: Derived dependencies for the sentence from Figure 5.6, after Blake (2007)

Figure 5.7 shows dependencies labelled 'nsubj' between 'Christo' and 'wraps', and 'dobj' between 'wraps' and 'Reichstag', respectively indicating a subject and an object for the action 'wraps'. Such combinations of subject, action, and object constitute triples, which were the primary means of determining RTE. The subject and object of any action inferred what that action entailed (Blake, 2007).

In addition to content words, then, prepositions in triples well indicated dependencies in RTE. In the absence of explicit subjects or objects, those roles were implied by a process called collapsing preposition paths. The preposition 'by' was removed, and the passive verb 'was wrapped' replaced by the active form, 'wraps' (Blake, 2007).

A further study that considered prepositions employed a technique called chunking, which reduced sentences to more manageable phrases. Square brackets show how the sentence 'The red car is parked on the sidewalk' yielded the chunks [The red car] [is parked] [on the sidewalk]. Prepositions offered important clues to identifying chunks in reference texts. In that way, the Swedish Parole corpus provided 16 million chunks, while the KTH News Corpus gave 10 million. Those chunks constituted n-grams of varying size, including bigrams and trigrams. On subsequently comparing chunks from novel texts with those reference n-grams, rare chunk sequences were reported as possible errors (Sjöbergh, 2005).

Points of Note from Reviewed Studies

The above studies demonstrated the utility of stop words, especially prepositions, in resolving ambiguity. Those works re-state the importance of context, but for deriving relationships between words rather than for disambiguating them. Another point of note concerns so-called modifying nouns, which were important in RTE (Blake, 2007). That supported the idea of treating one component of any noun pair as an adjective. Further, triples of the type *subject-action-object* offered a normalised form for representing relationships.

The issue of human intervention, though, arises once again from that study of chunking, which was claimed to aid unsupervised processing. In fact, human coders adjudicated decisions about rare chunk sequences reported as potential errors. All such reports were checked manually to see if they were genuine errors, or false alarms (Sjöbergh, 2005). On the other hand, work on RTE by Blake (2007) showed an exemplary, albeit rare, level of automation. Prepositions, then, prove useful should they be considered at all; such an approach of applying stop words to resolving ambiguity in GRiST mind maps is given next.

5.4 Stop Words and Ambiguity in GRiST Mind Maps

Section 5.3 demonstrated two shortcomings of WordNet that must be addressed. The first difficulty concerned assigning POS to ambiguous words. Ambiguity arose when a particular word had several senses, say as a noun or as a verb. The second drawback of using WordNet, and of approaches to machine learning reviewed earlier, was a lack of attention to stop words. Overcoming one of those challenges will help to resolve the other, as described next.

Completely unambiguous words have just one POS, and a single sense for that POS. In other words,

unambiguous words are monosemous (Mihalcea & Moldovan, 2000). That seems excessive for the earlier example of 'house', where the noun could be used in many more ways than the verb. Considering 'house' as a noun should give the correct interpretation, in most cases. For that reason, a particular POS will be treated as predominant when it outranks the next most likely type by a given number of senses. That, though, does not help when possible POS have similar numbers of senses. By itself, WordNet can offer no clues as to the most likely meaning of such words.

The reviewed approaches to resolving ambiguity stressed the importance of context. Ambiguous words were considered in relation to neighbouring words, rather than in isolation. For example, medical senses of 'doctor' and 'nurse' were taken when those words occurred together. By themselves, it was unclear whether respective possibilities of 'PhD. holder' or 'nanny' were appropriate (Resnik, 1995). N-grams of two or more unambiguous POS encoded reliable patterns of usage.

In that way, several n-grams arose from an unambiguous appearing next to an ambiguous word. Unambiguous POS were identical across such a set, while individual n-grams held a specific candidate POS for the ambiguous word. Reliable, unambiguous n-grams indicated likely patterns of usage from which to choose the best candidate in any such set. Novel n-grams that had no reliable counterpart were discarded. Of the remaining n-grams, the most frequent one gave the best interpretation. Accordingly, trigrams will be used here to resolve ambiguity in GRiST mind maps. Trigrams formed from unambiguous words will reveal reliable patterns of meta-types around prepositions. Subsequently, reliable trigrams act as templates for resolving trigrams that have ambiguous components. Trigrams, though, are not composed directly from POS reported by WordNet, as is shown next.

Deriving Meta-Types from POS

The proposed approach, then, uses trigrams based on prepositions. The remaining two components reflect POS for words found immediately before or after any preposition. Using POS reported by WordNet yields trigrams such as N-P-N, which Basili et al. (1996) used to represent a preposition between two nouns.

In fact, POS will not be used directly. Instead, trigrams will reflect what roles words might take in a phrase. In that respect, nouns represent things, while verbs express actions. Adjectives and adverbs are collectively treated as modifiers, as suggested by Miller et al. (1990). Trigrams will reflect prepositions surrounded by 't' for things, 'a' for actions, or 'm' for modifiers. Those letters will be called meta-types, to reflect how they come from actual types reported by WordNet.

Many words have implicit meta-types: articles and pronouns, for example, denote things. In contrast, conjunctions and prepositions will be taken as modifiers, because they do not depict firm concepts[1]. That

[1]Prepositions are treated as modifiers should they comprise either of the two associated words in a trigram.

approach permits more words into the analysis than if actual POS were considered. In that way, patterns in meta-type usage will emerge, rather than affinity between specific words such as 'doctor' and 'nurse'.

A Stop Word Adjunct to WordNet

WordNet's lack of stop-words is often overcome by using a 'stop list'. Such lists are used to avoid what are seen as irrelevant words (Murtagh, Mothe, & Englmeier, 2007). In contrast, such a list from Pedersen (2001) will be used to build an adjunct to WordNet. Inspecting that list yields the corollary of a synset that reports POS for stop words. No pointers exist in such pseudo-synsets, just an indication of a main category, with an optional sub-category. All the same, that WordNet adjunct allows stop words to return synsets that resemble those for content words supplied by WordNet.

Table 5.4 lists just the main categories of stop words. Sub-categories of stop words were for information only, and are omitted here; further, the main category has been assigned an appropriate meta-type, M-T:

Category	M-T	Stop Words
article	t	a, an, the
pronoun		him, her, I, it, somebody, something, this, that, what
conjunction	m	and, after, although, before, but, or, so, unless, when, while
preposition		as, at, before, by, down, during, from, like, since, to, with

Table 5.4: Examples from a stop word adjunct to WordNet, after Pedersen (2001)

Synsets based on Table 5.4 serve two purposes. The most important of those is to identify prepositions, from which to build trigrams. In addition, those synsets allow stop words to contribute meta-types for other trigram components.

Clustering

The method conceived by Bekkerman and Allan (2005) was an alternative to BOW approaches that rely on algorithms trained by supervised learning. A particular advantage of clustering was that it reduced the need for human supervision, which was restricted to specifying the number of clusters required. A further advantage was in overcoming 'statistical sparseness' in the underlying data, namely, the low ratio of bigrams to unigrams (Bekkerman & Allan, 2005). Indeed, GRiST mind maps might be seen as sparse in comparison with the relatively large LD corpus. That commends clustering as a technique for analysing n-grams from those mind maps, as the next section will show.

The problem remains of determining reliable, unambiguous patterns of words around prepositions. The earlier review of resolving ambiguity showed a way of doing that for bigrams. A technique called distributional clustering revealed groups of semantically related n-grams. That, in turn, reflected the distribution of n-grams over particular categories of text (Bekkerman & Allan, 2005). Although trigrams

will be used, clustering suggests a way of detecting patterns of word usage. Specifically, clustering will show associations between particular prepositions and specific flanking meta-types.

The clustering algorithm used here is called Correspondence Analysis (CA). That technique has been applied in various studies of word usage in text. The primary output from CA is a graph, which shows clusters of related points. From such clusters, researchers deduce patterns of words that characterise genres of text. Interpreting CA graphs was up to those researchers. Humans had to decide what particular words were associated with different genres. A review of such studies follows shortly. For now, it is the manual nature of CA that is important.

Instead of involving humans, machines will be made to interpret CA graphs by means of retaining CA results after completion. Raw results that would normally be discarded will be available for further analysis by machines. In that way, parameters derived from an initial CA allow a more detailed run to ensue, again automatically. From that subsequent run, machines will reveal patterns in how meta-types interact with prepositions. Any reliable trigrams that result will help to resolve ambiguous cases; when found next to a preposition, ambiguous words will take the POS indicated by predominant meta-types for that preposition.

Clusters of prepositions at the edges of graphs raise another aspect of automating CA. So-called outlier clusters may be removed, to improve models of any underlying data (Bullen, Cornford, & Nabney, 2003). Outliers are indeed removed, but only after assessing their contribution to any graph. Instead of seeing outliers as disruptive, they are taken here as very strong associations between prepositions and meta-types. Repeatedly identifying and removing outliers will yield a more graded analysis, which will allow machines to report a degree of certainty about any automated decision.

Relationships between Words that Surround Prepositions

Studies discussed in Section 5.3 revealed a focus on nouns. That reflected a need for tangible concepts when building semantic networks. Stop words represent less concrete ideas, and were largely removed from text prior to analysis. Chapter 2 dismissed that BOW approach because it destroys relationships between words. In contrast, such relationships will be treated as important knowledge in GRiST mind maps.

Rather than seeking to create a concept hierarchy, this thesis emphasises the roles that words fulfil in phrases. In that respect, nouns represent things, verbs describe actions, and adjectives and adverbs act as modifiers (Miller et al., 1990). Deriving meta-types from words allows roles to be assigned, say, to things as the subjects or objects of actions. Assigning roles to trigram components yields triples, of the type subject-action-object; in that way, triples might describe relationships in GRiST mind maps.

Additional metadata attach modifiers to the two other meta-types.

CA will associate certain propositions with specific meta-types. The arrangement of those meta-types suggests ways of rearranging them. Trigrams of the type A-P-N, for action-proposition-thing, might be rearranged as A-N, as Blake (2007) did for Christo. N-P-N trigrams, for propositions surrounded by things, might be rewritten as just the things, in reverse order. That stems from the way nouns sometimes act as modifiers (Fellbaum et al., 1990). The final form would be M-T, for a modifier followed by a thing.

Further novel clauses might arise from mutually opposed propositions, indicated by CA. For machines, that ability arises from analysing the axes of CA graphs; the order of meta-types established for one proposition might be reversed for opposing proposition. That suggests that trigrams of the form X-P-Y might be rearranged as Y-P-X, where X and Y represent any permitted meta-type. Rearranging words in that way is by no means assured of success: results might well be inappropriate. To check the validity of such transformations, examples of actual usage are sought in GRiST mind maps. Any single instance would not, by itself, constitute enough evidence for such novel clauses. Backed by results from CA, though, such sole examples are more indicative of a general rule.

5.5 Chapter Summary

The first part of this chapter gave an overview of WordNet, one of the principle tools for this thesis. After describing the psycholinguistic origins of that lexical database, two challenges were noted concerning its use. The first of those drawbacks arose from ambiguity in WordNet, while the second involved a lack of attention to stop words. Of the various approaches used to resolve ambiguity, distributional clustering of trigrams proved the most applicable to this thesis. That use of clustering was shown be applied here in the form of CA. After an overview that revealed CA's origin in classical mechanics, various studies were introduced to illustrate how it has been used to analyse text.

Following that, a new approach based on stop words, which involves deriving meta-Types from POS reported by WordNet. A stop word adjunct to WordNet was introduced to facilitate that process. Performing CA on matrices derived from those meta-types reveals patterns of word usage associated with prepositions. Clusters of corresponding prepositions and meta-types will help to resolve ambiguity, and derive relationships between words from GRiST mind maps. All stages of CA were shown amenable to machines, in a proposal for automating CA. Having suggested clustering as a means of compensating for WordNet's shortcomings, attention turns next to the specific tool proposed here: Correspondence Analysis.

6

Clustering Concepts from GRiST Mind Maps

6.1 Introduction

Previously, Chapter 5 revealed a technique called distributional clustering used by Bekkerman and Allan (2005) to analyse bigrams from news reports. Accordingly, this chapter introduces Correspondence Analysis (CA) as the specific clustering technique employed in this thesis. The chapter starts with an overview of CA, and the means by which it explains patterns in underlying data. Following that comes an overview of studies that have applied CA to researching text, after which CA will be shown as useful for analysing trigrams arising from researching words in WordNet.

In respect of full automation, machines will face various challenges, such as building input for CA, and then running it. More demanding challenges, though, will arise in interpreting unaided any resulting graphs.

6.2 An Overview of Correspondence Analysis (CA)

CA is a form of multivariate data analysis amenable to both quantitative and qualitative data. Applications of CA range from analysing the X-ray emissions of astronomical bodies, to stock market movements in economics. Whatever the domain of study, CA identifies factors that explain patterns in underlying data. Input to CA comprises the frequencies of occurrence for variables within certain categories from a domain. Those counts reflect a specific variable's presence in, or absence from, particular categories; from such categories arises a cross-tabulation between a set of attributes and set of observations made on those attributes. The main output from CA consists of graphs that aid humans in visualising any emerging patterns (Murtagh, 2005; Murtagh, Ganz, & McKie, 2008).

Given a matrix of observations across various categories, then, CA derives factors that explain variation between those categories. Subsequently, CA determines interactions between factors, from which clusters of related results arise; such clusters highlight trends in the underlying input data. Borrowing from classical mechanics, CA endows each point on the resulting graph with a proportionate mass. Subsequently, the inertia of any cluster indicates the strength of relationships between points on a graph. Successive factors F1...Fn explain ever smaller proportions of the total inertia. Resulting graphs describe a complex web of relationships between variables and categories from any domain of interest (Murtagh, 2005, 2008; Murtagh et al., 2007). Those basic principles underlying CA will now be treated in more detail, as described by Benzécri (1992).

The Input Matrix for CA

Input to CA, then, consists of a set of observations and a set of attributes, respectively called sets I and J. Set I comprises rows from the input matrix, while set J represents columns. Each column in the matrix represents a particular variable from any domain of study, while rows reflect measurements such as occurrences of those variables over various categories. Together, sets I and J yield a rectangular table of positive numbers. The first step in CA, then, calculates various totals from sets I and J. The index i specifies particular rows in the set I, while index j depicts columns from set J; intersecting values of i and j identify an individual cell $k(i, j)$ in the matrix body. From that follow three important totals:

$$\begin{aligned} \text{the total } k_i \text{ for row } i: \quad & k_i = \Sigma \{k(i,j) \mid j \in \mathrm{J}\} \\ \text{the total } k_j \text{ for column } j: \quad & k_j = \Sigma \{k(i,j) \mid i \in \mathrm{I}\} \\ \text{the grand total } k: \quad & k = \Sigma \{k(i,j) \mid i \in \mathrm{I};\, j \in \mathrm{J}\}. \end{aligned}$$

In deriving the first totals, k_i for each row, CA holds constant the row index i while summing successive cell values by varying j for as many columns as exist in J. In a similar way, the second total, k_j, sums cells under a particular column by adjusting i for any given value of j. The third expression, in contrast, sums all cells in the matrix by varying both i and j. Those three totals are annotated in Figure 6.1, which shows a matrix from the first phase of CA. Note, though, the extra row and extra column at the margins of the raw data table. A cell at index j of the marginal row holds the column total $k(j)$, while the marginal column holds the total $k(i)$ for row i. The grand total k appears in the bottom-right cell:

Figure 6.1: An extended input table for CA (Benzécri, 1992).

Totals from Figure 6.1 are important to the next stage of CA, where they contribute towards profiles that normalise values from the initial matrix.

Profiles

Correspondence between sets I and J might not be obvious from raw observations: cells having widely varying values yield marginal totals that obscure underlying trends. Normalising those totals reveals rows or columns having similar proportions, regardless of absolute values. The first step is to divide cells of any row i by the marginal total $k(i)$. A particular cell $k(i,j)$ receives that weighted value f_j^i as follows:

$$f_j^i = k(i,j) \ / \ k(i).$$

The weighted value f_j^i, then, reflects a proportion of the row total $k(i)$. Because of that, weighted values for any row or column add up to 1. Subsequently, profiles arise from such normalised values. The profile for row i is the set of f_j^i for all elements from J, the table columns. In other words, that is the set of all f_j^i for j taken over J. Accordingly, the profile f_J^i for row i is expressed as:

$$f_J^i = \{ f_j^i \mid j \epsilon J \}.$$

In a similar way, the profile f_j^I for an entire column j comprises weighted values over all rows:

$$f_j^I = \{ f_j^i \mid i \epsilon I \}.$$

Sets I and J, then, are depicted by respective profiles f_j^I and f_J^i. Identical profiles, though, offer just a baseline for comparison with later analyses; rather, it is dissimilar profiles, where $f_j^I \neq f_J^i$, that reflect particular correspondences between elements of sets I and J.

Figure 6.2 demonstrates how row profiles, for example, help to elucidate patterns within raw observations. That matrix comprises three sets of observations for two variables, giving a matrix of three rows by two columns. The additional marginal column shows totals for each row. Weighted values for each of those columns appear to the right-hand side of that figure:

Figure 6.2: Deriving row profiles for a CA matrix (Benzécri, 1992).

Note that the first and last rows from Figure 6.2 have identical profiles of $\{0.65, 0.35\}$, despite differences in actual values held as $k(i,1)$ and $k(i,2)$. That is important to the next topic of spatial representation.

Spatial Representation of Sets

Graphs arising from CA depict correspondences between profiles, which in turn reflect any interaction between attributes from sets I and J. Any CA graph shows sets of points as clouds in a multidimensional space. CA permits any number of dimensions, the actual number depending on the number of columns, that is, on the cardinality of set J. For a table of two columns, that space is a 2-dimensional plane. Three columns give a 3-dimensional volume. In generic terms, points from a table of n columns will always lie within a n-dimensional space.

Within any CA space, coordinates for individual points reflect values from profiles in set I or set J. Single profiles comprise a tuple of values that, together, specify a point's coordinates. Rows that have identical profiles designate the same point on a graph. Increasing differences between profiles yield correspondingly distant points on the graph. In that way, CA produces separate clouds of points, which give a spatial representation of diversity between rows and columns from a data table.

Points drawn from profile values, though, occupy just a portion of any CA space; in fact, points fit within a simplex, that is, a n-dimensional triangle. For a matrix of two columns, the space is a triangular area on a plane. In three dimensions, the shape is a tetrahedron. More than three dimensions, though, make it hard for humans to visualise the CA space, due to difficulty in imagining four or more axes at mutual right angles. CA eases comprehension by producing a faithful, though simplified, representation in a space of lower dimensions. That raises the issue of identifying the more important dimensions, which is done by considering points as having mass.

Imbuing Points with Mass

Rather than treating points as equally important, CA considers some points as more influential than others. To that end, CA assigns according masses to points within a cloud. Imbuing points with mass stems from applying classical mechanics to points on a graph, which means that points mutually attract or repel in proportion to their masses. Accordingly, any cloud of points might be reduced to a spatial mean, depicted as a single point that reflects that cloud's overall mass. To demonstrate, consider a cloud of n points on a graph. The x-coordinates of those points comprise the numbers $x^i \ldots x^n$. For the cloud as a whole, the mean of those x axis components is expressed as $\overline{x}$:

$$\overline{x} = (x^1 + x^2 + \ldots + x^n)/n.$$

Treating each axis of the CA space in that way yields coordinates for the spatial mean[1]. Note, though, that averaging coordinates creates points of equal relevance within a cloud. In contrast, assigning masses to components along each axis accounts for points of varying importance. To illustrate, the above x-axis

[1] In that way, coordinates for a spatial mean within a 2-dimensional space might be expressed as $\{\overline{x}, \overline{y}\}$.

components $\overline{x}$ of points $1 \ldots n$ are weighted by masses $m_1 \ldots m_n$ to give a weighted mean $\overline{x_w}$:

$$\overline{x_w} = (m_1 x^1 + m_2 x^2 + \ldots + m_n x^n)/(m_1 + m_2 + \ldots + m_n).$$

In short, the sum of weighted components is divided by the total mass. Treating all dimensions in the CA space in that way yields coordinates for an alternative spatial mean that constitutes any cloud's centre of gravity, which acts as if the total mass were concentrated there[2]. More precisely, the term barycentre emphasises how that centre of gravity arises from unequal masses.

In practice, CA derives the barycentre by means of row profiles, and of the CA matrix in Figure 6.2. Values from individual profiles are weighted by masses calculated from the matrix[3]. Consider the mass of a point i from set I, the matrix rows. That mass depends on the marginal total $k(i)$ from the row for that point. In addition, all masses depend on the grand total k from the matrix. In those terms, the mass f_i for row i arises from dividing the row total by the grand total:

$$f_i = k(i)/k.$$

In a similar way, the mass of any column j from set J is:

$$f_j = k(j)/k.$$

Masses shown as f_i and f_j are used as the masses $m_1 \ldots m_n$ in the earlier calculation of $\overline{x_w}$. Values from profile i, which specify coordinates along various axes, have associated masses f_i. Within a cloud, the resulting point for profile i plays a role proportional to its mass. In a similar way, the mass f_j indicates the importance of the point for column profile j.

[2]For example, an original spatial mean $\{\overline{x}, \overline{y}\}$ becomes the weighted mean $\{\overline{x_w}, \overline{y_w}\}$.
[3]Note that unweighted values from the original matrix reflect the relative importance of rows.

Forces between Masses

From mass stems a mutual force between points on a graph. As an example, take the points for row i and for column j in a matrix. Those points depict the profiles f_J^i and f_j^I respectively. Each value within those tuples specifies a position on a particular axis. Along those axes, the row and column components either attract or repel one another.

In order to determine the direction of any force on a particular axis, CA considers the value $k(i,j)$ at the intersection of row i and column j. That value $k(i,j)$ is compared with the product of respective row and column masses f_i and f_j, which are multiplied by the grand total k from the matrix. Points repel one another should the cell $k(i,j)$ be smaller than that combined product; conversely, points are mutually attracted should $k(i,j)$ exceed that product. The direction of any interaction, then, is summarised as:

$$k(i,j) = kf_if_j \quad : \quad \text{no force acts between } i \text{ and } j;$$

$$k(i,j) > kf_if_j \quad : \quad \text{components } i \text{ and } j \text{ are mutually attractive;}$$

$$k(i,j) < kf_if_j \quad : \quad \text{components } i \text{ and } j \text{ are mutually repulsive.}$$

The product of masses, shown as f_if_j, accounts for absolute differences in mass. That product will be unchanged as long as differences in the row mass f_i are balanced by changes in the column mass f_j.

To the right of the equals sign in those expressions, the grand total k is multiplied by the product of masses, f_if_j. Note that dividing both sides of those expressions by k shows that $k(i,j)/k = f_if_j$. In other words, the product of masses equals the cell's proportion of the grand total. That shows a lack of affinity between components i and j: no force acts between them. In such cases, masses merely reflect the raw observations, and have no effect.

A force arises when combined masses differ from any cell's proportion of the grand total. Should the product exceed that proportion, components i and j attract one another. In such cases, the product of masses exceeds what would arise from using standard masses; effectively, the raw observation $k(i,j)$ has such a standard mass. Extra mass arises when, together, marginal totals i and j notably contribute to the grand total, increasing attraction between components i and j. Low combined masses, on the other hand, reflect relatively small marginal totals for i and j; the product of masses is less than what would arise from using a standard mass. Relative reductions in mass lessen any attraction between such components, which act as if mutually repelled.

The actual force acting on any point, then, reflects combined interactions across n dimensions. In that way, CA groups together related points, while separating them from others. Resulting forces deform clouds, which will soon be shown to reflect factors in CA. In the absence of any force, components i and j

will not contribute to those factors. In addition to generating a force, though, imbuing points with mass permits allusion to a further property from mechanics: inertia, which is discussed next.

Inertia

Inertia reflects the shape of a cloud, that is, the dispersion of points around the barycentre. Inertia for any given point with respect to point P is shown as I_P. Further, any point of interest is shown as $M^i m_i$ to depict a point M^i having mass m_i. The inertia of that point with respect to P depends on the mass m_i and on the squared distance between P and M^i, shown as $d^2(P, M^i)$ in the following equation:

$$I_P(M^i, m_i) = m_i \, d^2(P, M^i).$$

The inertia of any point with relation to point P, then, is the product of that other point's mass, and the squared distance from P. Squaring the distance means that inertia increases sharply as points become more widely separated. Subsequently, the inertia of entire cloud N in relation to P arises from summing the inertias of points in the cloud, with respect to P. In that way, the overall inertia $I_P(N)$ becomes:

$$I_P(N) = I_P(M^1, m_1) \; + \; I_P(M^2, m_2) \; + \ldots + \; I_P(M^n, m_n).$$

Although the above works well for individual points, the inertia of the entire cloud N is best considered in respect of its barycentre. Specifically, the inertia $I_G(N)$ arises with respect to a point G, at the centre of gravity for cloud N. Rather than summing I_P across points in a cloud, $I_G(N)$ arises from adding together I_G for those points.

Cloud N, then, has inertia $I_G(N)$ with respect to the centre of gravity of that cloud. From that inertia, CA calculates the total variance of cloud N as $Var_{tot}(N)$, reflecting the degree of spread away from the barycentre. In other words, $Var_{tot}(N)$ reflects the dispersal of points in cloud N around its centre of gravity. CA obtains that measure by dividing inertia $I_G(N)$ by the mass of the cloud, m_{tot}:

$$Var_{tot}(N) = I_G(N)/m_{tot} \quad (\text{where } m_{tot} = m_1 + m_2 + \cdots + m_n).$$

In that way, the variance $Var_{tot}(N)$ measures mean squared distance from the centre of gravity, weighted by the mass of the cloud. Having derived variance, the end is in sight; all that remains is to derive the factors depicted by any CA graph.

Factorial Axes

The shape of any cloud, then, reflects distortions arising from forces between points; for that reason, the barycentre may be offset from the spatial mean. In addition, points around that barycentre may stretch further in particular directions. CA uses such deformities to project lines through any cloud, describing the principal axes of inertia known as factorial axes. Those axes are mutually perpendicular lines passing

through any cloud's centre of gravity. Inertia is at a maximum along the first axis, which connects the most widely dispersed points, and constitutes the principal factorial axis Δ_1.

Subsequently, axis Δ_2 arises at right angles to Δ_1 between the next most widely dispersed points. Ultimately, a system of axes arises on which to plot coordinates from profiles; the complete set of axes is called Δ_α, where α depicts the number of columns in the matrix. Successive values within profile f_J^i for row i dictate a position on the corresponding axis j from Δ_α.

Summary

Input to CA, then, consists of observations made on attributes taken from any domain of interest. Set I comprises rows from such a matrix, while set J represents columns; intersections between respective indices i and j identify individual cells. Three totals are calculated for rows, columns, and the matrix as a whole: respectively, k_i, k_j and k. From those totals arise profiles, which adjust raw observations to identify rows or columns having similar proportions. In addition, profiles yield coordinates for representing sets I and J spatially, on a graph.

Graph points are further imbued with masses proportionate to the contribution of k_i and k_j to the grand total k. In that way, points exert forces relative to their mass, in a direction that depends on the product of masses for corresponding rows and columns. Further, the inertia of any cloud reflects the overall mass of individual points, in addition to the distance between points and the cloud's centre of gravity. Inertia is lowest near any cluster's centre of gravity, increasing with distance away from that point. Consequently, widely spread points reflect a high degree of underlying variation. Having determined the shape of clouds in the CA space, factorial axes follow as lines between bulges on that space. Starting with the most pronounced, successive factors explain correspondingly less variation between sets I and J, the respective matrix rows and columns. That, then, is all one needs to know in order to understand CA, and to interpret any results (Benzécri, 1992).

Indeed, I would argue that to be more than one needs to know, and that an appreciation of profiles, along with masses and forces from secondary school Physics, is enough. In a similar way that psychologists might be unaware of the intricacies of the Analysis of Variance (ANOVA), yet use that tool widely, CA is seen here somewhat as a 'black box' whose machinations one trusts to greater mathematical minds. In that way, CA overcomes the problem that Buckingham et al. (2004) see in statistical tools demanding a strong numerate background, and provides what Murtagh (n.d.) describes as a multi-modal, multi-faceted analysis toolbox. Attention turns next to using that toolbox for analysing text.

6.3 Correspondence Analysis for Researching Text

Applying CA to discerning patterns in text is not a novel idea; indeed, it has been used to investigate literary style in various genres of written work. Bodies of text under scrutiny have ranged from the Christian Gospels to 19th Century fiction, in addition to various standard research texts.

The following review starts with an analysis of word usage across various genres of text, by means of CA. Then comes an example of superimposed graphs from sets I and J that aids human interpretation. Further studies reveal the importance of positive and negative sides of axes, and of the way in which CA graphs are split into quadrants. Then comes a study of trigrams, and further one that used WordNet alongside CA; those latter studies in particular inspire a novel approach to resolving ambiguity.

Analysing Word Usage across Genres of Text

The first example of applying CA to textual analysis offers relatively clear graphs, which serve as good illustrations. The domain of that study was the Lancaster-Oslo/Bergen Corpus (LOB), which comprises four million words from literary works, newspapers, and academic texts; it further covers fifteen categories that range from religion to science fiction.

Results from CA revealed groups of words that corresponded to particular genres: authors were more or less likely to use certain words in any particular genre. In addition, CA identified groups of genres that employed similar words (Nishina, 2007). In terms of the sets I and J that form the basis of any CA, various genres constituted set I, while words of interest comprised set J.

The first step, then, was to count occurrences of relevant words in the LOB. That yielded a matrix of 100 columns by 15 rows, to reflect word frequencies within text categories. For simplicity, categories were denoted as types A - R. Table 6.1 shows the first few columns and rows of that matrix:

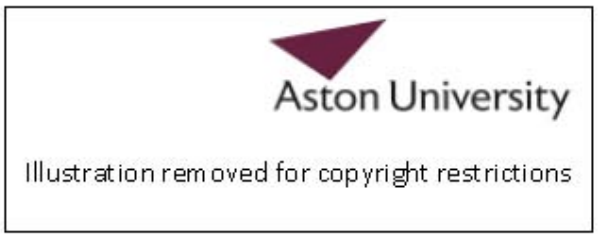

Table 6.1: Detail from a CA matrix of word frequencies by text category (Nishina, 2007)

Table 6.1 shows, for example, 2,018 occurrences of the word 'said' in texts from category A, Press Reportage. CA on that matrix produced the graph overleaf in Figure 6.3, which plots word frequencies against genres. Ellipses have been added to highlight particular categories, or groups of categories:

Figure 6.3: Plot of text categories based on word frequencies (Nishina, 2007).

Figure 6.3 shows points for categories J and A that appear outside the central cloud. Those categories were Learned and Scientific Writing and Press Reportage, respectively. In addition, a distinct group of points for categories K, L, N and P showed those genres to be quite similar. In fact, categories in that smaller cloud were some form of fiction[1]. In contrast, genres that congregate around the graph's origin had less well defined patterns of word-usage. All the same, the point for category R, Humour, is closer to the group of fiction genres than to other points. Humorous writing more resembled fiction than it did, say, academic journals in class J (Nishina, 2007).

[1]Fiction genres were K: general, L: mystery & detective, N: adventure & western, and P: romance & love story

Although CA identifies related genres in set I, words that characterised those groups do not appear in Figure 6.3. That information comes from Figure 6.4, which plots set J:

Figure 6.4: Plot of word frequencies based on text categories (Nishina, 2007).

The top-right quadrant of Figure 6.3 depicts the words 'different', 'form' , 'used' and 'system' as slightly removed from the central cloud. In the bottom-right quadrant, the words 'government', 'national' and 'year' comprise a more distinct group. Those groups constitute words that are likely to appear together, though in what genres is not clear. Just as the graph for genres lacked information about specific words, the graph for those words lacks information about genres. Those graphs must be compared in order to find correspondences between words and genres (Nishina, 2007).

Corresponding quadrants of the graphs from Figures 6.3 and 6.4 reveal relationships between words and genres. Take, for example, the top-right quadrant in the graph of genres, which contains an isolated point for category J, Academic Journals. The corresponding quadrant in the graph of words holds the group containing 'form' and 'system'. In a similar way, the bottom-right quadrants reveal a correspondence between category H, Government Documents and Industrial Reports, and the words 'government', 'national' and 'year'. Comparing graphs indeed showed that particular genres, or groups of genres, were characterised by different vocabularies (Nishina, 2007).

Issues Arising from the Study of Genres by Nishina (2007)

An issue with that study concerns categories A and J, academic texts and press reportage respectively. Those points appeared near the extremes of separate quadrants in Figure 6.3, leading to the conclusion that categories J and A are totally different. The word 'totally' suggests that words from one category

of text never appear in the other. It would be better to say that some correspondences between words and genres are stronger than others.

A further issue involves various labels on graphs, which reveal an important aspect of CA that received little attention. Headings and axes from Figures 6.3 and 6.4 bear percentages of any overall variation explained by resulting factors, reflecting their relative importance. The x-axis label, then, shows the first factor from CA to account for 59.04% of the total inertia. Factor two, on the y-axis, explained a further 15.10%. Summing those figures yields the total of 74.14% in the headings. Although both of those factors are important, the first was far more so. In addition, nearly a quarter of the inertia remains unexplained, which might point to further research.

In addition, note that axis labels on the graphs from that study use the word 'Dimension'. In fact, dimensions reflect the cardinality of set J in any analysis (Benzécri, 1992). Put another way, the space that CA creates has as many dimensions as it has matrix columns. The word 'factor' would be more accurate, as that is what CA actually projects on graphs (Murtagh et al., 2007). A further misconception arise as criticism of CA for failing to consistently divide genres into specific groups (Nishina, 2007). Rather than any fault in CA, though, it is the researchers' input matrix that failed to capture the desired patterns; indeed, Murtagh (n.d.) shows that data encoding is an important part of CA.

A final issue with the study by Nishina (2007) concerns text categories A and J; those points appeared near the extremes of separate quadrants in Figure 6.3, leading to the conclusion that academic texts in J and press reportage in A are totally different from the remaining types of text. The word 'totally', though, suggests that words from those categories never appear in other text genres. It would be better, then, to say that certain correspondences between words and genres are stronger than others. Having raised issues with that study of genre, then, attention now turns to approaches that addressed those issues. The first apparent improvement is in visualising CA results; rather than comparing graphs for sets I and J in isolation, superimposed graphs aid interpretation. That is permitted because CA maps both sets into the same space (Murtagh et al., 2008).

Superimposing Graphs from Sets I and J

A good example of superimposing graphs comes from a study of the synoptic gospels: Mark, Matthew, and Luke. CA analysed frequencies of occurrence for types of subordinate clause, such as comparative and conditional clauses, in Greek versions of those gospels. In addition to particular gospels and clauses, a third measure reflected variations in discourse. The narrative form, for example, does not report direct speech, whereas parables contain quotes from participants in a story (Unmans, 1998).

Because CA processes just two sets, discourse could not be addressed directly. Overcoming that constraint

meant compressing three measures into a two-way contingency table that CA could use. To that end, gospels were split into sections; each section reflected a specific type of discourse, and contributed a row to the CA matrix. Columns in that matrix stood for types of subordinate clause, while individual cells held the frequency of a particular type of clause in specific source (Unmans, 1998).

Figure 6.5 shows the resulting superimposed graphs, for the first two factors arising from CA. Points for the gospels appear in bold face, and end in a letter that reflects a specific type of discourse. Subordinate clauses are shown in normal type. The exact meaning of those labels is unimportant, here; rather, it is how overlaid graphs help to reveal correspondence between sets. Particular types of clause can be seen to congregate around groups of gospels. Some such clusters have been highlighted in ellipses:

Figure 6.5: Stylistic analysis of the synoptic gospels (Unmans, 1998)

Although little interpretation of Figure 6.5 was offered, it demonstrates the benefit of plotting CA graphs together. In addition to showing clear clusters of gospels and clauses, superimposed charts make obvious any correspondence between the two categories.

Take, for example, clusters that span the upper and lower left-hand quadrants. There, narrative sections from all three synoptic gospels form a well-defined cluster of points LkN, MtN and MkN. Clauses of type Pga, Pc1 and Pc2 cluster together in a similar way, respectively associating clauses having absolute participles with those having various conjunctive participles. Superimposed graphs make obvious the specific correspondence between those sections and clauses.

Although Figure 6.5 gives no percentages of inertia, the first two dimensions actually represented

78.1% of the total information content. Further to investigating subordinate clauses, further graphs plotted, for example, correspondence between gospels and certain Greek words. In all cases, attention was paid to the importance of factors arising from CA. Proportions of inertia attributable to the first two factors from those analyses ranged from 45.8% to 99.2%. The latter figure, in particular, showed that higher dimensions of the CA space held most of the essential information (Unmans, 1998).

Another aspect of note from that study of the gospels concerns the distances between points. While quadrants on a graph reflect major differences between sets, distances between points reflect the extent of any resemblance. Mutually close points reflect a high degree of correspondence, whereas points that are widely separated are very dissimilar. That applies to distances between row points and between column points[1]. Although particular points may not be close in the first two dimensions, they might well be so in higher dimensions. For that reason, unequivocal clustering must sometimes account for additional factors, not just the two most important ones (Unmans, 1998).

[1]It is important to add that CA allows comparisons both between and within those sets of points.

Using CA to Analyse Trigrams

Rather than counting individual words, the CA matrix for the next study comprised frequencies for trigrams; in fact, those trigrams came from essays by students of English as a Foreign Language (EFL), at five levels of education. Trigrams components were assigned POS by the TOSCA tagger; for example the trigram ADJ-N-PREP represented an adjective, followed by a noun, then a preposition (Tono, 1999). Resulting correspondence between age and patterns of trigram usage, then, appear as Figure 6.6, which in particular shows junior school children to use mainly nouns, while university students accounted for most trigrams involving prepositions. Those two clusters have been circled in blue:

Figure 6.6: Plot of the relationship between trigrams and age (Tono, 1999)

Points for junior school children from Figure 6.6 cluster at the outskirts of diagonally opposed quadrants; between those extremes, the three remaining age groups were alike in employing largely verb-related trigrams (Tono, 1999).

Introducing WordNet into CA

The last study under review combined WordNet and CA, to investigate any effect on intelligibility of replacing words in sentences. The similarity of replacements to original words was assessed, in part,

using WordNet's measure of semantic distance, which reflected degrees of separation between words from underlying semantic networks. For example, the distance between 'car' and 'gasoline' was smaller than that between 'car' and 'bicycle'.

Students at various levels of ability in English had to substitute words in sentences. Human annotators judged the suitability of replacements, using categories ranging from 'clear' to 'unintelligible'. Figure 6.7 shows the interaction between proficiency, and the semantic distance that gave clear substitutions:

Figure 6.7: CA graph showing semantic distance of acceptable word substitutions, by age (Izumi et al., 2007)

Figure 6.7 clearly separated students at level 9, in the top-right quadrant, from those having lower levels of proficiency. Very able students tolerated the greatest semantic distances, and appeared near the edge of the graph. Low ability students, on the other hand, showed a corresponding intolerance: substituted words had to be closely related to any original word. Remaining groups understood reasonably distant substitutions, and clustered around the centre of the graph.

In fact, WordNet was just one of several measures of semantic distance that was employed. Separate runs of CA were needed to assess the effect of any specific measure of distance. That was due to the need for simple contingency tables, which was seen as a clear limitation of CA (Izumi et al., 2007).

While demonstrating a notable combination of trigrams, WordNet and CA, that study raises a difficulty with interpreting graphs. Points for ability in English between levels 3 and 8 clustered around the origin, separating them from the lowest and highest levels. The problem arises of deciding just how

many clusters are reflected, and what points belong to each cluster. If asked for just three clusters, CA might show the central points as a single group. Four clusters would most likely split levels 7 and 8 into a separate cluster. Assessing the optimum number of clusters, then, poses a challenge to humans, and more so to machines.

Summary of CA for Analysing Text

Common threads emerged between applications of CA to researching text that will close this section. In general, the distribution across categories of one set were used to explain the spread in a second set, and the same in reverse. Graphs from CA largely plotted one factor on the x-axis against a second factor on the y-axis. The coarsest measure of correspondence was the split between positive and negative sides of a particular axis. The next level of detail arose from interactions between the two axes, giving four quadrants that reflected major sub-categories; further specific correspondences arose on inspecting the distances between points (Unmans, 1998; Nishina, 2007).

With the exception of Nishina (2007), the reviewed studies overlaid graphs to aid comparisons between sets I and J, which occupied comparable CA spaces. Another area of agreement lay in noting any contribution made by specific factors to any overall variation. The two factors normally shown in a CA plot, though, might account for just a proportion of that variation; a certain amount of variation will remain unexplained, should just a few dimensions be considered. That said, any first two factors generally accounted for an acceptably large percentage of overall variation.

Work on analysing trigrams showed the benefit of considering n-gram frequencies rather than counting individual words. Given the importance to this thesis of trigrams that contain prepositions, the study by Tono (1999) was of particular interest. Combined with WordNet, as in work by Izumi et al. (2007), CA offers a powerful tool for discerning patterns of preposition usage in GRiST mind maps. Rather than as an end in itself, though, such patterns will further aid in disambiguating important words from those mind maps, in turn enhancing the emerging information base.

A unanimous aspect of CA in analysing text, though, concerned degrees of human intervention. Indeed, all of the reviewed studies ran CA manually, with humans inspecting any results. Maximising automation in processing GRiST mind maps, though, requires machines to perform all aspects of CA unaided. While building matrices and running CA are easy enough, reading graphs will be more of a problem, as will be selecting optimum numbers of factors and clusters. Further, machines must collate row and column clusters, and interpret points towards the edges of graphs, whose axes must further be analysed. All of those functions are normally done by eye; what follows, then, is a proposed approach to automating CA for GRiST mind maps.

6.4 Automating CA for GRiST Mind Maps

The primary output from CA, then, comprises a pair of graphs that represent separate, though related, analyses: one for rows from the CA matrix, and a further one for columns. Due to that correspondence, superimposed graphs were commonly used to aid interpretation by humans. For CA to process GRiST mind maps automatically, machines must emulate that ability to interpret graphs. Rather than treating graphs as images, though, underlying tables of results are more readily processed by machines; indeed, those tables reflect the profiles that contributed graph coordinates in the first place. Retaining CA results in memory makes them available after actual runs are complete, allowing machines to perform subsequent analyses. The tasks that machines must perform unaided, then, are:

1. build an input matrix, and run the analysis;
2. determine optimum numbers of factors and clusters;
3. establish mappings between row and column clusters;
4. identify outlier points or clusters at the edges of graphs;
5. interpret the X and Y axes.

Apart from building an input matrix, those tasks are generally done by humans; retaining results tables arising from CA, though, will allow machines to emulate humans. One such table holds factor values for rows and columns, which explain various proportions of any overall variation. Further tables reflect any inertia attributable to those factors, while a third holds any clusters emerging from CA. The first of those tasks, then, demands that machines build an input matrix, and run CA, by means revealed next.

Building an Input Matrix, and Running CA

Columns for the CA matrix comprise pairs of meta-types fused into a single string. The first meta-type in such pairs represents a word that immediately preceded any particular preposition, while the second meta-type is for words that followed that preposition. The three meta-types ‘a’, ‘m’ and ‘t’, respectively standing for ‘action’, ‘modifier’ and ‘thing’, yield nine permutations that constitute matrix columns:

aa, am, at, ma, mm, mt, ta, tm, and tt.

Given the dependence of any results on those meta-type pairs, understanding exactly what they portray is vital. The first pair, ‘aa’, means that words immediately preceding and following a given preposition were both reported to be actions, that is verbs, by WordNet. In contrast, the second pair, ‘am’, indicates a modifier as the word that followed any preposition; in turn, modifiers constitute both adjectives and adverbs. Those nine pairs, then, reflect all permutations of meta-types around prepositions.

In contrast, rows record the frequencies of specified meta-types appearing around particular prepositions in text from GRiST mind maps. Consider, for example, the preposition 'to' that might be preceded by a noun and followed by a verb; that would be reflected by incrementing the value of the corresponding cell, at the intersection of that row and column. Individual cells, then, record how many times a specific preposition appeared between two given meta-types.

As for running CA itself, the algorithm provided by Murtagh (2005) was amended to retain results on completion, allowing machines to run CA and to subsequently process any results. Further enhancements were applied to allow results from an initial run to seed a further run, having determined optimum numbers of factors and clusters; that process is explained next.

Determining Optimum Numbers of Factors and Clusters

The next task, then, is to determine optimum numbers of factors and clusters for CA. Those settings are invariably set by humans, perhaps guided by a preliminary analysis. Having judged what numbers of clusters are likely to emerge, a subsequent run might produce more desirable correspondences based on those settings. To mimic that process, an initial CA allows machines to determine optimum numbers of factors and clusters for a second run, which produces a more discerning analysis.

Relatively unimportant factors from that initial CA will be subsequently ignored, should they account for less than a specified proportion of the total variation; that information is readily available from resulting CA tables of percentages of inertia explained. In contrast, deriving the best number of clusters requires more work, starting with a minimum number of clusters expressed as $\mathrm{cl}_{\mathrm{default}}$. Additional clusters shown as $\mathrm{cl}_{\mathrm{extra}}$ are justified by inspecting results tables from CA; $\mathrm{cl}_{\mathrm{extra}}$ is incremented on encountering large differences in factor Δ_1 between consecutive row results from set I. Summing the number of default and extra clusters, then, gives the optimum number of clusters $\mathrm{cl}_{\mathrm{opt}}$ shown in the following expression:

$$\mathrm{cl}_{\mathrm{opt}} = \mathrm{cl}_{\mathrm{default}} + \mathrm{cl}_{\mathrm{extra}}.$$

In fact, a value of $\mathrm{cl}_{\mathrm{default}} = 2$ is used in all cases, which assumes enough variation to exist as to identify at least two sub-divisions; otherwise, CA would largely be a waste of time.

Values for $\mathrm{cl}_{\mathrm{extra}}$, then, arise from identifying major differences in factor Δ_1, henceforth simply termed F1, within row results from set I. In fact, tuples from such results hold a label along with values for each factor. Sorting row results on F1, then, forces tuples having the lowest and highest F1 to opposite ends of the resulting list; row labels allow machines to keep track of particular tuples. From those extremes, termed $\mathrm{r}_{\mathrm{min}}$ and $\mathrm{r}_{\mathrm{max}}$ respectively, comes the absolute range of F1 values R_{F1}:

$$\mathrm{R}_{\mathrm{F1}} = |\mathrm{r}_{\mathrm{max}}.\mathrm{F1} - \mathrm{r}_{\mathrm{min}}.\mathrm{F1}|.$$

Armed with that absolute range of F1 values, the next step compares consecutive row results. Any absolute difference in F1 between such successive results is divided by the absolute range for the entire list. Subsequently, the difference d_{F1} between any two row results expresses the proportion of the overall range separating them. The number of extra clusters is incremented when d_{F1} exceeds a given percentage.

In that way, large F1 differences between consecutive rows reflect boundaries between clusters; adding any extra clusters to the default of two yields an optimum number of clusters for a further CA. Emerging graphs, though, depict two discrete, though related, analyses. That, in turn, raises the problem of creating mappings between row and column clusters, which humans do so easily when inspecting overlaid graphs.

Mapping Between Row and Column Clusters

The task of establishing mappings reflects why graphs are usually overlaid: humans easily discern related row and column clusters in combined graphs. Machines, though, must derive such mappings from retained CA results, in order to relate clusters from sets I and J, the respective rows and columns. To that end, the average F1 for any particular cluster will be taken as a spatial mean, depicting clusters from sets I and J as single points.

Reducing clusters to single points simplifies the challenge to machines; all that is needed is to sort CA results for sets I and J. Sorting those results on the first factor, F1, reflects the position of associated points along the X-axis. F1 generally explains larger proportions of total inertia, and is the major influence on where clusters appear. Sorting results, then, aligns them on F1 values, and in that way, by relative position on the X-axis.

In fact, two CA runs are needed; the first uses prepositions as rows and meta-type pairs as columns, while a subsequent run uses those same observations, but with rows and columns interchanged. Columns for that second analysis, then, constitute prepositions, while meta-type pairs become matrix rows. Sorting both sets of results on F1 aligns corresponding clusters from two such analyses. Further discrimination, should it be required, arises from treating F2 in a similar way.

By the means described, machines might emulate humans in interpreting CA graphs, so as to reveal affinities between meta-types and prepositions. One aspect of CA graphs, though, will prove particularly useful in analysing GRiST mind maps, in that points and clusters near the edges of CA graphs reveal extremes of variation; such outliers are addressed next.

Outlier Clusters

Outliers, then, appear at extreme positions on any CA graph. In fact, such extremes might be removed, to improve any data model (Bullen et al., 2003). That view, though, sees outliers as a disruptive influence; in contrast, I take outliers as strong correspondence between certain prepositions and meta-types. Points

within corresponding outliers for sets I and J are highly attracted, while remaining points on the graph are accordingly repelled. That influence of classical mechanics on CA suggests interpreting outliers as particularly large masses in a state of equilibrium with smaller ones, arising from balancing forces between points. Removing any outlier, then, would demand reconfiguring the graph in order to restore equilibrium.

Having noted any strong correspondence between outliers, removing them will spread remaining points more evenly. To that end, a further run is performed on removing outlier rows from the matrix; meta-type pairs in set J, though, are left to realign with remaining prepositions from set I. Eliminating highly influential row outliers encourages interactions between weaker forces. Meta-type pairs will change allegiance, once free of that stronger force. Repeatedly removing outliers until none emerge will yield graded associations between points on a CA graph. Points agglomerated in that way, though, are less strongly associated with a given cluster than were any original members.

Mapping clusters between sets I and J, in fact, will further identify outliers; peripheral clusters will have large mean F1 values. Further, extreme clusters become more obvious on requesting just two clusters from CA. The larger of two such resulting mean F1 values identifies the centre of that more distant cluster, which is deemed an outlier should its average F1 exceed a given limit. Removing outlier row clusters further returns corresponding column points to a pool; running CA on a reduced matrix realigns those meta-types with the next most attractive preposition. That will permit graded associations between prepositions and meta-types, and provide a measure of certainty for resulting decisions that resolve ambiguity. Having shown a means of identifying outliers, attention turns next to interpreting CA axes.

Interpreting Axes

Outlier clusters, then, reflect strong correspondence between members of sets I and J, and in addition distinguish them from other clusters. Further, splitting composite column labels back into pairs of meta-types reveals specific types within such clusters, one of which might dominate any outlier. That, in turn, suggests one extreme on any axis to reflect *either* things, *or* actions, *or* modifiers. In a similar way, corresponding prepositions at opposite ends of any axis will be opposed in what meta-types commonly surround them. Note, though, that such interpretations do not reflect concrete ideas; rather, an axis might be thought of a continuum expressing, say, 'thing-ness' at one extreme and 'modifier-ness' at the other.

A further way of contrasting prepositions is to locate symmetrical column labels, which record identical meta-types on either side of specific prepositions. In a similar way to any dominant meta-type within a cluster, the symmetrical labels 'aa', 'tt' and 'mm' respectively indicate extreme affinity for things, actions or modifiers. Should axes successfully distinguish between prepositions, such symmetrical labels might be found in mutual opposition along the X and Y axes.

A final point of interest concerns column labels that are mutually inverse, for example, ‘at’ and ‘ta’. Rather than a specific overall meta-type, such points place two meta-types in opposition, particularly when they appear as outliers. That, in turn, further places in opposition prepositions that correspond to those meta-types.

6.5 Chapter Summary

This chapter about Correspondence Analysis started with an introduction, which reviewed reasons for using clustering of any sort. A subsequent overview of CA revealed a form of multivariate data analysis having a wide range of applications. Input to CA was seen to comprise frequencies of occurrence within certain categories; such counts reflect specific variables’ presence in, or absence from, those categories. From such a matrix of observations, CA identifies factors that explain patterns in underlying data, and in addition, any interactions between those factors. Graphs, the main output from CA, aid humans in visualising any such emerging patterns.

Of particular importance were profiles, which reveal matrix rows having similar proportions. Further, values from profiles constitute coordinates for points on a graph, in a spatial representation of sets I and J that respectively reflect rows and columns. Imbuing points with mass followed from classical mechanics, which in turn led to forces between those masses; that, in turn, gives rise to inertia, which measures variation within any matrix. Subsequent interpretation of factorial axes was shown to further help in understanding output from CA.

Following that overview of CA came a review of studies that used it to research text, first in relation to analysing word usage across genres. Those studies emphasised that graphs are commonly superimposed, to aid identifying corresponding points from sets I and J. Further, specific quadrants of any CA graph were shown to make important distinctions between genres. Following that, CA was seen be useful in analysing trigrams that introduced WordNet into CA, suggesting a means of resolving ambiguity.

In respect of full automation, machines were shown to face various challenges; building an input matrix and running CA were the least of those problems. More demanding tasks involved obtaining optimum numbers of factors and clusters, and subsequently mapping between row and column clusters. That was shown to allow machines to identify and analyse outliers, which in turn, suggested interpretations of X and Y axes on resulting graphs. Finally, this summary closes the chapter, in order to present experiments in applying CA to resolving ambiguity in WordNet, and in GRiST mind maps.

7

Experiments in Resolving Ambiguity

7.1 Introduction

Chapter 5 described prepositions as function words, or stop-words, that are often removed before analysing text. Rather than ignoring stop words, though, the proposed approach uses one particular type, prepositions, to resolve ambiguity in GRiST mind maps. To that end, reliable patterns of word usage around prepositions will subsequently act as templates for resolving ambiguous cases. Clearly, machines must be able to process stop-words, particularly prepositions.

Therein, though, lies a problem: WordNet does not hold information about stop words. To overcome that problem, the 'stop list' provided by Pedersen (2001) was used to build an adjunct to WordNet, which provides the corollary of a synset for words from that stop-list. Although no pointers exist in such pseudo-synsets, they do indicate a main POS type and an optional sub-category. That WordNet adjunct, then, will allow machines to research stop words as if they existed in WordNet.

There are, in fact, six steps in analysing mind maps, for which an important distinction must be made; that is between the words 'triple' and 'trigram'. For current purposes, triples comprise a preposition and two actual words; trigrams, on the other hand, arise from transforming the POS for those words into meta-types. The required six steps are, then:

1: Identify triples of the form $\{word_1,\ preposition,\ word_2\}$ in GRiST mind maps.

2: Of the triples from step 1, select just those that contain unambiguous words.

3: Using POS reported by WordNet, transform unambiguous triples from step 2 into trigrams of the form $\{meta\text{-}type_1,\ preposition,\ meta\text{-}type_2\}$.

4: Perform CA on the frequencies of occurrence of unambiguous trigrams from step 3.

5: Select triples from step 1 that contain ambiguous words, transforming them into trigrams as in step 3.

6: Use CA clusters for unambiguous trigrams from step 4 to resolve ambiguous triples.

The above steps were implemented in a bespoke Java class called `MidmapPOSAnalysis` that performed CA on trigrams centred on prepositions. The remainder of this chapter presents experiments that assess that class in respect of resolving ambiguity in GRiST mind maps.

7.2 Identifying Triples Centred on Popular Prepositions

The `MidmapPOSAnalysis` class, then, detected prepositions by means of an extension to WordNet, which processed stop words from lists created by Yang and Pedersen (1997) and by Pedersen (2001). The resulting adjunct was a new Java class called `AstonWordNet`, which encapsulated researches for stop words in WordNet, be means of a further class called `WNet`[1]. That sub-class held senses for the various POS found, as well as a gloss, if available. In addition, `AstonWordNet` created pseudo-synsets for stop words, mimicking WordNet's own results for POS from content words[2].

That new facility for stop words, then, allowed `MidmapPOSAnalysis` to identify prepositions. In fact, just triples having one of the ten most frequent prepositions were selected. Further, just one instance of any specific triple was processed, although separate full counts were kept for comparison. Triples, then, comprised a particular preposition, and the words immediately preceding and following it. Because WordNet was not yet consulted, resulting triples contained both reliable and ambiguous words. Any adjacent words in triples will be addressed shortly; for now, attention focuses on the preposition components.

Method I for identifying prepositions

The first step, then, was to identify in GRiST mind maps any triples centred on prepositions. Because any specific triple might be embedded in text from various nodes, just the first occurrence of any given triple was processed. In practice, then, a particular order of words around a specific preposition was counted just once.

The `MidmapPOSAnalysis` class started by retrieving a full list of mind map nodes from the database. Those nodes were processed sequentially, with text from any particular node held in the variable `nodeText`, which shows the formatting style for Java code, here. For each node, individual words were obtained by means of the `split()` method of the `String` class. In that way, the following Java code yielded an array of words from the `nodeText` variable:

```
String[] wordArray = nodeText.split(\\W);
```

The regular expression '`\\W`' dictated that words should be alphabetic strings, separated by non-alphabetic characters such as spaces and punctuation marks; special characters, though, were stripped away, as Java uses such separators just to identify boundaries between words.

Subsequently, `AstonWordNet` detected prepositions in arrays of words from mind map nodes. Should a preposition appear at index i of any array, words on either side of that preposition lay at respective

[1] The new `AstonWordNet` class was based on code from Beckwith et al. (1993).
[2] The number of senses in such pseudo-synsets from `MidmapPOSAnalysis` was always given as 1.

indices $i-1$ and $i+1$. Misspelled words were corrected, where necessary, by spelling corrections arising from Chapter 2. Resulting triples took the form $\{word_1,\ preposition,\ word_2\}$. Counts for unique triples, and for all triples, were then accumulated across the collection of GRiST mind maps.

Results I of identifying prepositions

Table 7.1 shows the ten most frequently used prepositions from GRiST mind maps. The first row shows full counts for nodes having a specific preposition, whereas the second row shows unique occurrences:

Counts	Prepositions									
	about	as	by	for	in	like	of	on	to	with
All Triples	91	82	42	140	311	42	626	225	951	178
Unique Triples	82	75	41	124	276	38	507	119	660	156

Table 7.1: Ten GRiST Prepositions for analysis.

Frequencies from Table 7.1 varied greatly, from as low as 41 unique triples centred on 'by' to 660 for 'to, with 'of' closely behind with 507. Duplicate triples, though, were ignored; Table 7.2 gives such a triple, which occurred in several different nodes:

Triple	Examples from GRiST mind map nodes
{history, of, abuse}	{*history of abuse*} potentially important factor
	often people with a {*history of abuse*}
	2 kids under 5 and a partner they don't get on with and a {*history of abuse*}

Table 7.2: Multiple occurrences of the triple {*history of abuse*}.

In practice, the first such node encountered supplied a representative tuple. Just the first tuple from table 7.2, then, was used in the main analysis.

Discussion I of identifying prepositions

The greatest variation between full and unique counts from Table 7.1 arose for the preposition 'to'. That was due in part to the template mind map used in GRiST, which led to particular nodes repeating across the 46 resulting mind maps. Repeated occurrences of the node `[abuse to client]` are a good example; counting that node just once for the preposition 'to' gave a more representative figure. The preposition 'of' gave the next greatest difference in counts. In that case, the node `[state of mind]` was over-represented.

In addition, certain nodes occurred under several risks within GRiST mind maps. For example, the node `[abuse to client]` appears under both `[life history]` and `[personal details]`. Accounting for just a sole representative of any particular node, then, normalises frequencies for the CA matrix.

7.3 Identifying Unambiguous Triples

Triples comprising a preposition and two unambiguous words, then, indicate reliable patterns of usage. Those patterns will later act as templates for helping to resolve ambiguous cases. The problem is that WordNet often reports several POS for any given word; completely unambiguous results, though, must just one POS. By that definition, many words remain potentially ambiguous; in such cases, machines must examine the number of senses reported by WordNet for each candidate POS, in order to reveal the most likely one by means of familiarity.

Method IIa for collating unambiguous triples

Words that appeared around each of the top ten prepositions were researched in WordNet. Words with several POS were considered unambiguous, should one have two more senses than did any remaining candidates. To that end, the new `AstonWordNet` class uses the `JWNL` Java package, which comes with WordNet. That package provides a `Dictionary` class that allows any Java program to research words. A further `JWNL` class called `IndexWord` retrieves synsets from WordNet. Instances of `IndexWord` represent a particular POS, and carry fields such as the number of senses. That information is retrieved by using the `getSenseCount()` method in the `IndexWord` class. The following code, then, shows how `AstonWordNet` researches a word:

```
Dictionary dictionary = Dictionary.getInstance();
IndexWord indexWord = dictionary.lookupIndexWord(pos-type, word);
int senses = indexWord.getSenseCount();
```

Having created an instance of the `Dictionary` class, the `lookupIndexWord()` method retrieves information for a specified POS; although the subsequent call to `getSenseCount()` has no arguments, the instance of that method pertains just to the retrieved `IndexWord` object for a specific POS. The `getSenseCount()` method, then, was invoked separately for each of the four POS held in WordNet. Should all four attempts fail to identify a word, the stop-word adjunct in `AstonWordNet` provided a pseudo-synset, if possible. All results were stored as instances of `WNet`. Any specific preposition, along with results for the two associated words, were stored in a further bespoke Java class called `Triple`.

Any specific `Triple` object, then, held a particular preposition, two words from GRiST mind maps, and a unique key, *nodeID*, of any associated node used to retrieve associated node texts. Further, two `WNet` objects represented each of any triple's content words, along with those words' familiarity: the sum of all of the senses reported by WordNet for a specific POS. Should several candidate POS arise, the preferred one had a familiarity of two or more senses greater than any remaining POS. Further, all POS held a ratio of the highest sense count and the next highest.

Results IIa of collating unambiguous triples

Figure 7.1 shows an example `Triple` object created by `MidmapPOSAnalysis`; that triple carried two unambiguous words, 'known' and 'culture', respectively reported as a verb and a noun. The first line in that triple holds a unique identifier for the representative node. Subsequent indented lines show the two `WNet` objects associated with that triple:

```
{known ABOUT culture} in node 11781: "what known about culture sub/culture"
  ["known"     verb (11), action, [adjective (1)] (0.09090909) (fam=12)]
  ["culture"   noun (8), thing, [] (fam=8)]
```

Figure 7.1: Unambiguous instances of the new WNet class.

The two `WNet` instances from Figure 7.1 report unambiguous POS for 'known' and 'culture'. Indeed, the `WNet` instance for 'culture' holds just a sole POS, which was accepted without further analysis. The word 'known', though, had two candidates, as a verb and as an adjective; the large difference in senses of 10 preferred the verb, which was duly assigned a meta-type of 'action'[1]. Remaining entries in square brackets show alternative interpretations that were declined, in this case, just the adjectival POS for 'known' that had just one sense; that, in turn, gave an overall familiarity of $11 + 1 = 12$. Accordingly, the ratio of senses for 'known' was $11 : 1 = 0.09$, when rounded to two decimal places.

Table 7.3 next presents triples that were treated as unambiguous in that way. Even though some words had several candidate POS, one form was predominant; examples in the first column depict such words in italics, with the associated preposition in bold[2]. The next column restates those words, for clarity. After that come the POS reported for any word, as adjective, adverb, noun, or verb. The column Δ_s gives the difference between the highest sense count and the next highest. The final column shows the selected POS, which had two more senses than the next most likely interpretation:

Examples from GRiST mind maps	Ambiguous Word	POS Senses					Best POS
		Aj	Av	N	V	Δ_s	
[felt **like** *battering*] wife	battering	-	-	1	3	2	V
[focussed **on** *specific*] thing	specific	4	-	2	-		Aj
constant [*smell* **of** urine]	smell	-	-	5	3		N
how surprised / not surprised [*still* **with** us]	still	6	4	4	4		Aj
rq : SH if it provides some [kind **of** *release*]	release	-	-	11	9		N

Table 7.3: Correctly identified unambiguous words, for senses difference = 2.

POS for ambiguous words from Table 7.3 were deemed appropriate, yielding triples centred on the prepositions 'like', 'on', 'of', and 'with'. A difference of 2 senses arose from subtracting low counts, such

[1] In fact, just the meta-type 'a' was stored on any `WNet`, in a normalised way.
[2] Henceforth, that use of square brackets, italics and bold type for triple components in nodes will be adopted.

as $3 - 1 = 2$, through to higher values, such as $11 - 9 = 2$. Indeed, that is why the preposition 'of' has two entries: the latter example was just to show high sense counts. In that way, the word 'release' that followed 'of' was interpreted as a noun, due to having two more senses than as a verb. Note that the word 'still' from Table 7.3 might have been any of WordNet's four POS, although it was in preference seen as an adjective.

In contrast to such appropriate choices, Table 7.4 lists some incorrect ones that arose from a sense difference of 2, which in turn arose from relatively low sense counts. Prepositions in the following triples were, then, 'for', 'in', 'of' and 'as':

Examples from GRiST mind maps	Ambiguous Word	POS Senses					Best POS
		Aj	Av	N	V	Δ_s	
less [*services* **for** women] with children	services	-	-	1	3	2	V
get [feeling **in** *objective*] & subjective way	objective	4	-	2	-		Aj
depends on their past [*experiences* **of** services]	experiences	-	-	3	5		V
what see the [*future* **as** holding]	future	5	-	3	-		Aj

Table 7.4: WordNet results incorrectly identified as unambiguous by senses difference = 2.

Note that choices from Table 7.4 were either between an adjective and a noun, or between a noun and a verb. Both triples from that former category were taken as adjectives, while those from the latter emerged as verbs. Such ambiguous results accordingly reduced the number of unambiguous triples. While a senses difference of 2 yielded 1601 unambiguous triples, a difference of 3 gave 1345 unambiguous triples: 256 fewer than with the less stringent test for a difference of 2.

Discussion IIa of collating unambiguous triples

The unambiguous triple from Figure 7.1 showed the word 'culture' to have 8 verb senses. That word was perfectly specific: WordNet suggested no other POS. In that same triple, the word 'known' had 11 verb senses, but just one as an adjective. A large difference of 10 senses made the verb much more likely.

In contrast, words in triples from Table 7.3 had several POS, although they were treated as unambiguous due differences of 2 senses between competitors. Choices were largely between nouns and verbs; in all but one of such cases, the noun was preferred. The exception was the word 'battering', which was correctly treated as a verb. Of the remaining triples from Table 7.3, two correct distinctions between nouns and adjectives arose. The word 'release' did indeed act as noun rather than as a verb, while 'specific' really was an adjective for 'thing'.

A final point about unambiguous triples from Table 7.3 concerns the word 'still', which could have been any of the four WordNet POS. The adjective had 6 senses, whereas each remaining POS had 4 senses.

That meant treating 'still' as an adjective, although it was really an adverb; the predominant POS proved inaccurate in that case. In fact, that did not matter: adjectives and adverbs were treated jointly as modifiers, so eliminating the noun and verb senses sufficed.

Triples from Table 7.4, though, revealed inappropriate choices between candidate POS, arising from relatively low sense counts. For example, the word 'future' was treated as an adjective, when it was in fact a noun. Although that will contribute an incorrect POS to subsequent CA, Fellbaum et al. (1990) note a tendency for nouns to act as adjectives in the English language. That, though, applies just to contiguous nouns, rather than those separated by prepositions. Further research might take into account such heuristics, although they would be better determined by machines that imposed by humans.

Although triples from Table 7.4 were treated as unambiguous, then, inappropriate POS were selected. The word 'services' had 1 noun sense and 3 verb senses. Although the predominant verb was selected, 'services to women' clearly intended the noun. Indeed, the idea of servicing women has a distinct and unwarranted sexual connotation. In a similar way, the word 'experiences' had 3 noun senses and 5 for the verb, which was chosen incorrectly. Further, the nouns 'objective' and 'future' were wrongly treated as adjectives. All of the errors described, though, arose for words having low sense counts: particular POS for such words had relatively few distinct meanings.

In fact, using absolute differences in senses might be too coarse a measure. Future research might apply ratios between sense counts, instead; for the moment, that is calculated but not used. That said, differences of 2 senses proved largely adequate, as opposed to the approach taken by Mihalcea and Moldovan (2000) that insisted on monosemous words: just those having a single POS were considered as unambiguous. Indeed, removing triples having multiple POS that yet have a predominant type would risk discarding an appreciable portion of the data. Any affect on ensuing analyses will be made clear in Section 7.6, when CA will be run for sense differences of both 2 and 3.

7.4 Identifying Ambiguous Triples

Unambiguous words, then, preferably have just a single POS. Words having several interpretations, though, might be adequately specific should one POS be particularly familiar. That, in turn, depends on the relative sense counts of competing POS; those having similar sense counts lead to ambiguity. Section 7.3 gathered unambiguous triples of the form $\{word_1,\ preposition,\ word_2\}$ in which both word components were unambiguous. The next step was to build a separate list of triples that hold ambiguous words, which will subsequently be resolved by reference to unambiguous cases.

Method IIb for collating ambiguous triples

The method here resembles that described in Section 7.3 for unambiguous triples. The sole difference is the actual test applied: that was for words having a difference of less than 2 senses between likely POS. As before, an instance of the `Triple` class held two `WNet` objects from the new `AstonWordNet` class.

Results IIb of collating ambiguous triples

Figure 7.2 shows a `Triple` object for representative node number 4735. In that triple, both content words were ambiguous. Consequently, associated `WNet` objects held no preferred POS:

```
high ON drugs     in node 4735: "rq : number one high on drugs"
  ["high"   [noun (7), adjective (7), adverb (4)] (1.0) (fam=18)]
  ["drugs"  [noun (1), verb (2)] (0.5) (fam=3)]
```

Figure 7.2: Ambiguous instances of the new WNet class.

The first `WNet` object, for 'high' from Figure 7.2 shows identical noun and adjective sense counts of 7, with the adverb having just 4. The second such object, for 'drugs', showed similarly ambiguous sense counts of 1 as a noun and 2 as a verb. In both of those `WNet` objects, insufficient differences in senses meant that no predominant usage arose.

Accordingly, Table 7.5 shows results for the ambiguous word 'drugs' that appeared in triples comprising prepositions 'in', 'of', 'on' and 'to', with examples from GRiST moved to the right side of the table:

Word	POS Senses					Prep	Examples from GRiST mind maps
	Aj	Av	N	V	Δ_s		
drugs	-	-	1	2	1	in	alcohol and [*drugs* **in** significant] amounts
						of	their [*use* **of** *drugs*]
						on	rq : number one [*high* **on** *drugs*]
						to	individuals [response **to** *drugs*]

Table 7.5: Examples of GRiST nodes having the ambiguous word 'drugs'.

The word 'drugs' had a just 1 sense as a noun, and 2 as a verb, giving a difference of 1 sense. Note further that in two triples, the content words 'use' and 'high' associated with 'drugs' were likewise ambiguous.

Finally, ambiguity sometimes arose due to WordNet's lack of coverage. An example was the words 'carer' and 'carers'. Table 7.6 shows that no POS at all were suggested for that word:

Word	POS Senses					Prep	Examples from GRiST mind maps
	Aj	Av	N	V	Δ_s		
carer(s)	-	-	-	-	-	of	[*carers* **of** any] kind
						with	where possible [collaboration **with** *carer*]

Table 7.6: The lack of WordNet results for 'carer' and 'carers'.

Discussion IIb of collating ambiguous triples

Results from Table 7.5 show the difficulty for machines in deciding best interpretations of words having similar numbers of senses. By itself, WordNet is uncertain about such words. In fact, examples from GRiST of the word 'drugs' all used the noun sense, rather than the verb. While humans might clearly see that no object for the verb 'drugs' is apparent in any of those nodes, machines encounter difficulties. A similar problem arose for the word 'high' that appeared along with 'drugs'. The fact that those words were separated by the preposition 'on', though, allows humans to determine 'high' as a modifier of a particular state, with the noun 'drugs' as the cause.

In contrast, Table 7.6 revealed that WordNet had no entry at all for 'carer'. While surprising, that well reflects the basic problem with WordNet: it gives very useful information about words that it does contain, but offers no help at all should words be absent from the underlying synsets. Fortunately, CA offers a way of resolving those problems by detecting patterns in words used in association with prepositions. That, though, depends on first deriving meta-types from POS, in a way that is shown next.

7.5 Deriving Meta Types from Parts-of-Speech

In line with WordNet, POS were reduced to 'actions' for verbs, 'things' for nouns, and 'modifiers' for adjectives and adverbs; those POS were assigned respective meta-types of 'a', 't' and 'm'. In that way, lists of triples became lists of trigrams, of the form {*meta-type*$_1$*, preposition, meta-type*$_2$}. A sole trigram arose for each unambiguous triple, while several trigrams were needed to express ambiguous triples. Any particular trigram carried a specific permutation of meta-types around a specific preposition.

Treating adjectives and adverbs collectively as 'modifiers' rendered certain words unambiguous; words having just those two POS were simply given a meta-type of 'm'. In addition, certain stop words were treated as nouns. In that way, unambiguous triples might arise despite lacking support from WordNet.

Method III for deriving meta-types from POS

Unambiguous triples from Section 7.3 were processed sequentially, and the `WNet` objects for any individual `Triple` object examined independently. For each `WNet` object, preferred and candidate POS alike were given corresponding meta-types. In the absence of a predominant POS, `WNet` objects were examined for adjectives and adverbs; a meta-type of 'm' was assigned, should no further POS exist for the associated word. In addition, the augmented `AstonWordNet` class revealed certain articles and pronouns. Demonstrative and personal pronouns such as 'I' and 'she' became 'things', as did the expletive pronoun 'it'; along with articles such as 'the', a meta-type of 't' was duly assigned.

Results III of deriving meta-types from POS

Table 7.7 shows articles and pronouns that were treated as things. For these examples, stop words appear in italics within triples, and prepositions in bold:

POS	Words	Examples from GRiST mind maps
articles	a, an, the	[intentions **for** *the*] future clear [evidence **of** *a*] mental health problem
pronouns	it	haven't [thought **about** *it*]
	her, him, she, them, they, you	absolutely out of blue well [planned **by** *her*] what [had **on** *him*]
	anybody, anyone, everybody, everything	often become [counsellors **for** *everybody*] [rare **for** *anyone*] to bleed to death

Table 7.7: Pronouns and articles that were treated as things.

Should assigning meta-types in that way resolve any ambiguity, such results were transferred to the list of unambiguous triples. That was the case for the triple 'rare for anyone', where WordNet reliably determined 'rare' as an adjective; further treating the indefinite pronoun 'anyone' as a noun made that triple unambiguous.

In addition to the 'things' in Table 7.7, reliable modifiers arose from words having just adjective and adverb POS. Table 7.8 gives examples of words treated in that way; no difference in senses appears, as that was irrelevant in such cases:

Word	Senses		Examples from GRiST mind maps
	Aj	Av	
all	2	1	[*all* **about** extents] and continuums
likely	4	1	women [are **as** *likely*] to be violent
just	4	5	i'ts [*just* **as** serious] to me

Table 7.8: Words that were safely treated as modifiers.

Discussion III of deriving meta-types from POS

Rather than treating words as POS, assigning meta-types stresses roles that words fulfil. That reflects the distinction WordNet makes between actions, things and modifiers. In addition, the new `AstonWordNet` class identified stop words such as articles and pronouns that reflect unambiguous 'things'. In addition, words that could be just modifiers, whether as adjectives or as adverbs, were treated as unambiguous. Overall, the pool of unambiguous trigrams for subsequent stages grew. Having transformed triples into trigrams, attention turns next to how CA found patterns within those trigrams.

7.6 CA on Unambiguous Trigrams: Pass 1

The sole automated stage of studies reviewed in Chapter 6 involved gathering observations, which comprised frequency counts that further constituted the bodies of any CA matrices. Observations arose from trawling thorough text corpora for given words or n-grams, where the sizes of various bodies of text used made that impractical for humans. Having created a contingency table of frequencies across certain categories, interpreting CA results was a purely human affair. In contrast, automating CA involves the following steps:

1. build an input matrix, and run the analysis;
2. determine optimum numbers of factors and clusters;
3. establish mappings between row and column clusters;
4. identify outlier points or clusters at the edges of graphs;
5. interpret the X and Y axes of graphs.

In conjunction with the `MidmapPOSAnalysis` class introduced earlier, a further new Java class, `AutoCA`, performed those steps automatically. Retaining results tables allowed machines access to them after CA itself completed; such CA results guide all of the above tasks.

In fact, `AutoCA` runs separate analyses for rows and columns, meaning that any resulting clusters must be mapped onto one another. Indeed, humans do that with little need for conscious thought. Another task to be automated is the identification of outliers, which express strong correspondence between certain prepositions and meta-type pairs. Any skew in analyses caused by outliers is eliminated by removing them prior to subsequent phases. Once mappings have been established, the analysis determines meta-types that correspond to the X and Y axes. All of the above tasks will ultimately help to revolve ambiguity in GRiST mind maps by providing reliable templates of word usage.

The first phase of CA

CA takes as input a matrix of rows and columns. In this section, labels for columns comprise pairs of meta-types. The first meta-type in such a pair represents a word coming before any particular preposition, while the second meta-type stands for the word following that preposition. Nine permutations exist for the three meta-types ‘a’, ‘m’ and ‘t’, namely:

aa am at ma mm mt ta tm tt.

The above pairs constitute the matrix columns for all analyses reported here. On the other hand, matrix rows have labels that denote prepositions. Individual rows record how many times a specific preposition

appeared between meta-types from the column labels. The contingency table compiled automatically by `MidmapPOSAnalysis` forms the basic matrix for CA. Matrix headers required by the Java implementation from Murtagh (2005) were added to control any analysis. That free-ware formed the basis of the `AutoCA` class invoked by `MidmapPOSAnalysis` to perform the required steps in CA.

In fact, CA was run twice on any given matrix. The first pass was for just 2 clusters of prepositions, and 2 clusters of meta-types. That was to identify outliers on resulting graphs, and further to suggest optimum numbers of factors and clusters for a second pass. In addition, text-based graphs from software by Murtagh (2005) were replaced by improved versions from new Java classes based on the `JFreeChart` shareware package (Object Refinery Limited, 2005–2009).

Method IV for a preliminary CA

Trigrams from unambiguous triples were processed sequentially, so as to accumulate counts of associated meta-types. The two meta-types in any individual trigram were joined, to form a pair that was counted for the preposition from the trigram. Having accumulated the required frequencies, adding a header gave the following matrix:

10 9 SUPNO 3 2 2	mm	mt	ma	tm	tt	ta	am	at	aa
about	1	13	3	5	28	2	4	21	7
as	7	10	4	14	18	5	9	8	2
by	0	9	4	1	5	1	3	18	1
for	6	19	3	10	50	6	6	24	3
in	16	46	4	23	89	7	24	72	4
like	0	7	0	7	10	1	1	11	4
of	7	67	3	52	304	42	2	31	7
on	3	14	2	9	32	0	18	39	4
to	15	41	144	19	72	98	24	94	167
with	6	21	0	14	50	2	8	53	4

Figure 7.3: CA matrix for phase 1

The matrix header begins with the numbers of rows and columns: 10 and 9 respectively. The next part, SUPNO, means that no supplementary rows were examined after the main analysis. After SUPNO, 2 factors are specified for the analysis. The last two numerical entries specify numbers of clusters required for rows and columns respectively. Last come the actual column labels, which are pairs of meta-types.

The matrix body in Figure 7.3 shows counts of meta-types across ten prepositions. A value of 4 in the bottom-right cell, for example, means that four instances of 'with' had words corresponding to actions on either side, shown by the column label 'aa'. Two CA runs were performed on that matrix, with the first giving optimum numbers of factors and clusters for a second pass. Just the two 'clusters' fields in the matrix header need be updated for that second CA: actual observations were left untouched. Raw

results tables for row and column, and associated clusters, were retained for further analysis of factors F1, F2... F_n that arose from CA.

Results IVa for a contingency matrix using 2 senses difference

Results from CA comprised factors for rows and columns, and a graph that plotted F1 against F2; an additional graph comes from a further analysis that uses optimum numbers of factors and clusters arising from that first CA. Because such runs used a matrix from any preliminary CA, just the resulting CA graphs are shown for such further analyses. Accordingly, Table 7.9 shows row results from the first pass of CA on the matrix of prepositions and pairs of meta-types from Figure 7.3. The first row lists the row labels, that is, the prepositions under analysis. Subsequent rows give F1 and F2 values, though results are sorted primarily on F1; columns appear, then, from the lowest F1 value to the highest. The last column gives the absolute ranges for factors F1 and F2:

Prep	of	with	in	for	on	like	as	about	by	to	R_{Fn}
F1	-538	-392	-377	-369	-299	-247	-225	-194	-27	852	1390
F2	-486	375	335	46	595	215	198	169	694	-74	1180

Table 7.9: CA results for prepositions from pass 1, sorted on F1

The prepositions 'of' and 'to', respectively at the extreme left and at the far right of Table 7.9, had relatively large values of F1, although the value for 'of' was negative, while 'to' showed the sole positive value. Between those extremes, prepositions had successively larger negative F1s, from left to right. Apart from 'of', F2 values had the opposite sign to corresponding F1 values.

In a similar way as for row results, Table 7.10 depicts column results from the CA of prepositions against meta-types, sorted by ascending F1. Note that column results showed a similar distribution to the corresponding row results:

Pair	tt	tm	mt	mm	at	am	ta	aa	ma	R_{Fn}
F1	-503	-420	-306	-118	-64	-6	552	1087	1179	1682
F2	-306	-39	104	321	521	708	-392	-104	-94	1100

Table 7.10: CA results for meta-types from pass 1, sorted on F1

The meta-type pairs 'tt', 'tm', and 'mt' at the extreme left of Table 7.10 had relatively large negative F1 values that gradually became less negative, before rising sharply to high positive values for 'aa' and 'ma' on the right side of the table. F2 values reflected similar variations in sign and magnitude, indicating coordinates for all four quadrants.

The graph of F1 and F2 from CA of prepositions against meta-types from Table 7.9 are superimposed in Figure 7.4. Points for prepositions appear as squares, while points for meta-types are dots. Scales

on the axes reflect coordinates taken from the CA software from Murtagh (2005). Note the marked separation of 'to' on the right of the X-axis origin:

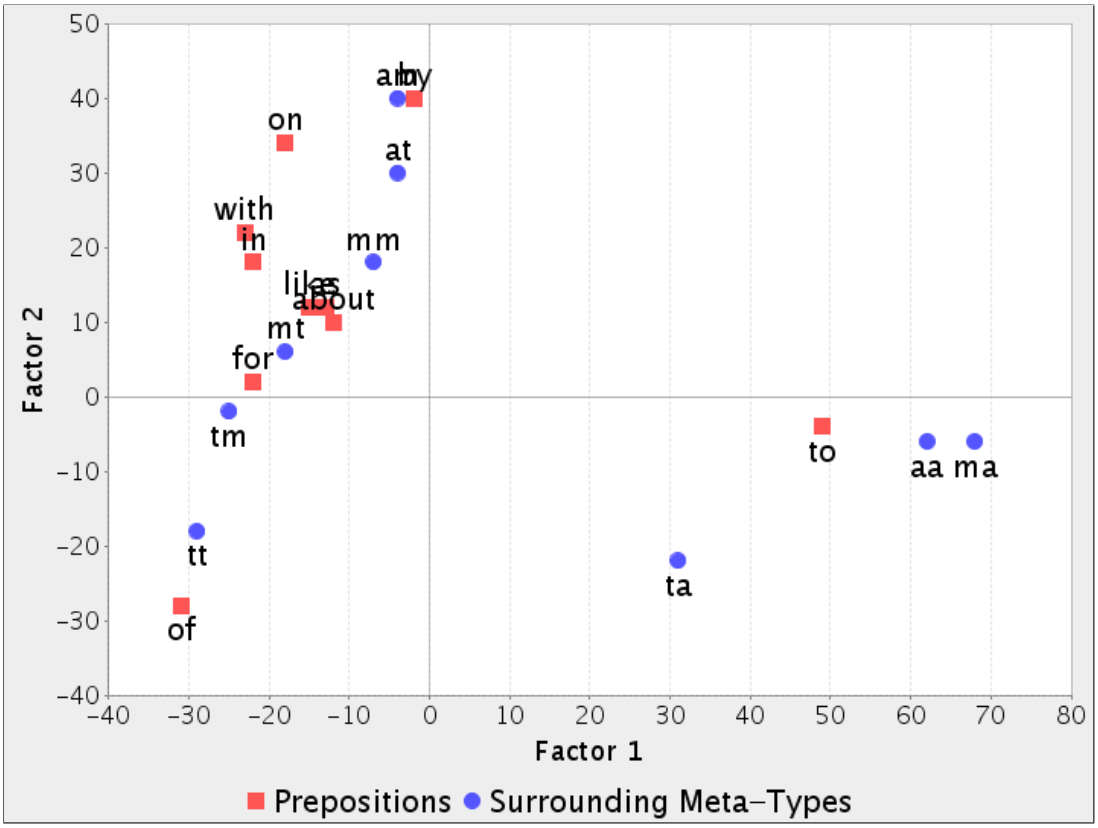

Figure 7.4: CA graph of F1 x F2 for pass 1

In addition to appearing to the right of the X-axis in Figure 7.4, the point for 'to' further inhabits the bottom quadrant on that side of the graph. Corresponding meta-types 'ta', 'aa' and 'ma' congregate in that same quadrant. Remaining prepositions and meta-types, in contrast, group together on the left-hand side of the graph, overlapping the upper and lower quadrants.

Discussion IVa of a contingency matrix using 2 senses difference

The graph for pass 1 exhibits obvious trends to a human viewer, such as the preposition 'to' being separated from remaining prepositions. A positive F1 meant that 'to' appeared to the right of the X-axis origin, while a large F1 value further forced that point towards the edge of the graph. In contrast to the X-axis, 'to' appears close to the origin on the Y axis, due to a relatively small F2. In fact, 'for' had a similar F2 value. Accordingly, 'to' and 'for' are barely separated on the Y-axis, while being far apart on the X-axis.

Apart from 'to', prepositions had negative F1s, and fell to the left of the X-axis origin. The word 'of' had large negative F1 and F2, so appears in the bottom left-hand corner. In contrast, a large positive F2 sent

'by' to the top of the graph. In fact, the result for 'by' was almost the opposite of that for 'to', in having a small negative F1 and a large positive F2, separating 'by' and 'to' into diagonally opposed quadrants. Medium F1 and F2 values put remaining prepositions nearer to both axis origins, though largely in the upper-left quadrant.

In comparison, meta-type pairs 'ta', 'aa' and 'ma' are clearly associated with 'to'. In fact, all of those four points had similar patterns of F1 and F2: a positive F1 value, and a negative F2. Due to that, those row and column points occupy the same quadrant. Further, those meta-type pairs for 'to' all ended in 'a'; as a result, ambiguous words immediately following 'to' will be treated as verbs. On the other hand, words coming directly before 'to' could take any meta-type, which does not help in deciding the best one. In practice, just ambiguous words that follow 'to' stand a chance of resolution.

On the left-hand side of the graph, 'tt' seems to correspond with 'of', as reflected by the relatively large negative values for both F1 and F2. None of the remaining column labels correspond so obviously with a specific row point. Given that difficulty for humans, machines face a serious challenge in identifying such correspondence. Before addressing an approach to that problem, though, results are presented for a supplementary CA that applied a stricter criterion of 3 to differences between senses for candidate POS.

Results IVb for a contingency matrix using 3 senses difference

The graph from Figure 7.5 shows that the effect of the criterion for determining predominant POS was having 3 or more senses than the closest competitor, rather than just 2:

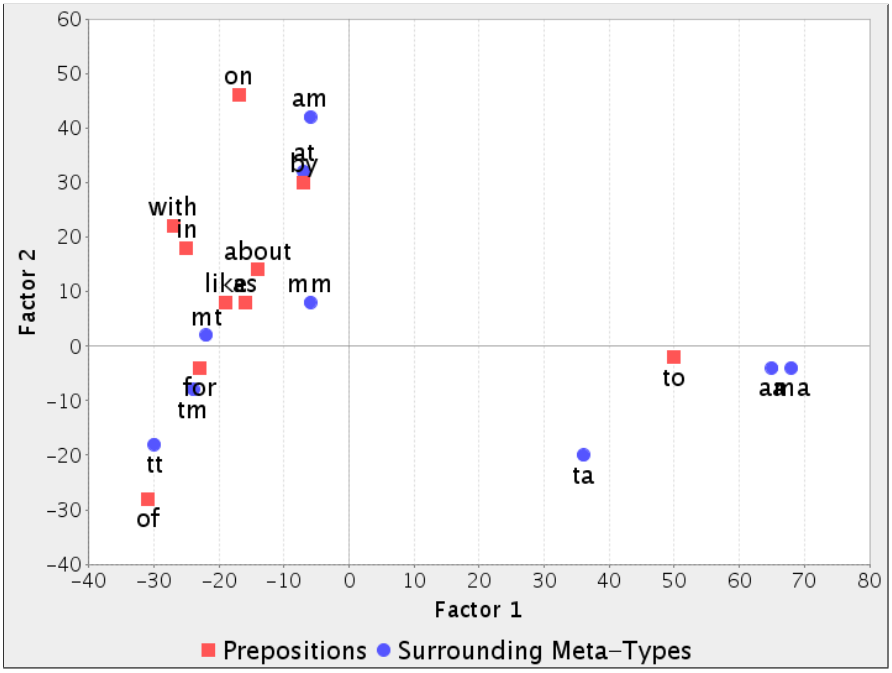

Figure 7.5: CA of F1 x F2 graph for pass 1, with WordNet senses difference = 3

Certain differences arose between the above graph and the one from a difference of 2 senses. Points for 'by' and 'for' moved towards the Y-axis origin, slightly south of their original positions. That was particularly pronounced for 'by'. As for the column points, 'aa' and 'ma' moved slightly closer together around 'to'.

Discussion IVb of a contingency matrix using 3 senses difference

The graph for that stricter criterion produced superimposed graphs that closely resembled those from the more lenient 2 senses difference. Despite small movements, points continue to be sharply split on the X-axis, with 'to' retaining identical meta-types as before. It appears, then, that a difference of 2 senses between competing POS adequately determines the most likely choice, for the purpose of this thesis. Having shown that to be the case, attention turns next to an experiment that identified outliers.

Identifying outliers

The initial phase of CA showed the strong influence of 'to', which skewed the entire graph. That preposition, along with corresponding column labels, was the sole occupant of the right-hand side of the graph, while remaining points bunched together on the left-hand side. That proved useful in identifying meta-type pairs corresponding strongly with 'to'; correspondence between remaining row and column points, though, was obscured.

Having noted that outlier 'to', removing it would force a new configuration of points; just because the meta-type pair 'ta', say, corresponded strongly with 'to' does not preclude it appearing around other prepositions. Removing just row outliers, then, leaves corresponding column points available to the next most attractive preposition.

To that end, requesting just two clusters from CA reveals any outlier. Remaining points form a cluster that might be marked 'others': they are of no interest, at this point. Which of those clusters was the outlier depended on the associated F1 values. In fact, F1 values were averaged across the two row clusters, and across the two column clusters, so as to reflect the overall inertia of any cluster. Sorting those clusters on average F1 aligned corresponding row and column clusters, as shown next.

Method V for identifying outliers

Row outliers were identified by means of CA for just two clusters; parallel runs were performed with the standard matrix, then with rows swapped with and columns in a second run. Subsequently, labels for prepositions from points in those clusters were matched with those from row results. In that way, prepositions were split into two sets corresponding to row clusters on the overlaid graphs. An average F1 value was calculated for each of those sets, and the process repeated for clusters from the second run. Further, averages for the two row clusters from each graph were divided by the absolute range of F1

values, which was calculated as follows. Row results were sorted on F1, forcing the lowest and highest results, respectively termed r_{min} and r_{max}, to opposite ends of the list. From those results, the absolute range of F1 values, R_{F1}, is calculated as:

$$R_{F1} = |r_{max}.F1 - r_{min}.F1|.$$

In a similar way, minimum and maximum results from the second run, with a revered matrix, gave the absolute F1 range for set I. Clusters were deemed outliers should the average F1 exceeded 50% of those absolute ranges from either run of CA.

Results V of identifying outliers

Table 7.11 shows clusters and factors from the first phase of CA, for 2 clusters; the left-hand side is for prepositions, and the right-hand side for meta-types from the second run. For prepositions, Cl_R shows the order in which clusters appeared from CA. The absolute range in F1 values comes next, headed R_{F1}, followed by the average F1 for any cluster, $\overline{F1}$. The absolute ratio of those two values comes immediately afterwards in the column headed $|\overline{F1}/R_{F1}|$. Labels of points, depicting prepositions in any cluster, are shown in the last column of the rows section. Similar headings depict parallel meta-type results from the second run of CA, in the right-hand part:

Cl_R	R_{F1}	$\overline{F1}$	$\lvert\overline{F1}/R_{F1}\rvert$	Prepositions	Cl_C	R_{F1}	$\overline{F1}$	$\lvert\overline{F1}/R_{F1}\rvert$	Meta-Types
0	1390	852	0.62	to	0	1180	939	0.80	ta, ma, aa
1		-296	0.21	of, by, on, in, with, for, about, as, like	1		-245	0.21	at, mm, am, tt, mt, tm

Table 7.11: Detecting outliers by means of CA results

The outlier 'to' appeared at index 0 in the sorted array of CA results, and had average F1 that constituted 62% of the overall F1 range, taking the value from column $|\overline{F1}/R_{F1}|$ as a percentage; that ratio was 21% for the cluster of remaining prepositions. Corresponding average F1 for column cluster 0 was 80% of the associated range, and 21% for cluster 1, coincidentally the same value as for cluster 1 of prepositions. In that way, corresponding meta-type outliers for 'to' were aligned at the same index, 0, in the sorted array.

Discussion V of identifying outliers

The results of automatically identifying outliers emulates humans in reading CA graphs: the point for 'to' was at an extreme edge, corresponding to similarly extreme meta-type pairs 'ta', 'aa' and 'ma'. In terms of the mechanics that underpin CA, a strong attractive force existed between those preposition and meta-type points. In turn, a strong mutual repulsion arose between outliers and remaining points from both sets, forcing them aside into a separate large cluster.

The skew in CA graphs introduced by outliers is both a drawback and an advantage. On the downside, bunching of remaining clusters in opposing quadrants makes interpreting such points difficult, for humans and machines alike. For that reason, subsequent phases of CA will be run after removing outlier prepositions; that will allow graphs to reconfigure once any over-strong forces are removed. On the other hand, outliers reflect strong correspondences that are more indicative than weaker ones that might emerge from subsequent analyses. Outliers, then, are welcomed when they arise, but are removed to reveal finer gradations in remaining points. A further use of outliers, though, is in determining optimum numbers of factors and clusters for those runs of CA, as shown next.

Determining optimum numbers of factors and clusters

The first pass of CA requested 3 factors, F1, F2 and F3. Upon completion, factors having low rates of inertia that accounted for relatively little variation were omitted from subsequent analyses. In a similar way, the minimum number of clusters for any analysis was 2. Large differences in F1 between CA row results suggested a need for extra clusters; to that end, individual F1 values were expressed as a proportion of the overall F1 range. Large differences between consecutive row results led to incrementing the number of extra clusters. Adding that extra to the default of 2 gave the optimum clusters for a subsequent analysis.

Method VI for determining optimum factors and clusters

Input comprised the rates of inertia for CA factors, which were retained after CA was complete. A count was incremented for each factor whose inertia explained 10% or more of the total inertia, to yield an optimum number of factors for a subsequent analysis. In a similar way, an optimal number of clusters arose from CA that requested 2 clusters. Consecutive row results were compared, and the absolute difference in F1 between adjacent results, d_{F1}, divided by the absolute F1 range, R_{F1}. In that way, differences in F1 that separated neighbouring results were expressed as a proportion of the overall range. The number of extra clusters was incremented should that proportion exceed 50%. Any resulting extra clusters were added to the default of 2; should none arise, that default provided the minimum for CA.

Results VI of determining optimum factors and clusters

Figure 7.6 summarises the rates of inertia for factors F1 to F3 from the first pass of CA. That output came from a method called `getFactors()`, which further decided whether to include successive factors:

```
getFactors()
  F1 rate = 0.6904   include in CA
  F2 rate = 0.2327   include in CA
  F3 rate = 0.0442   exclude from CA
```

Figure 7.6: Rates of inertia from CA of 10 prepositions

The F1 value from Figure 7.6 accounted for nearly 70% of the total inertia, with around 23% explained by F2. In contrast, F3 reflected a relatively tiny 4%, and was ignored. In a similar way, an optimum number of clusters arose from row results sorted on F1, as shown in Table 7.12. The first row of that table shows labels of points on the CA graph: the names of prepositions. Subsequent rows show F1 values, and those values as an absolute proportion of the F1 range. The value for 'to' exceeded the criterion of 50% of the range, and is highlighted in bold:

Prep	of	with	in	for	on	like	as	about	by	to	R_{Fn}
F1	-538	-392	-377	-369	-299	-247	-225	-194	-27	852	1390
$\lvert F1/R_{F1} \rvert$	0.39	0.28	0.27	0.27	0.22	0.18	0.16	0.14	0.02	**0.61**	-

Table 7.12: CA row results for pass 1, sorted on F1

The single highlighted value for 'to' in Table 7.12 meant adding just one cluster to the default of 2, giving 3 clusters in all. The next highest proportions were at the left-hand side of the table, where 'of', 'with', 'in' and 'for' yielded values ranging from 27% to 39%.

Discussion VI of determining optimum factors and clusters

In contrast to factors F1 and F2, then, factor F3 accounted for a very small proportion of the overall inertia within the CA space; for that reason, subsequent phases of CA use just 2 factors. Indeed, it is safe to do that because further factors will explain yet less inertia. The exact meaning of such factors, though, remains unclear, it can, in fact can be derived from optimised results, as will be seen from a later experiment. For now, note that large differences in F1 between consecutive rows indicated boundaries between clusters. Just one such boundary was detected; adding that to the default of 2 gave 3 as the best number of clusters. The next set of results reveals how that affected a subsequent run of CA.

A second pass of CA using an Optimum Number of Clusters

The initial pass of CA determined that 2 factors and 3 clusters would best suit a second pass. The matrix header was changed to reflect those values, and CA re-run. That produced clusters that a human might well discern. Two further automated processes were applied to the ensuing clusters. The first involved mapping clusters of prepositions to clusters of meta-types, to show correspondence between the two sets. That problem is overcome in a similar way as taken for outliers, which involved averaging and sorting results to reveal corresponding clusters. The second task faced here is in identifying what meta-types are associated with particular axes. First, though, to an experiment in mapping between clusters of prepositions and clusters of meta-types.

Results VII of mapping between row and column clusters

Table 7.13 presents mappings between the three clusters from pass two. Columns Cl_R and Cl_C give the indices of clusters in respective lists. Prepositions and meta-types were taken from labels of points in those clusters, and results sorted on the mean F1 for row clusters:

Cl_R	$\overline{F1}$	Prepositions	Cl_C	$\overline{F1}$	Meta-Types
0	-538	of	1	-410	tt, mt, tm
2	-266	by, on, in, with, for, about, as, like	0	-81	at, mm, am
1	852	to	2	939	ta, ma, aa

Table 7.13: Mappings between 3 clusters from pass 2 of CA.

The cluster at index 0 for the preposition 'of' to the left side of Table 7.13 aligned with meta-type cluster 1 from the right-hand side that comprised meta-types 'tt', 'mt', 'tm'. Of those. Remaining preposition clusters 1 and 2 respectively mapped to meta-type clusters 2 and 0. Note that F1 values between corresponding preposition and meta-type clusters, though different, are of similar magnitude. Take, for example, the meta-type F1 value of -410 from the first row of Table 7.13; that value was closer to of -538 for the preposition value from index 0 than it was to -266 from index 1.

Discussion VII of mapping between clusters

Abandoning factor F3 due to a relatively small amount of overall inertia meant that little remained to be explained after assessing F1 and F2. In other words, the graph for pass 1 from Figure 7.4 was very nearly a complete depiction of interactions between prepositions and meta-types. Subsequently, mappings arising from that CA reflected the positions of corresponding clusters in sets I and J, and allow machines to emulate humans in interpreting CA graphs. In fact, success in using mean F1 from results for prepositions was due partly to few clusters arising. Should more clusters emerge, taking F1 and F2 together might offer a more discerning approach; the relative signs of F1 *and* F2 would help to resolve cases where F1 alone could not determine appropriate mappings.

Corresponding clusters from parallel runs of CA, then, revealed affinities between given meta-types and prepositions. In that way, the preposition 'to' was strongly associated with meta-types dominated by 'a' for 'action', as was explained in the discussion for outliers. Further, using an optimum number of 3 clusters revealed similar strong affinity between 'of' and meta-types heavy in 't'. The preposition 'by', in contrast, had as many 'm' meta-types as 'a' and 't' combined. Having allowed machines to detect such patterns, attention turns next to using them in interpreting X and Y axes from CA graphs. That, in turn, will formalise such relationships between prepositions and things, actions, and modifiers, and ultimately raise important intensional knowledge about GRiST mind maps.

Method VIII for interpreting X and Y axes

Note that the following CA involved just the standard matrix of prepositions against meta-types, rather than comprising separate parallel runs. Labels from resulting column clusters for set J were split into individual meta-types, and frequencies of occurrence compiled for those discrete meta-types within any column cluster. A particular meta-type dominated if it accounted for 50% or more of any cluster's meta-types. In such cases, the F1 and F2 values of those CA results indicated the relevant axis.

Results VIII for interpreting X and Y axes

Table 7.14 presents CA results for 3 clusters that arose from the second pass of CA. The first group of columns show prepositions from the labels of points in row clusters. After those labels come the average F1 and F2 values for any row cluster. The second part deals with column clusters: first, the labels, then counts for individual meta-types within those composite values. The highest count for each meta-type appears in bold face, with the percentage of any entire cluster shown in the last column:

Row Clusters	$\overline{F1}$	$\overline{F2}$	Col Clusters	Meta-Types			
				a	m	t	%
of	-538	-410	tt, mt, tm	0	2	**4**	67
by, on, in, with, for, about, as, like	-266	939	at, mm, am	2	**3**	1	50
to	852	-81	ta, ma, aa	**4**	1	1	67

Table 7.14: Cluster mappings for phase 1

The last three columns of Table 7.14 show dominant meta-types, highlighted in bold type, that accounted for 50% or more of any cluster. Further, minimum and maximum values for mean row F1 of -538 and +852 put respective prepositions 'of' and 'to' at opposite ends of the X-axis. Corresponding meta-types for 'of' were dominated by 't' for 'thing', while 'to' strongly attracted points having 'a' for 'action'; in that way appeared X-axis meta-types of 't' at the negative extreme, and 'a' at the positive end.

In a similar way, minimum and maximum values of mean F2, for column clusters on the Y-axis, were -410 and +939. That revealed corresponding axis meta-types of 't' at the negative extreme occupied by 'of', and 'm' at the positive end that contained 'by'. Figure 7.7 shows output from `MidmapPOSAnalysis` that summarises the automated interpretation of axes:

```
      X-Axis                Y-Axis
--------------------  --------------------
-ve         +ve       -ve         +ve

thing       action    thing       modifier
```

Figure 7.7: Axis mappings from phase 1 of CA.

The final result, then, was that the negative extreme of the X-axis reflected things, while the positive end of that axis tended towards actions. In contrast, ascending the Y-axis depicted a transition from things to modifiers.

Discussion VIII for interpreting X and Y axes

Using the average F1 of clusters is reminiscent of the spatial mean, introduced in Chapter 6, that depicted entire clusters as single points. That was the case for all of the column clusters from this CA, and for the row cluster that contained 'by'; prepositions 'of' and 'to', though, formed singleton clusters having mean values equal to the associated F1 values themselves. The extreme positions of prepositions 'to' and 'of' on the X-axis suggested that axis to reflect a continuum between things and actions. In contrast, 'by' and 'of' at Y-axis extremes suggested a transition from things to modifiers. To emphasise those axis meta-types, superimposed graphs seen earlier in Figure 7.4 are reproduced here as Figure 7.8:

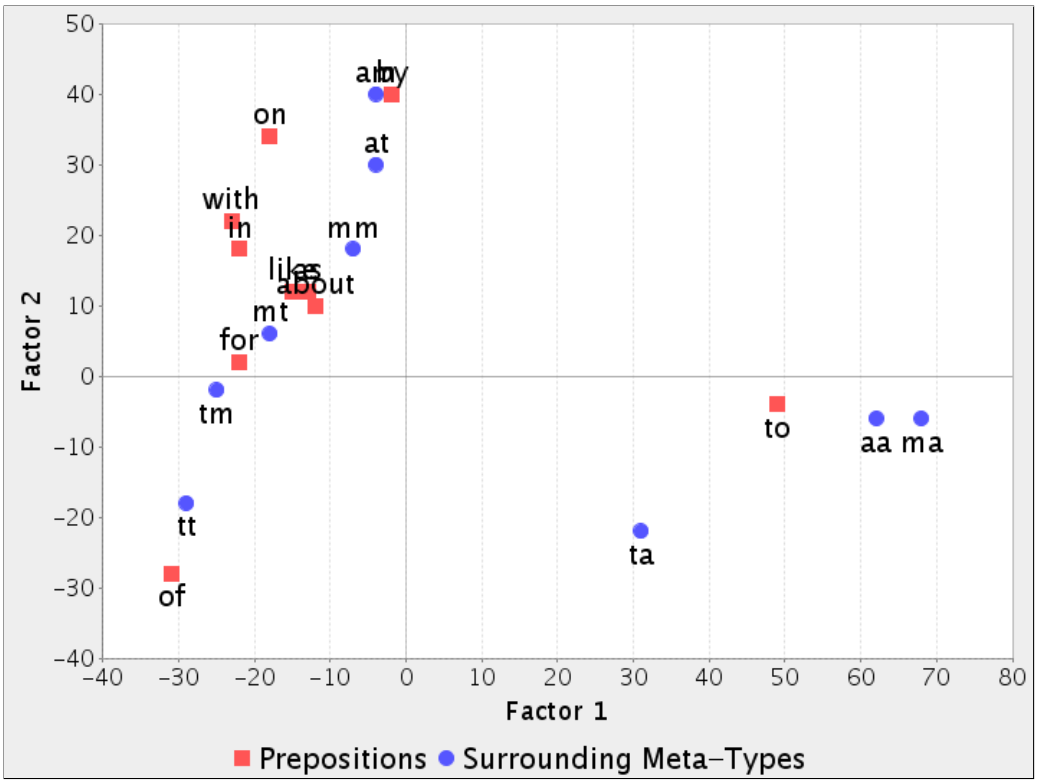

Figure 7.8: CA graph of F1 x F2 for pass 1 (reprise).

The bottom-right quadrant of Figure 7.8 shows 'of' at the 'thing' end of the X-axis, while 'to' lies towards the 'action' end. In contrast, prepositions 'of' and 'by' respectively occupy the 'thing' and 'modifier' ends of the Y-axis. Indeed, those results for automatically interpreting axes further validate the optimum three clusters derived from the preliminary CA. That number faithfully reflected the three main clusters in differing quadrants of the resulting graph.

Note further that symmetrical meta-type pairs ‘aa’, ‘mm’, and ‘tt’ appear in quadrants that express those types most strongly. In addition, symmetrical pairs ‘mt’ and ‘tm’ are separated across the X-axis, where they might be seen as belonging to the group around ‘for’, just above the Y origin. CA, rather, determined that they belonged with ‘of’; in that respect, a machine performed at least as well as a human, if not better. Symmetrical meta-type pairs ‘at’ and ‘ta’ are similarly separated, though diagonally across bottom-right and top-left quadrants. That goes in addition for ‘ma’ and ‘am’, although that latter point is partially obscured by the preposition ‘by’.

Results from CA, then, produced valuable intensional knowledge for the emerging information base of GRiST mind maps. Corresponding row and column clusters, along with the derived axis extremes, will shortly be seen to determine meta-types for ambiguous or missing words from WordNet. That will be inhibited, though, by the strong influence of ‘to’ that skewed the graph, and made remaining clusters less obvious. In order to develop a better model of the left-hand side of the CA graph, for all prepositions but ‘to’, that involves a subsequent pass of CA, after having removed that outlier row.

7.7 CA on Unambiguous Trigrams: Pass 2

The first phase of CA identified the preposition ‘to’ as outlier that was, in addition, a singleton cluster that contained no further points. Although singletons are not necessarily outriders on a graph, they are sufficiently removed as to contrast sharply with remaining clusters. Having noted such strong correspondences, outlier row clusters are removed in order to allow any released meta-types to reconfigure around the next most strongly attracting preposition.

Indeed, the next most extreme outlier from any pass is likely to be the one removed on the following pass. In that way, graded correspondences arise: the later they appear around any preposition the weaker the attraction between them. All the same, useful trigrams might arise that were not available from earlier passes, as will be seen from experiments presented next.

Method V for subsequent phases of CA

The `MidmapPOSAnalysis` class identified outliers for each phase. After noting any corresponding column points, matrix rows for outliers were removed before the succeeding phase, though remaining observations were untouched. In all, 5 phases of CA were performed, in paired runs: the first run in any phase identified optimum numbers of clusters for a second pass, which re-ran CA with an updated matrix header. That process was repeated for successive row outliers, gradually reduced the number of prepositions, while leaving intact all columns of meta-type pairs.

Results for all phases of CA

Figure 7.9 summarises the clusters and outliers from phases 1 to 5 of CA:

```
pass 1:  clusters = 3, outlier = [to] --> [ta, ma, aa]
pass 2:  clusters = 4, outlier = [of] --> [tt, ta, aa, mt, tm]
pass 3:  clusters = 3, outlier = [as] --> [ma, tm, mm, ta]
pass 4:  clusters = 3, outlier = [by, about, like] --> [am, at, ma, aa]
pass 5:  clusters = 4, outlier = [on] --> [am, at, aa]
```

Figure 7.9: Summary of clusters and outliers from CA phases 1 - 5.

The number of clusters varied between 3 and 4, and an outlier was detected in all phases. By phase 5, just four prepositions remained. Any fewer rows would stand no chance of filling the quadrants of a CA graph. For that reason, processing stopped at phase 5.

The composition of clusters changed from one phase to the next. In phase 2, the outlier 'of' paired up with meta-type pair 'aa', once that pair was free of the strong attraction exerted by 'to'. For phase 3, the preposition 'as' corresponded strongly with modifiers: 3 out of 4 pairs contained the meta-type 'm'. The symmetrical pair 'mm' further bound 'as' to modifiers. Phase 4 gave an outlier that comprised several points; note, though, that outliers for all other phases were singletons. After the clusters themselves, the other main output from CA comprised axis meta-types. Figure 7.10 shows how axes were interpreted in the five phases of CA:

	X-Axis		Y-Axis	
	-ve	+ve	-ve	+ve
pass 1:	thing	action	thing	modifier
pass 2:	thing	action, modifier	modifier	action
pass 3:	modifier	action	action	thing
pass 4:	modifier	action	action	modifier
pass 5:	modifier	action	modifier	thing

Figure 7.10: Axis mappings from CA phases 1 - 5.

Phase 2, then, gave meta-types of action *and* modifier for the positive X-axis. That was in opposition to 'things' at the negative end. Subsequent phases all gave single axis meta-types. Note, though, that the types for the X-axis in phase 4 are reversed for the Y-axis.

Results from Phase 2 of CA

Figure 7.11 shows the overlaid plots from phase 2 of CA, having removed the outlier preposition 'to' that skewed the first, full CA:

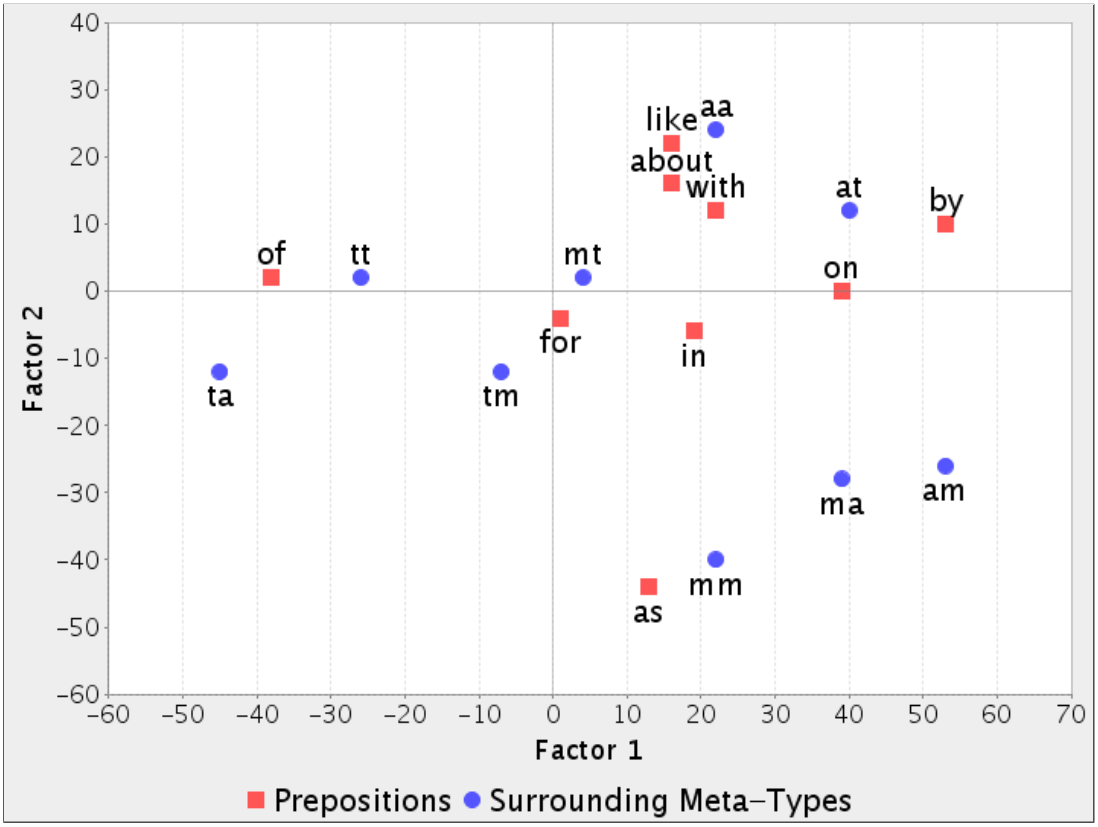

Figure 7.11: CA graph of F1 x F2 for phase 2

Removing the outrider 'to' from the matrix had a dramatic effect. The point for 'of' moved away from the bottom of the phase 1 graph, ending up slightly above the Y-axis origin. In addition, the preposition 'as' moved diagonally downwards, from the top-left quadrant to the bottom-right. Respective corresponding pairs 'tt' and 'mm' underwent a similar transformation. Remaining row and column points largely moved sideways, and were more evenly distributed than when the full CA matrix was used.

The singleton cluster for 'as' formed a distinct cluster in the second phase, rather than being amalgamated with various other prepositions in phase one. In a similar way, 'for' and 'in' seem to be associated with pairs 'mt' and 'mm', set apart from other points. To refute or confirm those associations, Table 7.15 (overleaf) presents mappings from the second phase of CA. In the same way as for the first phase, results were sorted on the mean F1 for row clusters, of which four arise after removing 'to':

Cl_R	$\overline{F1}$	Prepositions	Cl_C	$\overline{F1}$	Meta-Types
1	-520	of	1	-488	tt, ta
2	145	for, in	0	90	aa, mt, tm
0	187	as	2	531	ma, mm, am
0	410	with, about, like, by, on	2	559	at

Table 7.15: Mappings between 4 clusters from the second phase of CA.

Those results show that column points 'mt' and 'tm' did indeed correspond with 'for'. In addition, that preposition attracted the pair 'aa' that previously corresponded with 'to'; remaining meta-types 'ma' and 'ta' from that cluster respectively congregated around 'as' and 'of'.

Discussion of Phase 2 of CA

Removing row outliers encouraged new configurations of any associated column points. In all but phase 5 of CA, outliers were singleton clusters, which obviated the need to view such clusters as a spatial mean. Specific correspondence between single prepositions and meta-types, though, were less clear in a single, all-inclusive run. In contrast, successive phases of CA forced meta-type pairs to join new clusters. At each phase, the attraction between row and newly attracted column points was less than in preceding phases. That gave a graded analysis that will find use shortly in resolving ambiguous words from GRiST mind maps.

Corresponding row and column results helped to interpret graphs from CA. A human might have associated the meta-type pair 'aa' pair with 'like', 'about' and 'with'. In fact, the automated process assigned that point to the cluster comprising 'for' and 'in'. That recalls a similar situation between points for the prepositions 'for' and 'of' in phase 1, where a machine performed at least as well as might humans.

7.8 Using CA Results to Resolve Ambiguous Trigrams

This section applies the results from CA to resolving ambiguous trigrams. Two forms of ambiguity arose, in fact: the first type concerned words being absent from WordNet, while the second type involved POS having similar numbers of senses. Those forms of ambiguity were, in fact, handled differently.

The first type of ambiguity arose due to words missing from WordNet; because meta-types could not be assigned, corresponding parts of trigrams were left unfilled. In those cases, meta-types came from the axes of CA graphs. Locating points on the graph involved factors F1 and F2, which dictated respective positions on the X and Y axes. The most extreme of the X and Y coordinates suggested a meta-type, depending on the signs of those coordinates. Figure 7.12 reviews the axis meta-types from the first pass of CA:

X-Axis		Y-Axis	
-ve	+ve	-ve	+ve
thing	action	thing	modifier

Figure 7.12: Axis mappings from pass 1 of CA (reprise).

Applying axis meta-types, then, involves using F1 and F2 values from retained CA results. In that way, the locations of prepositions on any graph suggest meta-types for ambiguous parts of triples; the exact meta-type depends on which quadrant of a graph holds the associated preposition.

Conversely, words that did exist in WordNet allowed a more detailed analysis by way of reliable triples from Section 7.3. Those triples were further transformed into trigrams of the form {*meta-type*$_1$, *preposition, meta-type*$_2$}, and the same done for ambiguous triples from Section 7.4. While a sole trigram represented any unambiguous triple, several were needed to express ambiguous triples, in order to reflect permutations of particular meta-types.

The challenge, then, was to select the most appropriate combination, in context of the preposition in any specific trigram. To that end, unambiguous trigrams showed the meta-types commonly found with particular prepositions, which was reflected in the clusters of meta-types found by CA. Those CA clusters guided the selection of most likely meta-types, in the context of a particular preposition. Points nearer to the edge of any graph were judged to reflect the corresponding axis meta-type. That was assessed by the ratio of such F1 and F2 values to the overall range for respective sets I and J.

Method I for words unknown to WordNet

Triples containing unknown words were selected from the list created in Section 7.4; CA results for the preposition in any given triple were retrieved from tables retained from analyses. The row result for any specific preposition bore F1 and F2 values that dictated respective positions on the X and Y axes. To that end, the four quadrants of a CA graph were reflected in the signs of X and Y coordinates, against which particular F1 and F2 values were compared.

The row result corresponding to any particular preposition was retrieved, and its F1 value divided by the total F1 range R_{F1}. That was repeated for F2 using the F2 range R_{F2}. Further, signs on F1 and F2 values showed to which sides of the X and Y origins points fell. Generally, the X-axis meta-type at the corresponding F1 extreme was taken; exceptions were Y-axis outliers having relatively large F2. In fact, the Y-axis meta-type was used when the ratio $^{F2}/_{R_{F2}}$ was three or more times that of $^{F1}/_{R_{F1}}$.

Results I for words unknown to WordNet

Figure 7.13 shows partial output from the `MidmapPOSAnalysis` class, for three unknown words. The first line in each group shows a triple that contained such a word, which is highlighted by asterisks. Line two gives the CA row result for the preposition in the triple. That result holds F1 and F2, the corresponding ranges, and the resulting ratios. The last line shows which side of the X or Y axes yielded a meta-type, along with the accepted interpretation:

```
unknown word: history OF *impulsivity*
row result for "of": F1 = |-538/1390| = 0.39, F2 = |-486/1100| = 0.44
use negative X-axis F1: "impulsivity" = thing

unknown word: plan TO *reconnect*
row result for "to": F1 = |852/1390| = 0.61, F2 = |-74/1100| = 0.07
use positive X-axis F1: "reconnect" = action

unknown word: *disinhibited* BY mental
row result for "by": F1 = |-27/1390| = 0.02, F2 = |694/1100| = 0.63
use positive Y-axis meta-type: "disinhibited" = modifier
```

Figure 7.13: Treatment of unknown words by the Java class `MidmapPOSAnalysis`.

The first group from Figure 7.13 was for the word 'impulsivity' that did not exist in WordNet. The row result for the preposition 'of' gave respective F1 and F2 ratios of 0.39 and 0.44; those ratios were barely different, which led to taking a meta-type of 't' from the negative end of the X-axis.

The second entry from Figure 7.13, for 'reconnect', had the preposition 'to'; the corresponding row result held respective F1 and F2 ratios of 0.61 and 0.07; the large F1 ratio on that result suggested a meta-type of 'a', from the positive end of the X-axis. In contrast, the last triple from Figure 7.13

for 'disinhibited' applied the CA row result for 'by', which gave ratios of 0.02 and 0.63 for F1 and F2 respectively. As a consequence, the meta-type 'm' was taken from the positive end of the Y-axis.

Meta-types for words unknown to WordNet, then, arose from referring to axes on CA graphs; the next set of results from that process deals with further omissions from WordNet: 'carer' and 'carers'.

Table 7.16 presents the results for assigning axis meta-types to 'carer' and 'carers'. Note that the first entry shows the word 'kind' in brackets, after 'any', to provide context for that phrase:

Word 1	Prep	Word 2	X-Axis		Y-Axis		Use	M-T
			F1	$F1/R_{F1}$	F2	$F2/R_{F2}$		
carers	of	any (kind)	-538	0.39	-486	0.44	–ve X-axis	thing
collaboration	with	*carer*	-392	0.28	375	0.34		

Table 7.16: Disambiguation of 'carer' and 'carers' that were unknown to WordNet.

Words from Table 7.16 appeared around two different prepositions: 'of' and 'with'. Row results for both of those prepositions had relatively large negative F1 values, reflecting points well towards to the left side of the graph. Signs for the F2 values, while of similar magnitude to F1, had different signs. That reflects the point for 'with' above the X-origin, and below it for 'of'. Together, those F1 and F2 values specify the bottom-left quadrant for 'of', and the top-left quadrant for 'with'. In neither case did the F2 ratio sufficiently exceed that for F1, so meta-types of 't' for 'thing' were taken from the X-axis.

Considering axis meta-type further helped with the unknown word 'rq', which co-occurred with the proposition 'as'; the location of that proposition on the graph correctly suggested the meta-type 't' for 'rq'. In a similar way, 'cjs' was deemed a noun by the corresponding preposition 'in'. The main helpful preposition, though, was 'of', which led to treating further abbreviations 'cnr' and "cmht' as nouns. In a similar way, non-words such as 'derealisation', 'affectivity' and 'parasuicide' were deemed nouns by applying axis mappings for 'of'. The word 'avoidant', though, was seen as a noun in that way, when in fact it is a modifier.

As opposed to nouns, words such as 'disinhibit' and 'sh', were interpreted as verbs by means of the preposition 'to'. Modifiers, on the other hand, included the non-word 'disinhibited' that was correctly classified in relation to the preposition 'by'. The related word 'disinhibiting', though, was further treated as a modifier. That, then, concludes results for words having no entry in WordNet; a discussion of those results now follows.

Discussion I for words unknown to WordNet

Words that failed to find entries in WordNet, then, were assessed in relation to axes from associated CA graphs. From that analysis arose various abbreviations, such as 'rq' for 'risk quotient', 'sh' for 'self harm',

and 'cjs' for the Criminal Justice System, all being correctly deemed nouns. In addition, medical terms such as 'parasuicide' were further seen as nouns. In fact, interpreting 'sh' as a noun arose from the node `[acts of sh]`. In contrast, the node `[now started to sh again]` used 'sh' as a verb, which was correctly determined by correspondence with the preposition 'to'. A further appropriate verb arising from 'to' was the word 'disinhibit'. In fact, unknown words appearing with 'to' were invariably seen as actions; that proved wrong for the words 'insightless', 'prequel' and 'rejector'.

Words that were unknown to WordNet, though, were treated largely as nouns, which in most cases reflected intended meanings. Exceptions were 'avoidant' and 'disinhibiting', which were actually modifiers that were falsely interpreted by reference to 'of', the sole occupant of the bottom-left quadrant. Because the X-axis meta-type 't' corresponded strongly with 'of', those words were treated as nouns.

An interesting exception concerned the word 'disinhibited' that was determined to be a modifier, due to the associated preposition 'by' having a strong F2 component. That correctly gave the meta-type 'm' from the positive extreme of the Y-axis. The results presented here, then, show CA to resolve POS for words that were absent from WordNet. A slight problem, though, arose from strong correspondence between 'to' and verbs, leading to classifying several 'things' as 'actions'. The problem is inherent in CA, in that it measures overall trends in data; occasionally, such patterns can be misleading. All the same, ambiguity was resolved in the sense that words absent from WordNet were assigned POS; attention turns now to words that are found in WordNet, but have similar numbers of senses.

Method II for ambiguous words in WordNet

This step compares ambiguous trigrams from Section 7.4 with unambiguous ones from Section 7.3. The latter list provided reliable trigrams of the form {*meta-type*$_1$*, preposition, meta-type*$_2$}. Now, novel trigrams were created that held constant any prepositions and unambiguous meta-types; in that way, each novel trigram represented a candidate meta-type for any particular ambiguous word. The best of those novel trigrams was determined by consulting CA clusters from unambiguous ones.

Results II for ambiguous words in WordNet

Before presenting details of how CA resolving ambiguity, Table 7.17 presents just a single example that demonstrates how subsequent tables should be interpreted. In fact, that table comprises four parts. The first part has two columns: the one headed Word 1 holds words that preceded particular prepositions, and the associated M-T column holds one or more meta-types. Unambiguous words will have a single, reliable meta-type, whereas ambiguous words have several candidate meta-types shown as a comma-separated list. The second part of the table has a sole column headed Prep to indicate the preposition from any triple. Following that, the third part of the table comprises two columns for the words following any preposition, and any corresponding meta-types reported by WordNet.

The fourth part of Table 7.17 comprises four columns that concern meta-type clusters corresponding with any given preposition. The first of those columns, headed Meta-Type Permutations, shows possible combinations of meta-types for Word 1 with those for Word 2, with the pair that was ultimately selected shown in bold face. That selected pair arises from comparison with labels from the column cluster, which appears in the last column. Between those permutations and clusters, the column headed N shows the pass from which any cluster emerged, while column S shows whether the corresponding row cluster was a singleton, containing just a sole preposition. Details of a single trigram, then, occupy one line in the table as for the given example of 'drugs' that WordNet reported might be a verb or a noun, respectively shown by meta-types 'a' and 't':

Word 1	M-T	Prep	Word 2	M-T	Meta-Type Permutations	Column Cluster		
						N	S	Labels
drugs	*a*, ***t***	in	significant	m	*a*m, ***tm***	2	N	aa, mt, **tm**

Table 7.17: Example of disambiguation by means of CA.

The column headed Word 1, then, holds the ambiguous word 'drugs' that might be a verb or a noun; accordingly, 'drugs' appears in italics to reflect that a decision is required. In a similar way, associated meta-types under the M-T column lists candidate meta-types 'a' for 'action' and 't' for 'thing'. Further highlighting in bold type notes that the meta-type 't' was the one ultimately selected.

The second part of the table, comprising just the Prep column, holds the preposition for that triple; in this case, the word 'drugs' preceded the preposition 'in'. The third part lists words and meta-types for words that followed prepositions, and are formatted in the same way as were the first two columns for words preceding any preposition. In the current example, the preposition 'in' was followed by 'significant', which was an unambiguous modifier.

Finally, the fourth group of columns reveal details of the column cluster corresponding to 'in'. The column headed Meta-Type Permutations shows possible combinations of meta-types between entries from columns Word 1 and Word 2. In that way, candidates of 'a' and 't' for 'drugs' combined with the meta-type 'm' from 'significant' to give permutations of 'am' and 'tm', which are shown in italics to note that a decision is required. That decision depends on labels from the last column , headed Column Cluster, where the candidate pair 'tm' was indeed found. The meta-type preceding 'in' from that label is 't', which appears in bold to reflect the POS of the ambiguous word 'drugs'.

The column headed N shows that the meta-type 't' was chosen due to a cluster from pass 2 of CA, after having removed the row for 'to'. The column headed Singleton, though, shows that 'in' shared a cluster with other prepositions. Having described the format of tables that follow, here are the remaining results for disambiguating the word 'drugs'.

Disambiguating 'drugs'

In preparation for presenting CA results related to 'drugs', Table 7.18 reviews research in WordNet from which ambiguity arose. Examples are given of nodes from GRiST mind maps. In those nodes, 'drugs' could be either a noun or a verb; sense counts of 1 and 2, though, make that distinction hard to discern:

Word	POS Senses					Prep	Examples from GRiST mind maps
	Aj	Av	N	V	Δ_s		
drugs	-	-	1	2	1	as	not [same **as** *drugs*]
						in	alcohol and [*drugs* **in** significant] amounts
						of	their [*use* **of** *drugs*]
						on	rq : number one [*high* **on** *drugs*]
						to	individuals [response **to** *drugs*]

Table 7.18: Review of the ambiguous word 'drugs'.

The word 'drugs', then, could either a noun or a verb. Table 7.19 shows how CA resolved that ambiguity, using corresponding clusters of prepositions and meta-types. Note that although the earlier example for 'drugs' and 'significant' is not included, it was the sole instance of that ambiguous word following a preposition. Further note that the result for 'of' has a further ambiguous word, 'use', as does 'on' with the word ambiguous word 'high':

Word 1	M-T	Prep	Word 2	M-T	Meta-Type Permutations	Column Cluster		
						N	S	Labels
same	m	as	*drugs*	***a***, *t*	**m*a***, m*t*	2	Y	**ma**, mm, am
response	t	to			**t*a***, t*t*	1	Y	**ta**, ma, aa
use	*a*, ***t***	of		*a*, ***t***	*aa*, *at*, *ta*, ***tt***	1	Y	**tt**, mt, tm
high	***m***, *t*	on			*ma*, ***mt***, *ta*, *tt*	3	N	aa, **mt**, tt

Table 7.19: Disambiguation of 'drugs' by means of CA.

Column N from Table 7.19 shows that relevant clusters arose from passes 1 to 3 inclusive; the Singleton column further shows row clusters for 'of' and 'to' as singletons from pass 1, as was the preposition 'as' in pass 2. In contrast, 'on' from pass 3 shared a cluster with additional prepositions. The M-T column shows that in the first two results, 'drugs' was interpreted as a verb, while remaining results gave nouns.

The first result from Table 7.19 showed the modifier 'same' preceding the preposition 'as'. The sole matching permutation demanded that 'drugs' be a verb, to form 'ma'; that decision, though, was incorrect: 'drugs' is a noun in all of those examples. That result for 'as' was both a singleton and an outlier in pass 2 of CA; the second result, for 'to', was further the outlier from pass 1. With 'response' being reliably deemed a noun by WordNet, the possible permutations were 'ta' and 'tt'. Just the former, though, existed in the corresponding column cluster, which led to wrongly interpreting 'drugs' as a verb.

Remaining examples, though, see 'drugs' as a noun due to the prepositions 'of' and 'on'. Further, the remaining word on those triples were ambiguous: 'use' might have been a verb or a noun, while 'high' acts both as an adjective and as a noun. The corresponding meta-type cluster for 'of' comprised pairs made from just 'a' and 't'; the sole matching column meta-type was 'tt' that interpreted both 'use' and 'drugs' as nouns.

Candidate meta-type pairs for 'high' and 'drugs', though, matched two corresponding permutations: 'mt' and 'tt'. While that made 'drugs' a noun, 'high' remained ambiguous. In fact, 'by' appeared at the 'modifier' extreme of the Y-axis of the graph from pass 1 of CA, and remained above the Y-origin after removing 'to'. That was due to the strong F2 component for 'by': F2 was 678, compared with -299 for F1. At over twice the size, F2 indicated that the Y-axis should be used. Because the positive end of that axis represented modifiers, the pair 'mt' was preferred over 'tt'; that made 'high' as a modifier, and 'drugs' a noun. Having shown CA in disambiguating the word 'drugs', attention turns to a further important GRiST concept: 'abuse'.

Disambiguating 'abuse'

The method for refining 'abuse' resembled that applied to 'drugs', and will not be repeated. Further, no specific prepositions or corresponding meta-type clusters are shown; rather, just candidate meta-types are shown for ambiguous words. Examples of nodes from GRiST mind maps show triples enclosed in square brackets, with ambiguous words in italics, and prepositions in bold type:

Word 1	M-T	Example from GRiST
abused	***a***,*m*	[*abused* **as** children]...ending in psychiatric system
	a,***m***	[*abused* **by** friends]
abuse	*a*,***t***	often people with a [history **of** *abuse*]
		[history **of** *abuse*] potentially important factor
		2 kids under 5...and a [history **of** *abuse*]
		tranquillisers as a [method **of** *abuse*]

Table 7.20: Disambiguation of 'abuse' by means of CA.

The first entry of Table 7.20 shows 'abused' to be interpreted as a verb, though it might have been a modifier. Indeed, that was the preferred meta-type for the second entry, 'by'. The latter part of that table shows 'abuse' as a noun due to the preposition 'of'. Three of those results involve the reliable noun 'history' in various contexts; the remaining result has the similarly unambiguous noun 'method'.

Discussion II for ambiguous words in WordNet

The ambiguous word 'drugs', then, arose from just 1 sense as a noun, and 2 as a verb; that ambiguity was resolved by trigrams comprising the prepositions 'as', 'in', 'of', 'on' and 'to'. Indeed, 'of' proved

particularly accurate in such cases; a strong correspondence with 'things' gave that interpretation for words that truly were nouns. In a similar way, the word 'in' justified treating 'drugs' as a noun, when it preceded that preposition. The modifier 'significant' following 'in' led to that decision; in fact, 'in' tended to expect a preceding noun and a following modifier; given that unambiguous modifier 'significant', 'drugs' was most likely a noun in that context.

Further justification for 'drugs' as a noun came from trigrams bearing 'on', which corresponded with preceding modifiers and trailing nouns. In that way, 'on' being followed by 'drugs' meant taking that ambiguous word as a noun, rather than as a verb. In the trigram from that case, though, both words were ambiguous: the word 'high' might further have been a modifier, in that a person might be described as 'high', or a noun, as in the 'high' that drugs give users. In fact, 'high on drugs' describes a state; because the preposition 'on' corresponded with preceding modifiers, that was correctly taken as the preferred POS for 'high'.

Correct choices, then, were made for both of the ambiguous words from that triple. In fact, the decision for 'high' relied on a meta-type cluster from pass 3 of CA; had outliers not been successively removed, and CA re-run, stronger attractions elsewhere in any graph would have masked that correspondence. Further, the Y-axis meta-type of 'm' discouraged selecting 'use' as a verb should it appear before 'on'. In that case, that decision arose from F2 rather than F1, which well demonstrated the utility of having machines 'read' superimposed graphs, and interpret X and Y axes.

In contrast to reliable patterns revealed for 'of', trigrams for 'on' and 'to' were somewhat misleading; 'drugs' was incorrectly seen as a verb, rather than as a noun. That was because of strong correspondence between 'to' and actions; in fact, any word that followed 'to' would be seen as a verb. Conversely, the relatively central position of 'as' on the CA graph from pass 1 suggested relatively small correspondence with any particular meta-types. On removing the outlier 'to, the point for 'on' moved sharply towards the bottom of the graph from pass 2. The resulting singleton cluster, though, was strongly associated with actions and modifiers, but contained no instances of the meta-type 't' for 'thing', which precluded taking 'drugs' as a noun.

The last set of results for ambiguous entries in WordNet concerned the key concept of 'abuse', which is of great importance in GRiST mind maps. Two forms of that concept, 'abuse' and 'abused', met with varying degrees of success in determining the best POS. Of those words, 'abused' might be a verb or a modifier; although the verb was chosen, 'abused' was really a modifier. So similar are those two usages, though, as to make the distinction pedantic. Indeed, those phrases might be re-written in a standard form, using parentheses to denote unknowns; that might yield '(They were) abused as children' and 'friends abused (them)'. That the verb was given precedence in fact reflects a more useful relationship,

in terms of a subject, a verb, and an object.

The remaining word 'abuse' was correctly deemed a noun in all cases, by means of strong correspondence between 'of' and 'things'. In that way, the information base of GRiST mind maps receives important intensional knowledge about roles that 'abuse' plays in mind map nodes as a concrete concept, rather than a participle of the verb 'to abuse'. The relationship between noun and verb forms, though, will allow machines to infer such relationships, which in turn demands further analysis in order to find subjects and objects for such actions.

Supplementary Method for Bigrams arising from Trigrams

Having analysed triples of the form $\{word_1,\ preposition,\ word_2\}$ from nodes in GRiST mind maps, a further trawl of those mind maps revealed novel combinations of words involved. Rather than triples, though, it was bigrams of the form $\{word_1,\ word_2\}$ and $\{word_2,\ word_1\}$ that were sought. In the first type of bigram, words appearing immediately before or after any preposition were researched as contiguous words, that is, with the intervening preposition removed. In the latter type, though, words were further reversed; in that way, the word preceding any preposition, say, became the second word of a novel bigram.

Words from mind map nodes were reduced to an array by using the `split()` method, as in earlier experiments. Given a preposition at index i of any resulting array, indices $i-1$ and $i+1$ identified words to either side of that preposition. Bigrams of the first type, then, comprised words at respective indices $i-1$ and $i+1$, whereas in the latter, that order was $i+1$ and $i-1$. In addition, words conflated by stems were included in that process.

Supplementary Results for Bigrams arising from Trigrams

Table 7.21 shows nodes having words originally separated by the prepositions 'for', 'in', 'to', and 'with'; such nodes appear to the left-hand side of the table. Removing those prepositions yielded novel bigrams that maintained words in their original order. Prepositions themselves appear in the middle column, while examples of nodes bearing such novel bigrams are shown on the right. Bigrams and triples appear in square brackets, and any prepositions highlighted by italics. Longer nodes have been edited in order to save space, although important parts remain intact:

Words Separated by Prepositions	Prep	Contiguous Words in Identical Order
rq : thats [cause *for* concern]	for	[causes concern]
[situations *in* which] drinking	in	[situation which] prompted... current problems
lack of [access *to* services]	to	[accessing services]
brought into [contact *with* services]	with	avoid [contact services]

Table 7.21: Supplementary Results of Bigrams arising from Trigrams.

Due to including stemmed words, the first entry from Table 7.21 shows the unambiguous noun 'cause' transformed into a verb on removing the preposition 'for'; in fact, 'cause' had 5 noun senses but just 2 as a verb. A similar situation arose in the third entry, for the preposition 'to'; the noun 'access' in that case became the verb 'accessing'. In contrast, words associated with 'in' retained their original POS, although the intended meaning of last novel pair, on the right-hand side of the last entry, is unclear.

In a similar format to Table 7.21, results in Table 7.22 depict nodes containing words that were separated by 'of', which was removed to yield novel bigrams; the redundant column for prepositions has been removed. The right-hand column, then, shows nodes that contained words contiguously, but with their order in the original node reversed:

Words Separated by 'Of'	Contiguous Words in Reverse Order
[pattern *of* behaviour]	change in [behaviour patterns]
uniforms..[figures *of* authority]	relationship to [authority figures]
[dynamics *of* family] is very complex	[family dynamics]
increasing the [risk *of* suicide]	high [suicide risk]

Table 7.22: Supplementary Results of Reversed Bigrams arising from Trigrams.

All of the results from Table 7.22, then, contained the word 'of' in previously identified nodes. Further nodes contained words from those triples contiguously, but in reverse order; all such words were, in fact, reliable nouns.

Supplementary Discussion of Bigrams arising from Trigrams

The word 'causes' that should be interpreted as a verb in the phrase 'causes concern' is, in fact, more likely to be a noun; because WordNet reports three more noun than verb senses, it would be deemed a reliable 'thing' by the criterion used here. Indeed, that leads WordNet to treat the suffix '-s' as a plural, rather than as the third person singular of 'to cause'. Although demanding that any dominant POS have two or more senses than remaining candidates was largely successful, that measure of familiarity clearly needs refinement.

With respect to the preposition 'in', the phrase 'situations in which' denotes a context for ' drinking'; in contrast, the phrase 'situations which' suggests a more concrete relationship, as the immediate subject of 'prompted'. The difficulty raised in that example was reflected in results from CA on meta-types, where 'in' appeared well away from any axis extreme, indicating little distinction from fellow members of the associated cluster. The opposite, though, was found for the preposition 'to'. In that case, a concept of 'access to services' was found in the related form of 'accessing services'. The strong correspondence between 'to' and 'actions' reflects that the noun phrase 'access to' might further be seen as 'accessing'.

All the same, that approach failed when intended meanings would be obscure even to humans; such was the case for the phrase 'avoid contact services' [sic.] that made more sense in the form 'contact with services'. Indeed, that phrase might justify machines transforming nouns into verbs when accompanied by 'with', as was done for the preposition 'to'. Unfortunately, 'with' showed less strong correspondences with meta-types, and further shared a cluster with other prepositions. HC analyses, might, though, enable more discerning approach to handling larger clusters arising from CA.

Following the pass of CA that isolated 'to' as an outlier, the subsequent pass revealed 'of' to be the next most distinctive preposition. The high correspondence between 'of' and 'things' was reflected in nodes containing adjacent words, but in reverse order to that in originating triples. Indeed, examples showed that nouns flanking the word 'of' might be swapped, once that preposition is removed. In that way, 'pattern of behaviour' was synonymous with 'behaviour patterns', as were the phrases 'authority figures', 'family dynamics', and 'suicide risk'. Nodes containing such phrases might be rewritten, with no loss of truth value. That will, indeed, be done in experiments to be reported shortly in Chapter 8.

7.9 Chapter Discussion

CA, then borrows from classical mechanics in endowing graph points with proportionate masses. In turn, those masses exhibit inertia, which indicates the strength of correspondences between sets I and J: the respective matrix rows and columns. Successive factors arise that explain variation between categories from column headings, and observations on those categories in matrix rows. Indeed, Chapter 6 described various approaches to analysing text by means of CA; accordingly, the first of the following two discussions revisits those approaches in light of experiments presented here. Subsequently, a further discussion assesses the benefits and pitfalls of this novel approach to resolving ambiguity. To begin, then, by assessing that approach in relation to existing work.

Automated CA in Contrast To Reviewed Approaches

The first reviewed application of CA was to analysing word usage across genres of text. To that end, Nishina (2007) used CA to reveal words that authors were more or less likely to use in particular genres. That study, failed to superimpose graphs from sets I and J from the associated CA. A further study by Unmans (1998) showed overlaid graphs to be far more informative, which is why that format was adopted here. Further note was made of that study's use of quadrants from resulting graphs, as a more detailed view of correspondences; that approach was further taken here.

That study, though, did not consider the proportions of variation explained by particular factors. In contrast, information retained here after CA had finished identified weak factors that were removed

prior to subsequent runs. That said, Unmans (1998) did show the importance of considering distances between points, which reflect the extent of any correspondence. That was adopted here, in particular for identifying outliers, interpreting axes, and deriving optimum numbers of clusters. In addition, Unmans (1998) overcame CA's limitation of allowing just two measures by compressing three into a two-way contingency table. In a similar way, meta-types from before and after prepositions were amalgamated in to pairs, for use as column headings.

Indeed, Tono (1999) noted that humans have difficulty in deciding just how many clusters are depicted in any graph, and what points comprise those clusters. Despite that problem, the automated approach taken here successfully identified optimum numbers of clusters; further, corresponding row and column clusters were aligned by machine, rather than by eye. All the same, that study suggested applying CA to trigrams, although comprising actual POS, rather than the meta-types employed here; by transforming stop words such as pronouns into meta-types, my analysis was less limited by WordNet's lack of coverage, and better reflected the roles that such words fill.

Further inspiration for the approach proposed here came from Izumi et al. (2007), who introduced WordNet into CA. That, though, was to measure any semantic distances between words, in deciding acceptable word substitutions. Here, in contrast, WordNet contributed POS for both ambiguous and unambiguous words. In addition to helping to build a CA matrix, deficiencies in WordNet itself were subsequently rectified by CA of further, ambiguous trigrams. In that respect, words missing from WordNet were assigned POS, while ambiguous ones were interpreted in the context of prepositions.

The overwhelming difference between those studies and the one described here is the degree of automation. Graphs from those studies were read by humans, while here, all steps of CA, from building the matrix to interpreting graphs was done fully automatically. In addition, removing outliers after having noted such strong correspondences allowed subsequent more discerning runs of CA.

Disambiguating Mind Map Concepts by Means of CA

Chapter 5 described prepositions as function words, or stop-words, that are often removed before analysing text. Rather than ignoring such words, an adjunct based on lists of stop words yielded WordNet-like synsets for prepositions, articles, pronouns and so on. In that way, occurrences of the ten most popular prepositions from GRiST mind maps contributed to a CA matrix. Patterns of word usage around those prepositions, it was thought, might help to resolve ambiguous words appearing in similar contexts.

Indeed, that was so for the preposition 'of', which corresponded strongly with nouns. Conversely, modifiers tended to group around the word 'by', while verbs were strongly associated with things. Those patters, in fact, described the axes of the CA graph, which were very useful in determining any best

POS. In fact, such strong correspondence was sometimes a problem; the preposition 'to', for example, so strongly reflected verbs as to mask any alternative interpretation. That was due to the dual role played by 'to', as a normal preposition, and as the standard way of expressing infinitives in English. The challenge for future research is to distinguish between nodes such as `[wife about to leave]` that uses the infinitive 'to leave', and `[as opposed to the day before]` where 'to' is just an ordinary preposition.

The automated CA presented here, then, goes further than humans usually do with that technique; using CA's own output to refine subsequent runs has proved very beneficial. In particular, repeatedly removing outliers brought out patterns that were masked in earlier runs. Thinking in terms of physical bodies under the influence of gravity gives insights into how CA might be manipulated into depicting the best correspondences. That analogy, though must not be taken too far, as gravity does not repel points as might clusters within a CA space.

The weakest point of this approach concerns points that congregate around the origin of any graph, revealing that little exists in the underlying matrix to differentiate between such observations. Even having removed outliers and re-run CA, novel correspondences that arose were accordingly weaker than any preceding ones. In that sense, the problem lies not in CA but the very measurements that it analysed: there simply was insufficient information recorded in the matrix. Using a larger number of mind maps might help in that respect, but is impossible due to the mind mapping phase of GRiST being over. Indeed, the problem is that English does not employ those propositions in ways that differentiate between them.

Although automating CA was generally beneficial, there were areas that need further work, such as the problem with 'to'. The outstanding success arising from trigrams centred on 'of', though, proved the efficacy of this approach. Further, aspects of that automation might be improved, such as the mapping algorithm between row and column clusters; that will prove less reliable for larger numbers of clusters, though it worked adequately here. In addition, it might prove useful to measure distances between points on graphs directly, by means of JFreeChart `Point` objects that hold X-Y coordinates.

Reconfiguring Phrases that Contain Prepositions

Supplementary results from CA of prepositions and meta-types showed the words 'to' and 'of', in particular, suggested respective 'actions' and 'things'. Those distinctions, in turn, suggest addressing the roles that words play in nodes. In a similar way to reflecting antonyms and hypernyms as separate groups, tuples based on intensional knowledge allow researchers to contrast, say, 'abuse' as an 'action' as opposed to as a 'thing'. Having determined what roles words play, subjects and objects of verbs arise from co-occurring words. That would greatly improve the psychological knowledge structures about forms of mental-health risk said to be at the heart of GRiST (Buckingham et al., 2004; Buckingham, Adams, & Mace, 2007).

CA combined with stemming and WordNet, then, supplements any hierarchical associations inherent in GRiST mind maps, which are enriched with cross-links that Novak and Cañas (2006) thought absent from concept maps, and by implication, from mind maps. Further, the normalisation process that yielded intensional knowledge rectifies mind mapping's lack of formalism that Cañas and Carvalho (2004) criticised. All such extra knowledge will shortly be shown by means of extensions to FreeMind; for now, attention turns to refining the structure of GRiST mind maps, after a summary of this chapter.

7.10 Chapter Summary

The introduction to this chapter stressed that responses from WordNet are sometime ambiguous, or even non-existent. A further lack of stop words in WordNet hampers any approach based on such words. Experiments were reported that assessed the application of CA to resolving ambiguity. Specifically, that involved identifying reliable patterns of word usage around prepositions, which subsequently acted as templates for resolving ambiguous cases. To that end, an adjunct to WordNet allowed machines to research stop words as if they existed in WordNet.

Following that introduction, an experiment was reported that identified triples of the form {$word_1$, *preposition,* $word_2$} in nodes from GRiST mind maps. The resulting list was subsequently split into unambiguous and ambiguous triples. Unambiguous triples comprised words having either a single WordNet entry, or a clearly dominant POS. Transforming such triples yielded trigrams of the form {*meta-type*$_1$, *preposition, meta-type*$_2$}, which were subjected to CA. The matrix for that analysis, though, was supplemented by deriving meta-types from POS, to allow, for example, words that represented 'things' to participate as nouns.

The first phase of CA compared results for considering both 2 and 3 senses, to assess any differences in changing the criterion for accepting predominant POS; in fact, little difference was found. Following that, an experiment in identifying outliers clusters was reported, which further helped to determine optimum numbers of factors and clusters for a second, finer analysis. In addition, X and Y axes on the graph from that first pass reflected continuums of varying degrees of 'things', 'actions' and 'modifiers'. In that way, axes indicated POS for words missing from WordNet. A further aspect of that first run of CA concerned outliers; those identified in any particular run were removed, prior to subsequent runs of CA that realigned points on resulting graphs. In that way, finer distinctions arose that were earlier obscured by such strong forces.

Having determined patterns of proposition usage, ambiguous trigrams were compared against clusters that arose from successive passes of CA. Should a generated trigram of the form {*meta-type*$_1$, *preposition,*

meta-type$_2$} match the label on a point from the CA graph, the corresponding meta-type was taken. Although that proved useful for words having ambiguous entries, certain problems associated with that approach, along with its advantages, were covered in a more general discussion. This summary now closes the chapter, in order to further consider CA as a means of refining the structure of GRiST mind maps.

8

Refining the Structure of GRiST Mind Maps

8.1 Introduction

Knowledge at the heart of GRiST arose from interviews with a multidisciplinary panel of mental-health experts. Forty-six such interviews were recorded as mind maps, which were subsequently integrated into a single, combined mind map; the resulting node hierarchy comprehensively represented knowledge about risk-assessment. By means the GRiST web-site, experts gave feedback for refining that combined mind map, to reveal associations between specific risk factors and cues exhibited by service-users. Refining the combined mind map in that way, though, demanded great effort and forbearance by GRiST researchers: in particular, getting experts to fulfil their tasks was a struggle. Even so, agreement of over 90% arose between two independent reviewers who checked correspondence between individual and combined mind maps; that considerable manual effort was further ratified by a focus group (Buckingham & Adams, 2006; Buckingham, Ahmed, & Adams, 2007).

Refining the initial tree of GRiST knowledge into a final hierarchy involved savage pruning, inspired by the rules of normalisation for relational databases. Although that process was automated, human experts monitored the suitability of any cuts. In the resulting combined mind map, higher-level nodes reflected a greater number of concurring experts than did less vital concepts, further from the root node. That process of refinement, though, was hindered because the template for individual mind maps evolved over time (Buckingham & Adams, 2006; Buckingham, Ahmed, & Adams, 2007).

A combined mind map, then, reflected consensus between forty-six mental health experts. In that respect, GRiST researchers faced the challenge of reconciling various representations of key ideas from individual mind maps. Take, for example, differences in hierarchical structures that concerned forms of self harm. Figure 8.1 shows examples from mind maps created by experts designated numbers 2 and 3, with relevant nodes shaded grey:

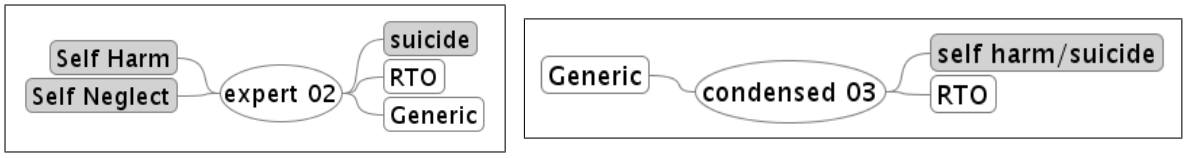

Figure 8.1: Structural differences in representing self harm and suicide, from GRiST experts 2 and 3.

Figure 8.1 shows that expert number 2 separated the concepts of self harm and suicide; in contrast, expert number 3 combined both of those concepts into the single node `[self harm/suicide]`, while omitting entirely the node `[self neglect]`. Both of those specialists, though, included the nodes `[RTO]` and `[Generic]`, respectively for Risk To Others and for less specific risk factors. Yet another specialist, number 12, repeated nodes for the combined concept `[self harm/suicide]` at two levels; Figure 8.2

shows that node was represented as a child of itself:

Figure 8.2: Repeated nodes for self harm and suicide, from GRiST expert 12.

Figure 8.2 shows that expert 12 concurred with expert number 2, in that both used the node `[self neglect]`. Variation arose, though, in that as well as the combined concept `[self harm/suicide]`, expert 12 further used separated versions in the nodes `[self harm]` and `[suicide]`. Indeed, that node hierarchy might be seen as appropriate in terms of mind mapping, in that a higher-level concept branches into two more specific concepts. There was, though, little to be gained: that arrangement encapsulated no additional knowledge, but merely repeated an existing association.

A final example of divergence between experts in describing concepts appears in Figure 8.2. Specifically, the concepts 'drugs' and 'alcohol' were represented in different node structures, though all in relation to substance abuse and suicide. Note that the identities of experts are shown in brackets, following the words in any node[1]:

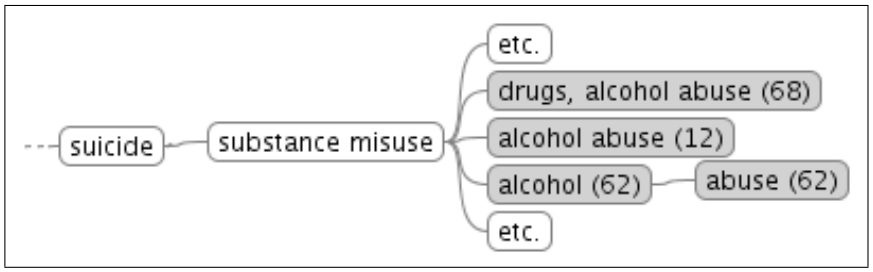

Figure 8.3: Structural differences in representing 'alcohol', from various GRiST mind maps.

The three examples from Figure 8.2, then, exemplify variations between individual GRiST mind maps. Whereas specialist number 68 combined three concepts of 'drugs', 'alcohol' and 'abuse' in a single node, expert 12 combined just two of those concepts, 'alcohol' and 'abuse'. Expert 62, while omitting 'drugs' in the same way as did expert 12, represented 'abuse' as branching from the more general concept of 'alcohol'.

Originally, such differences were resolved by pruning away less commonly used nodes; although a software algorithm suggested nodes for removal, a panel of experts ratified any such cuts (Buckingham, Ahmed, & Adams, 2007). In contrast, this thesis allows machines to account, unaided, for such variations

[1]Originally, those IDs were removed during refinement, and stored separately (Buckingham, Ahmed, & Adams, 2007).

in expression without discarding knowledge. Dealing with such structural differences will, in fact, be a two-step process which starts by gathering together nodes that express related concepts. Subsequently, appropriate overall hierarchies must be determined. To that end, CA's facility for clustering will be applied slightly differently to the way that helped in resolving ambiguity.

From the examples in Figures 8.1 and 8.2, humans might easily spot related concepts of 'alcohol', 'drugs' and 'substances'; that those words bear no resemblance hides such relationships from machines. Further, that GRiST experts associated those concepts is no guarantee that they are, in fact, related; that arises from the lack of control imposed by mind mapping in general. Now, although stems from Chapter 4 will conflate words that express specific forms of some overall concept, such as 'abus' for 'abuse' and 'abusing', related words from the given examples bear no resemblance. For that reason, WordNet will be needed to reveal relationships between morphologically different words that express similar meanings. Accordingly, the proposed roles for WordNet and CA in refining mind map structures are addressed next.

8.2 Tools for Refining Structure: CA and WordNet

Chapter 4 showed that CA produces clusters of points on graphs; mappings between row and column clusters revealed patterns in the type of words surrounding specific prepositions. In a similar way, CA clusters will reveal candidates for re-structuring. In fact, it is not single nodes which participate in that process. Rather, it is entire paths, from the root node to any node of interest. That analysis will use an aspect of CA that has not yet been covered, namely, Hierarchical Clustering (HC). As Chapter 4 showed, the required number of clusters was coded in any matrix header; requesting a higher, optimum number of clusters further split those emerging from a preliminary run. In fact, CA has the necessary information to hand, regardless of the number of clusters actually requested. That is because CA clusters are hierarchical in nature (Murtagh, 2005). Any particular cluster might comprise several smaller ones, as shown next.

8.2.1 The Hierarchical Nature of CA Clusters

A previous example of CA will serve to describe HC, namely, the study by Nishina (2007) that applied CA to identifying genres of text that employed similar words; various genres constituted set I, while words of interest comprised set J. Figure 8.4 reproduces that graph, on which ellipses highlighted clusters of categories in opposing quadrants. For example, widely separated clusters for Learned and Scientific Writing in category J and Press Reportage in category A showed those genres to use very different styles. A further well-defined group for fiction, comprising categories K, L, N and P, revealed relatively similar language usage. Genres congregated around the graph's origin, though, had less distinct styles:

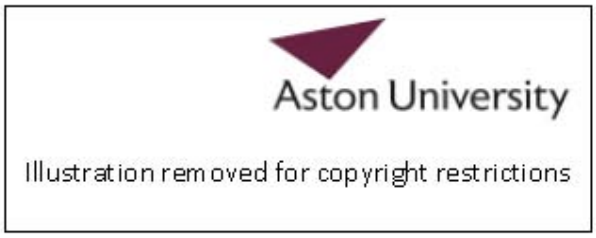

Figure 8.4: Plot of text categories based on word frequencies (Nishina, 2007).

In fact, clusters from Figure 8.4 were further analysed by HC, which is commonly shown as a dendrogram that depicts larger clusters made up from numbers of smaller ones (Murtagh, 2005). In that way, Figure 8.5 depicts a hierarchical breakdown of genres as a dendrogram, with three main clusters dividing repeatedly into more specific text categories (Nishina, 2007):

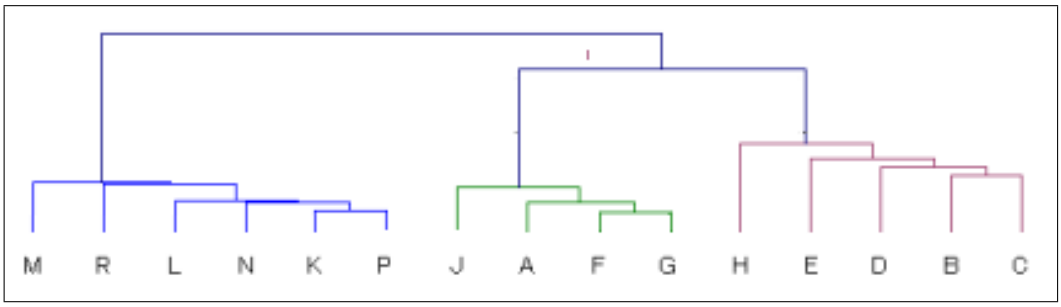

Figure 8.5: Hierarchical clusters within 15 categories of text, adapted from Nishina (2007).

Although HC was used in that study of genres, scant interpretation of the resulting dendrogram was given; a more detailed discussion is offered here, to make up for that omission. The first main cluster from Figure 8.5, then, comprised genres M, R, L, N, K and P. In fact, the last four of those genres, L, N, K and P, were highlighted in the graph from Figure 8.4 as a discrete cluster of fiction genres. Nearby points for categories M and R, though, appeared in the same quadrant as did that former group of four, which justifies clustering all six points together. Indeed, points M and R respectively covered Science Fiction and Humour, which might well belong with other genres of fiction.

Points for categories M and R, on the other hand, were closer together near the origin, somewhat removed from the remaining four points. Although all six points appeared in the same quadrant of Figure 8.4, points M and R might be seen as a separate cluster; that well demonstrates the need for human judgement in interpreting CA graphs. Even so, such decisions are open to doubt. It is important to note, though, that the number of clusters must be specified in advance (Murtagh, 2005). In fact, a mere three clusters was too coarse a measure; had more been requested, points M and R might well have constituted a top-level cluster by themselves. Thankfully, HC allows such conflicting interpretations to be held simultaneously, regardless of any specified number of clusters.

A better use of HC was evident in a study of texts taken from Wikipedia; that work aimed to create an ontology, similar to the one presented in Chapter 2 that was created by Sure et al. (2002); remember that such ontologies express subsumption relationships between atomic concepts. To that end, 405 nouns from 13 Wikipedia texts were treated as successive triples. Notably, stemming was not applied, and a stop list avoided so-called irrelevant words. Rather than processing all permutations of nouns, though, triples reflected a time-series over which changes in meaning were sought. That was done by considering

triangles within the CA space described by points from any triple; in particular, isosceles triangles revealed subsumption relationships in favour of dominant points within triples (Murtagh et al., 2007).

Rather than that process itself, though, it is the application of HC that is important here. In fact, a concept hierarchy arose directly from HC; branches in the resulting dendrogram indicated a complete concept hierarchy for the associated ontology. In that way, CA led to an ontology based on subsumption relationships between terms (Murtagh et al., 2007). That suggests using CA to cluster together similar paths from GRiST mind maps, and further generating a combined mind map by means of HC.

Generating Mind Maps from Hierarchical Clusters

Before addressing entire paths of nodes, the utility of hierarchical clusters will be demonstrated by means of CA between prepositions and meta-types, from Chapter 7; the first experiment reported there arose from CA on a matrix of 10 prepositions and 9 meta-type pairs. Tables retained from CA gave the HC matrix shown as Table 8.1. The first column cl_{row} shows the order of emergence of row clusters for set I, followed by a column for corresponding prepositions. The right-hand side of Table 8.1 shows internal labels generated by CA. Even from these raw data, a hierarchy of clusters is discernible; the first HC column holds the value 19 for all rows, whereas the second column has two unique values: 9 in the first row, and 18 in remaining rows. Subsequent columns show successively larger numbers of subdivisions:

cl_{row}	Prep	HC Labels									
0	to	19	09	09	09	09	09	09	09	09	09
1	of	19	18	07	07	07	07	07	07	07	07
2	by	19	18	17	14	14	14	03	03	03	03
3	on	19	18	17	14	14	14	08	08	08	08
4	in	19	18	17	16	13	13	13	05	05	05
5	with	19	18	17	16	13	13	13	10	10	10
6	for	19	18	17	16	15	04	04	04	04	04
7	about	19	18	17	16	15	12	12	12	01	01
8	as	19	18	17	16	15	12	12	12	11	02
9	like	19	18	17	16	15	12	12	12	11	06

Table 8.1: HC matrix from pass 1.

The cluster hierarchy from Table 8.1 becomes more evident on removing duplicated cells, as in Table 8.2, in which rows are shaded to emphasise cells that are common to consecutive entries:

cl_{row}	Prep	HC							
0	to	19	09						
1	of	19	18	07					
2	by	19	18	17	14	03			
3	on	19	18	17	14	08			
4	in	19	18	17	16	13	05		
5	with	19	18	17	16	13	10		
6	for	19	18	17	16	15	04		
7	about	19	18	17	16	15	12	01	
8	as	19	18	17	16	15	12	11	02
9	like	19	18	17	16	15	12	11	06

Table 8.2: Refined HC matrix from pass 1.

An immediately apparent trend from Table 8.2 is that paths grow longer for successive groups of prepositions. Entries for 'to' and for 'of', though, are distinct from the remainder in that they comprise specific rows in the table; those rows correspond to the outlier point 'to', and to the next most extreme point, 'of', that was the outlier from the very next pass of CA. In contrast, subsequent rows for 'by' and 'on' had identical values under the first four columns, before diverging in the last cells for those rows; entries for 'in' and 'with' showed a similar trend. From the row containing 'for', values of 15 appear in the column where, earlier, 'by' diverged from 'on'; that shared value indicated a cluster comprised of 'for', 'about', 'as', and 'like'. In the subsequent column, a common cell holding 12 depicted a smaller cluster of the last three of those prepositions, which further branched into a yet smaller cluster for 'as' and 'like'.

In fact, such embedded clusters are evident from the second HC column onwards; the first column, though, does not differentiate clusters. Repeated values of 18 in the subsequent column make 'to' distinct from remaining prepositions, which form a separate cluster. Should just two clusters be requested, then, that latter cluster would contain all but 'to'. Requesting larger numbers of clusters would force divisions at successively smaller clusters indicated by HC. In that sense, the actual number demanded is irrelevant, as clusters may be addressed at whatever granularity by means of the HC matrix. All the same, deriving optimum numbers of clusters remains useful for indicating the best level of detail. Indeed, taken too far, HC would yield single-row clusters that would not be very helpful.

Such sub-divisions within CA clusters, then, are commonly represented as dendrograms. Note, though, that dendrograms comprise nodes separated by branching lines, an so qualify as semantic networks just as do mind maps. That relationship suggests that nodes from dendrograms might simply be re-coded as mind map nodes, in a hierarchy determined by HC. Transforming results for prepositions and meta-type pairs in that way gave the two mind maps in Figure 8.6:

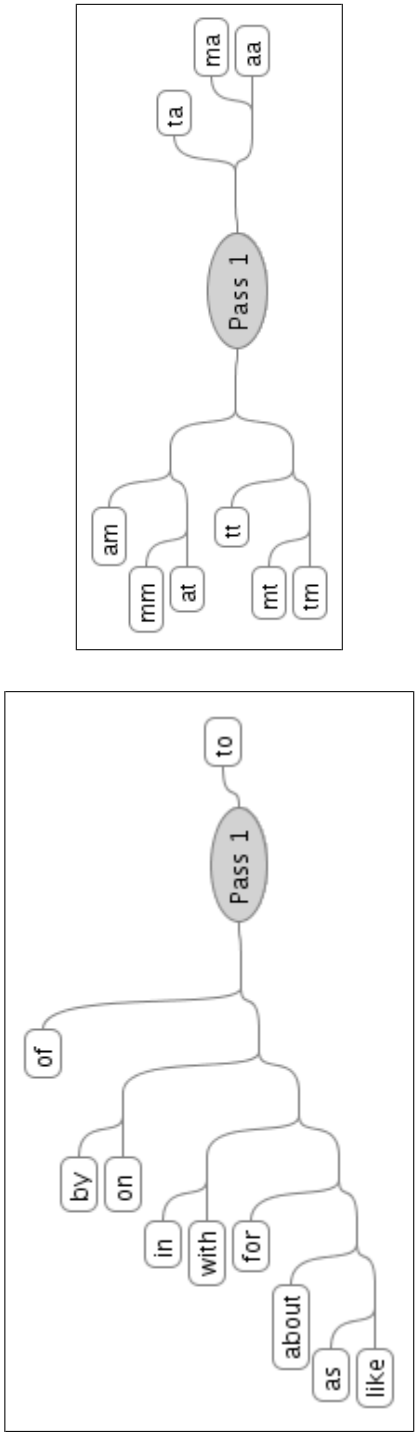

Figure 8.6: Cluster hierarchies of prepositions and meta-type pairs from pass 1 of CA.

Clusters of prepositions from the left-hand side of Figure 8.6 show 'to' and 'of' as singleton clusters, while remaining words form a large cluster that gradually reduces to several smaller ones. Note that 'in' and 'with' form a distinct sub-group, just as the HC matrix dictated; further, 'as' and 'like' correspond more closely with one another than they do with 'about', and even less so with 'for'.

The right-hand side of Figure 8.6 shows three main clusters of meta-type pairs that correspond to preposition clusters of to the left of them. Meta-type pairs for 'to' show the correspondence with 'actions' noted in Chapter 7. Further, the point for 'tt' is somewhat separated from the remaining meta-type pairs for the preposition 'of', which corresponded so strongly with 'things'. In fact, the three clusters in each of the mind maps from Figure 8.6 bear little sub-division, due to the small numbers of points involved. Paths to related concepts in GRiST mind maps, on the other hand, are more plentiful; such larger clusters might be amenable to further analysis by HC.

In addition to HC, further information from retained CA results will reflect the relative importance of concepts in any resulting knowledge structures. The first such information concerns correlations between factors and particular points on graphs; further, CA reports the contribution that points make to the overall inertia within any CA space (Murtagh, 2005). Together, correlations and contributions indicate the relative importance of nodes within any cluster; that should not, though, be confused with the MG measure that Buckingham et al. (2004) used to reflect the relative importance of particular cues to specific risk factors. Rather, high correlations indicate rows from any CA matrix that particularly correspond to given factors; even highly correlated points, though, might contribute little inertia, due to a scarcity of data in any underlying matrix.

Hierarchical clusters from CA then, will be transformed into novel mind maps. The matrix for that analysis will comprise normalised paths from Chapter 2; reducing duplicated nodes to a sole representative means processing just unique paths from the collections of GRiST mind maps. Paths to related ideas, though, might differ; accordingly, the aim here is to create a single, idealised version of all such paths. Further, spelling corrections and stems, as presented in Chapters 5 and 6 respectively, will contribute to identifying related words. The problem remains, though, of identifying distinct word forms that yet have related meanings. The tool for addressing that problem is WordNet, as will be shown next.

8.2.2 Drawing Related Meanings from WordNet

The overview of WordNet in Chapter 5 described separate semantic networks comprising the four main POS: nouns, verbs, adjectives, and adverbs. In particular, WordNet's network of nouns supports relationships such as hypernymy that embodies subsumption relationships. Pointers between synsets in that network lead to more or less specific nouns, depending on the direction of any search. In addition, antonyms express opposing concepts that will, in turn, place certain mind map nodes in opposition.

Hypernyms, though, arise predominantly from nouns; verbs, on the other hand, form verb-groups that reflect related actions, while glosses provide short sentences as examples of word usage. All of those facilities will help to identify words from GRiST mind maps that have related meanings.

A further WordNet facility reflects the degree to which any two words are related, in the form of a semantic distance. A problem arises, though, that is analogous with the Edit Distance, L, in that large distances mean relatively low similarity. Indeed, all nouns would ultimately be sub-concepts of 'Thing' at the top of the hierarchy. Relationships, then, must be kept within sensible bounds, which in turn means imposing an upper limit on such distances. Similar to the way that excessive L avoided inappropriate spelling corrections in Chapter 3, distantly related words will be not be used. Even in cases of acceptable semantic distances, words from mind map nodes might have so many different uses as to be too general.

In fact, WordNet has facilities that will help to overcome those problems. The first of those is the distance reported by WordNet, in terms of the number of levels between words in the network's hierarchy. Further, a measure called familiarity reflects the various senses that words might have; more frequently occurring words tend to have higher numbers of distinct meanings (Beckwith et al., 1993). In that way, relationships offered by WordNet will be declined should words in any comparison be over-familiar.

8.2.3 Combining Hierarchical Clustering and WordNet

The overview of CA in Chapter 6 revealed that input to any analysis was a matrix of observations on certain categories; in this chapter, categories comprise the stems arising from Chapter 4. Observations, then, will be made of nodes that contain specific stems. In that way, any particular matrix row reflects whatever stems were found in a given representative node. Subsequent CA on that matrix will reveal clusters of nodes that contain related word forms such as 'abuse' and 'abusive'.

Using just morphological similarities, though, will miss connections between dissimilar words having related meanings. Accordingly, hypernyms, antonyms, verb groups and glosses from WordNet will reveal such semantic links. Any word found by means of stemming will further be researched in WordNet to yield a list of sufficiently related words. Nodes containing any word from that list will be counted as occurrences of the stemmed word that drove research in WordNet. In that way, words of similar form *and* of related meanings will form matrix rows. Particular cells from any row indicate what stems, and further, what related words, were found in a given node. Indeed, more than one column might be set for that row, which will draw together nodes that express any number of concepts.

A further complication is that nodes which express a given concept might occur at any level. Because the analysis that follows aims to find related paths, rather than discrete nodes, the level at which any concept arises must be recorded. CA, though, allows just two measures; information about levels will,

then, be encoded in row labels. In that way, row labels will depict any given path's order of appearance in the collection of GRiST mind maps, and in addition, the level at which any node of interest occurred within that path. Subsequent CA of such matrices will reveal clusters of paths having nodes that express related concepts, as will be shown by experiments in refining mind map structure.

8.3 Experiments in Refining Mind Map Structure

Experiments presented here will, in fact, be restricted to the upper levels of GRiST mind maps; that is due to the tendency noted by Buckingham and Adams (2006) for key ideas to be formulated close to the root node of any given mind map. Specifically, nodes will come from immediately under any root node, and a further two levels down any hierarchy; those levels will be termed 0 - 2 inclusive. A subsequent CA will address nodes from between levels 4 and 5 inclusively.

Further, two stages are reported here, firstly from using just stems to identify morphologically related words. That analysis will use a simple contingency table, where cells record '0' by default, and '1' should a given stem be detected. A more discerning approach, though, will adjust cells according to whatever stems occur in nodes from any particular path. That will give a matrix of decimal values, rather than of integers, which will reveal more detailed correspondences between paths from GRiST mind maps.

In addition to morphologically related nodes from mind map paths, a further CA will be supplemented by semantic relationships suggested by WordNet. That will yield clusters of nodes related both by words conflated by stems, and by words that have similar meanings, despite taking differing forms. Regardless of the manner in which related words are detected, HC nodes will be transformed into mind maps that reflect the hierarchy derived from CA. Further, the important concept of abuse, conflated by the stem 'abus', was chosen to demonstrate the current approach. All of the following experiments involved a further bespoke Java class called `MindmapPathsAnalysis` that gathered nodes for analysis, researched WordNet, performed CA, and generated novel mind maps.

8.3.1 Deriving Clusters of Nodes Related by Stems

The following experiments, then, use stems to identify morphologically related words from GRiST mind map nodes. The first of those experiments retrieved nodes from just levels 0 - 2 in paths to the stem 'abus', using a simple contingency table of cells containing a default value of 0, or 1 to record the presence of a particular stem in any node. After that experiment comes one which improved on such basic matrices, by means of weighted observations recorded as decimal numbers. That approach was applied to nodes from levels 0 - 2 in any resulting paths, and further at levels 3 and 4. The first stage, then, was to gather nodes for analysis by CA, as described next.

Method I for retrieving nodes from levels 0 - 2 in paths to 'abus'

The first step identified nodes containing the stem 'abus'; that was done in the same way as for the CA of prepositions in Chapter 4, which applied the Java method `split()` to node texts. Corresponding paths to such nodes were extracted by traversing successive parent nodes until reaching the corresponding mind map's root node. In fact, Chapter 9 will show the exact mechanism; for now, suffice to say that a list of paths to nodes expressing some form of abuse was compiled across the 46 GRiST mind maps. Spelling errors in any such nodes were corrected on retrieval, by means of results reported in Chapter 3.

Further, tuples of the form $nodeID \rightarrow nodeID$ from Chapter 4 dictated sole representatives for any duplicated nodes. Normalising paths in that way sometimes yielded entire paths that were identical but for differing instances of particular nodes. Such duplicates were removed by compiling lists of Comma Separated Values (CSV) from any paths retrieved; in that way, each path was transformed into a string of node keys separated by commas, with no intervening white space. Loading such strings into a Java `TreeSet` object, which stored just unique entries, easily eliminated duplicated paths.

Although paths in this experiment were to nodes that expressed some form of abuse, words conflated by further stems were identified in nodes from any paths retrieved. Indeed, just those stems comprised columns for a CA matrix: the stem 'abus' was not itself used, because all nodes at the end of such paths contained it. With those stems as columns, then, matrix rows recorded instances of particular stems in specific nodes. Initially, all cells were set to a default of '0'. For any cell $C_{i,j}$ at the intersection of row i and column j, the default of '0' was changed to '1', denoting a specific stem in a particular node.

Two measures, then, made up the matrix: node IDs and stems. A third factor was introduced by means of composite row labels, which included the position of any node from a particular path. Accordingly, labels of the form *P-i-j* broke down into *P* for 'path', *i* for the position of that path in the list of unique paths retrieved from GRiST mind maps, and *j* for the node index within the containing path[1]. For example, the label P-5-2 was generated for a node from path number 5, at level 2. That can be seen in Figure 8.7, which presents detail for the first few rows and columns of the resulting matrix; pertinent cells for row P-5-2 appear in bold type:

22 10 SUPNO 3 3 3	alcohol	client	drug	history
P-0-2	0	1	0	0
P-1-1	0	0	0	0
P-2-2	0	0	0	1
P-5-2	**1**	0	**1**	**1**

Figure 8.7: Detail from an integer matrix for CA of paths to nodes containing 'abus'.

[1]The CA implementation from Murtagh (2005) required the first character of any label to be alphabetic, hence the 'P'.

The header of the matrix from Figure 8.7 specified 22 rows of paths to nodes containing 'abus', against 10 columns of stems from nodes in those paths. Further, three factors and three row and column clusters were specified; those optimum numbers arose from a preliminary CA in exactly the same way as for the CA of prepositions and meta-type pairs in Chapter 7. Subsequent column headings show some of the 10 stems other than 'abus' found in nodes from those 22 paths. Highlighted values show that stems 'alcohol', 'drug' and 'history' arose in the node at index 2 of path number 5; although cells in the row $P - 1 - 1$ are all '0', that is due to omitting columns for clarity; any row would have at least one cell set to '1'.

The resulting cluster hierarchy from that optimal CA was retained after completion, and any row labels replaced by text from corresponding nodes. To that end, labels were split into constituent parts by using the Java method `split()`, with '-' as a delimiter, giving the path number i and level j of any node in the original list of unique paths. Those original node texts were subsequently re-coded automatically into the format used by FreeMind, which will be described fully in Chapter 9.

Results Ia for retrieving nodes from levels 0 - 2 in paths to 'abus'

These results show the benefit of representative nodes in normalising mind map structures. In fact, the node `[self harm/suicide]` appeared near the root of 22 GRiST mind maps. Further, 12 instances of the atomic node `[suicide]` arose, while `[substance misuse]` occurred 8 times. The process of normalisation introduced in Chapter 2, then, was furthered by substituting such duplicated IDs with that of the first occurrence. For example, the first of twenty two `[self harm/suicide]` nodes, in paths to nodes containing 'abus', was node 11156, which represented the remaining instances. In addition, nodes such as `[suicide/self harm]` having the same words in different order were deemed equivalent.

Because the current approach works between specific levels within node hierarchies, duplicates arose on discarding lower levels. Two full paths might be unique, but the first three levels, say, might be the same. That was the case for the earlier example of alcohol abuse, which is reproduced here as Figure 8.8:

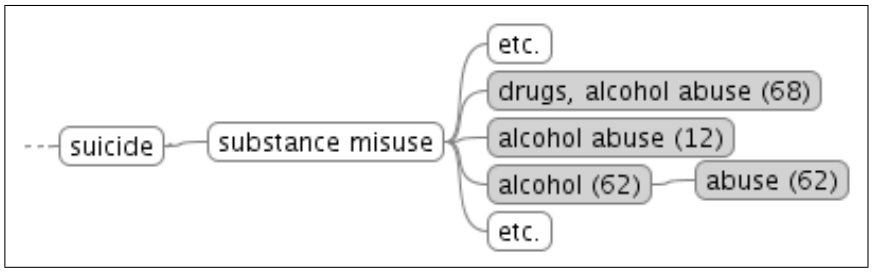

Figure 8.8: Structural differences in representing 'alcohol', from various GRiST mind maps.

Nodes for [suicide] from Figure 8.8 actually branched from [self harm/suicide]; three such paths were identical up to and including the node [substance misuse]. That shared path is better shown in Table 8.3, which gives the levels in those hierarchies and the keys of any representative nodes:

0	1	2	3	4
11156	11176	10355	11293	-
11156	11176	10355	12002	3123
11156	11176	10355	1366	-

Table 8.3: Duplicate path keys between levels 0 and 2, for paths to nodes containing 'abus'.

Paths from Table 8.3, then, diverged at level 3. Because just the first three levels participated in CA, a sole representative of that sub-path sufficed.

Discussion Ia of retrieving nodes from levels 0 - 2 in paths to 'abus'

Although reducing the number of paths participating in CA will lead to quicker processing, that is not the main point of removing duplicated paths. Rather, it is to reflect what Date (1975) called putting one thing in one place. Any information base benefits from such normalisation, in that it makes information easier to understand, both for humans, and more importantly, for machines. By identifying representative nodes and removing any duplicates that arise, an important first step has been made towards a combined mind map for the concept of abuse. Results from the next step, of applying CA to such paths, now follow.

Results Ib for CA on nodes from levels 0 - 2 in paths to 'abus'

A preliminary CA on a full version of the matrix from Figure 8.7 showed that at least three factors should be requested. That was due to the almost identical amounts of variation explained by those factors, as shown by the rates of inertia in Figure 8.9:

```
Rates of inertia for three factors:

F1 rate = 0.1841  include by default
F2 rate = 0.1841  include in CA
F3 rate = 0.1840  include in CA
```

Figure 8.9: Rates of inertia from CA of the integer matrix from Figure 8.7.

Further to the factors identified in Figure 8.9, the algorithm for detecting major differences in F1 between successive members of set I gave just one extra cluster over the default of two, making three in all. Details of a subsequent optimised CA produced results such as those in table 8.4. The first column of that table shows row labels from the original matrix; three sets of values then follow for each factor:

In fact, several results bore identical factor values, so just representative rows were shown. The three sets of results for each factor comprise Fn, the factor values themselves, the correlation CO2 with any

Example Label	Factor 1			Factor 2			Factor 3		
	F1	CO2	CTR	F2	CO2	CTR	F3	CO2	CTR
P-1-1	0	0	0	0	0	0	0	0	0
P-2-2	0	0	0	0	0	0	1969	574	125
P-0-0	0	0	0	768	87	19	0	0	0
P-5-1	0	0	0	2892	896	270	0	0	0
P-0-2	3937	1069	500	0	0	0	0	0	0

Table 8.4: Selected row results from an integer matrix for paths to 'abus'.

factor, and the contribution of any result to the overall inertia explained by that factor, CTR.

Table 8.4 shows just one major subdivision for factor F1, where it jumps from 0 to 3937; the optimum number of clusters, then, was $cl_{default} + cl_{extra} = 2 + 1 = 3$. Indeed, such large differences appeared throughout those results; values of F1, F2 and F3 were either zero or positive four-digit integers. Further, representative results from Table 8.4 were repeated several times, with just row labels changing. Values for correlations from the CO2 columns were similarly zero or large positive integers; the exception was a relatively low CO2 of 87 for path 'P-0-0'. Although values for CTR were smaller than corresponding CO2s, a similar pattern emerged: CTR was either zero or a positive value of the same magnitude as corresponding CO2s. Further note that the first entry from Table 8.4 comprised solely zero values.

The three clusters suggested by F1 from Table 8.4 appear in Table 8.5. The first column shows the order in which clusters were produced by CA. Path numbers and levels in those paths occupy the next two columns, P and N. Following those come nodes that contained stems from the matrix column headings, which appear in bold type. The last column, headed Main Risk, shows the node directly under any mind map root that led to a particular path, should the node in question not itself be one of those main risks:

Row clusters				Main Risk
n	P	N	Text	
0	0	2	**client** presentation	self neglect
	8	2	**client** episodes	rto
1	5	2	past **history** of **drug**/**alcohol** abuse	rto
	25	2	**history**	self harm/suicide
	16	2	**drugs**	self harm/suicide
	15	2	**alcohol**	self harm/suicide
	2	2	life **history**	rto
	8	1	**history**	rto
2	20	2	**subst**ance misuse	self harm/suicide
	5	1	**subst**ance misuse	rto
	6	2	strong concordance with **subst**ance abuse	rto
	1	1	**person**al details	rto
	19	2	**person**al details	self harm/suicide
	28	2	**person**ality **disorder**	self harm/suicide
	10	1	mental/cog/**person**ality **disorder**s	rto
	23	2	mental/cog/**person**ality **disorder**s	self harm/suicide
	19	1	**suicid**e	self harm/suicide
	29	0	**suicid**e	-
	14	0	**self harm**/**suicid**e	-
	0	0	**self** neglect	-
	24	1	**self harm**	self harm/suicide
	30	0	**self harm**	-

Table 8.5: Clusters from an integer matrix for paths to 'abus'.

Note that certain nodes recur in Table 8.5 due to appearing at various levels; the node `[history]`, for example, arose both at level 1 and at level 2. The main risk categories for those nodes, though, differed.

The first cluster from Table 8.5 arose from the sole stem 'client', which was a cue for two risk factors: RTO and self-neglect. The next cluster, number 1, reflected the stems 'history', 'drug' and 'alcohol', which corresponded to RTO and the combined concepts of self-harm and suicide. Indeed, the third cluster had those same risk categories, and accounted for several corresponding stems: 'subst', 'person', 'disorder', 'suicid', 'self' and 'harm'.

Discussion Ib of CA on nodes from levels 0 - 2 in paths to 'abus'

The first cluster from table 8.5 comprised two nodes related by the stem 'client', at the same level. That discovery suggests a normalised hierarchy based on a new node, `[client]`, which would branch into child nodes `[episodes]` and `[presentation]`. Corresponding risk categories, though, are not included in any such idealised path; that would violate normal forms that serve to eradicate duplication. Accordingly, a

normalised hierarchy centred on 'client' should appear just once, rather than being repeated under both `[self harm/suicide]` and `[rto]`. A later experiment, though, will demonstrate that process; it is the clusters arising from an integer matrix that are of concern here.

The second cluster from table 8.5, then, was less well defined than was the first one; all the same, the longer node `[past history of drug/alcohol abuse]` contained several stems, around which congregated nodes expressing just one of those stems. In terms of a combined mind map, those concepts are well enough related to justify treating them together. That cannot be said of the last cluster, though; nodes that contained the stem 'subst' for 'substance' appear in the same cluster as nodes conflated by 'self', 'harm', or 'suicid'. Further, the stems 'personal' and 'disorder' constitute a third possible group within that cluster. Clearly, integer matrices reflect insufficient variation for CA to resolve the required clusters.

Indeed, factor values from row results in table 8.4 are extremely polarised, in that they are either large positive values, or zero. The lack of negative values means that just two of the possible four quadrants of any graph would be occupied. A further problem is that the first entry from Table 8.4 lacked correlation with *any* of the three factors requested. In fact, considering factors F2 and F3 might help to determine optimum numbers of clusters; F2 showed three sub-divisions, while F3 reflected two. That approach, though, would ignore the underlying problem of a general lack of variation. Although HC might further allow finer distinctions, it would be better if the main analysis gave an optimal starting point.

Improving on Simple Contingency Matrices

In order to introduce additional variation into CA matrices, two adjustments were made to formulating cells. The first of those adjustments involved calculating observations to six decimal places, instead of using integers[1]. The second adjustment was to apply weightings in proportion to the column under which any stem-containing node was recorded. Such weightings will yield gradually decreasing column values, from left to right in the matrix. Although that might be seen as diminishing the importance of successive stems, the required outcome of generating more discerning clusters was achieved, as reported next.

Method III for decimal matrices

In fact, the Java implementation of CA by Murtagh (2005) looped endlessly should matrices comprise predominantly zeroes in a decimal form. For that reason, a default observation of 0.000001 was used, rather than zero to six places. Cells set to 1.000000, indicating that nodes contained particular stems, were adjusted in the following way. First of all, an interval I was calculated as $1/n$, where n was the number of columns in any CA matrix. Using zero-based indices, an adjustment for the cell at index j in any particular row was calculated as $j * I$, which was then subtracted from the raw observation.

[1] Initial experiments with four decimal places proved little better than with just the integers 0 and 1.

For cell $C_{i,j}$ then, in row i under column j, an observation was calculated by the formula:

$$\begin{aligned} C_{i,j} &= 1 - (j * I) \\ &= 1 - (j * (1/n)). \end{aligned}$$

A zero-based index means that $j*(1/n)$ will always be zero for the first column, where $j = 0$; observations under that column, then, were unaffected. In that way, the integer matrix for paths involving abuse from Figure 8.7 was recalculated as the decimal matrix in Figure 8.10. The ten columns of that matrix yielded an interval of $1/10 = 0.1$. Observations highlighted by a border demonstrate the resulting gradation of values; note, though, that columns were removed for reasons of space, meaning that particular adjustments might not reflect any exact position in the portion of the matrix shown here:

22 10 SUPNO 3 5 5	alcohol	client	drug	history
P-0-2	0.000001	0.900000	0.000001	0.000001
P-1-1	0.000001	0.000001	0.000001	0.000001
P-2-2	0.000001	0.000001	0.000001	0.500000
P-5-2	1.000000	0.000001	0.700000	0.500000

Figure 8.10: Decimal CA matrix of nodes between levels 0 and 2, in paths to nodes containing 'abus'.

Having performed CA on the full version of the matrix from Figure 8.10, hierarchical row clusters were derived by the process described in Section **??**. Cluster mappings drew together corresponding row and column clusters, to show what stems corresponded to which nodes. Further, HC clusters were transformed into mind map format, and viewed in FreeMind.

Results IIIa for a decimal matrix from levels 0 - 2 in paths to 'abus'

Selected results from CA on a full version of the matrix from Figure 8.10 are given in table 8.6. Labels printed in bold type had associated F1 values that contributed to an optimum number of clusters:

Example Label	n	F1			F2			F3		
		F1	CO2	CTR	F2	CO2	CTR	F3	CO2	CTR
P-0-2	2	-867	99	44	-2541	849	375	612	49	22
P-25-2	6	-867	111	24	847	106	23	612	56	12
P-19-1	6	-110	0	0	0	0	0	-1885	70	23
P-20-2	3	-72	0	0	0	0	0	-415	7	2
P-1-1	5	1511	338	59	0	0	0	469	33	6

Table 8.6: Selected row results from a weighted decimal matrix for 'abus'.

Each of the last three results in table 8.6 revealed a sub-division of F1, which caused cl_{extra} to be incremented. Accordingly, the optimum number of clusters, expressed as $cl_{default} + cl_{extra}$, was $2+3 = 5$. Those clusters are apparent from Figure 8.11, although two appear very close together, even though just a sole label from each column cluster is displayed; row labels are omitted completely for further clarity:

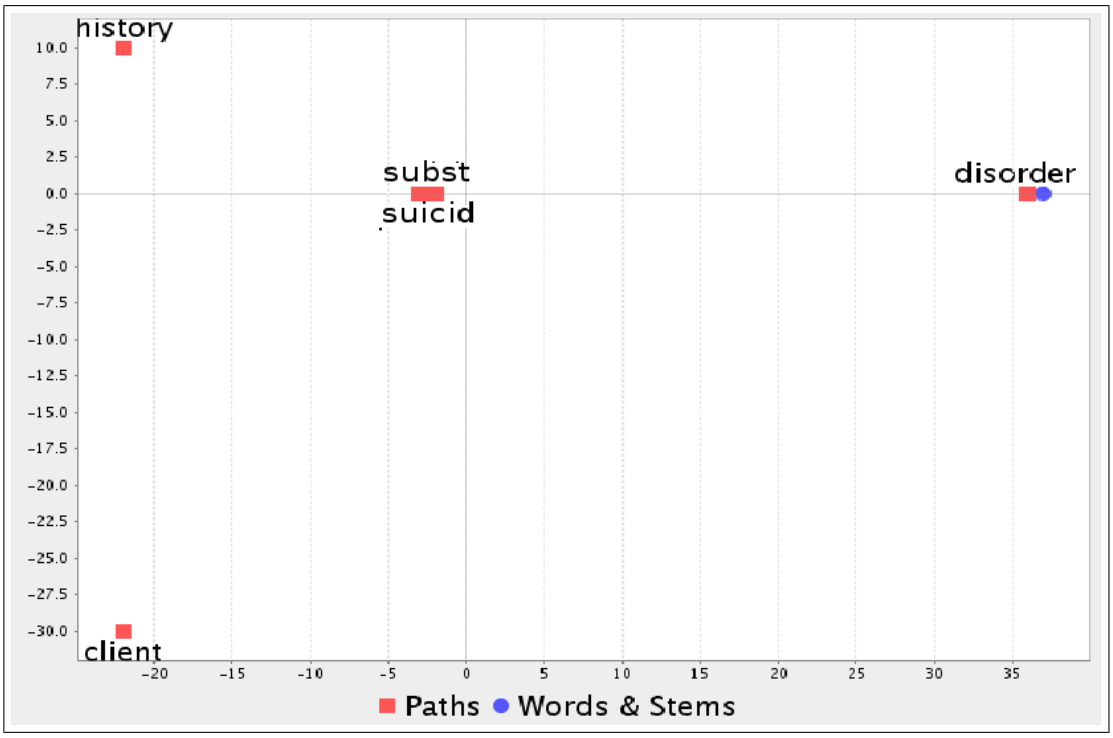

Figure 8.11: CA graph resulting from the matrix in Figure 8.10.

Points for row and column results in Figure 8.11 were so similar that they often overlapped; indeed, the sole visible column label is 'disorder'. Remaining column labels are hidden by the corresponding row result points[1]. The five clusters themselves were well differentiated, and showed variation along both the X and Y axes, for factors F1 and F2 respectively. Clusters containing the labels 'subst' and 'suicid' appeared as a more central cluster, in sharp relief to remaining points. For the stem 'disorder', that was reflected in a wide separation on just the X-axis, with Y coordinates being almost identical. Points for the stems 'client' and 'history', though, lay at similarly large distances along the negative X-axis, although separated at the far extremes of the Y-axis.

[1]In fact, the labels 'subst' and 'suicid' had to be separated slightly in KolorPaint, an Ubuntu Linux image editor.

Mappings between row and column clusters are shown in table 8.7, where node IDs have been replaced by the corresponding text. In this case, row labels are omitted, as they merely point to rows in the original array of path keys; the level i of nodes in any path, though, is retained. Further, table 8.7 has two extra columns for the associated CA clusters of stems from set J:

Row Clusters			Col Clusters		Main Risk
n	i	Text	n	Text	
0	2	**client** presentation	0	client	self neglect
	2	**client** episodes			rto
1	2	**subst**ance misuse	1	subst	self harm/suicide
	1	**subst**ance misuse			rto
	2	strong concordance with **subst**ance abuse			rto
2	1	**person**al details	2	disorder person	rto
	2	**person**al details			self harm/suicide
	2	**person**ality **disorder**			self harm/suicide
	1	mental/cog/**person**ality **disorder**s			rto
	2	mental/cog/**person**ality **disorder**s			self harm/suicide
3	2	**history**	4	history alcohol drug	self harm/suicide
	2	life **history**			rto
	1	**history**			rto
	2	**drug**s			self harm/suicide
	2	past **history** of **drug**/**alcohol** abuse			rto
	2	**alcohol**			self harm/suicide
4	1	**suicid**e	3	suicid harm self	self harm/suicide
	0	**suicid**e			-
	1	**self harm**			self harm/suicide
	0	**self harm**			-
	0	**self** neglect			-
	0	**self harm**/**suicid**e			-

Table 8.7: Clusters from a decimal matrix for 'abus' between levels 0 and 2.

In contrast to the three clusters from CA on an integer matrix, Figure 8.7 reveals five clusters that arose from using decimal observations. While clusters for 'client' and for 'history', 'alcohol' and 'drug' remain unchanged, three clusters arose from sub-dividing what was formerly a single large cluster. As a result, row clusters 1, 2 and 4 respectively hold three, five and six nodes. The corresponding cluster of six stems was broken into separate clusters of one, two and three stems in respective clusters 1, 2 and 4. The largest of those novel clusters, number 4, was no larger than row cluster 3 from using an integer matrix.

The mind map generated from row clusters in Table 8.7 appears next as Figure 8.12, in which nodes containing the identity of corresponding HC clusters branch into clusters of actual nodes from GRiST mind maps. Stems from corresponding column clusters were marked automatically with an asterisk in the text of matching nodes. The root node shows the stem 'abus' capitalised in a shaded node, to reflect the stem that drove CA:

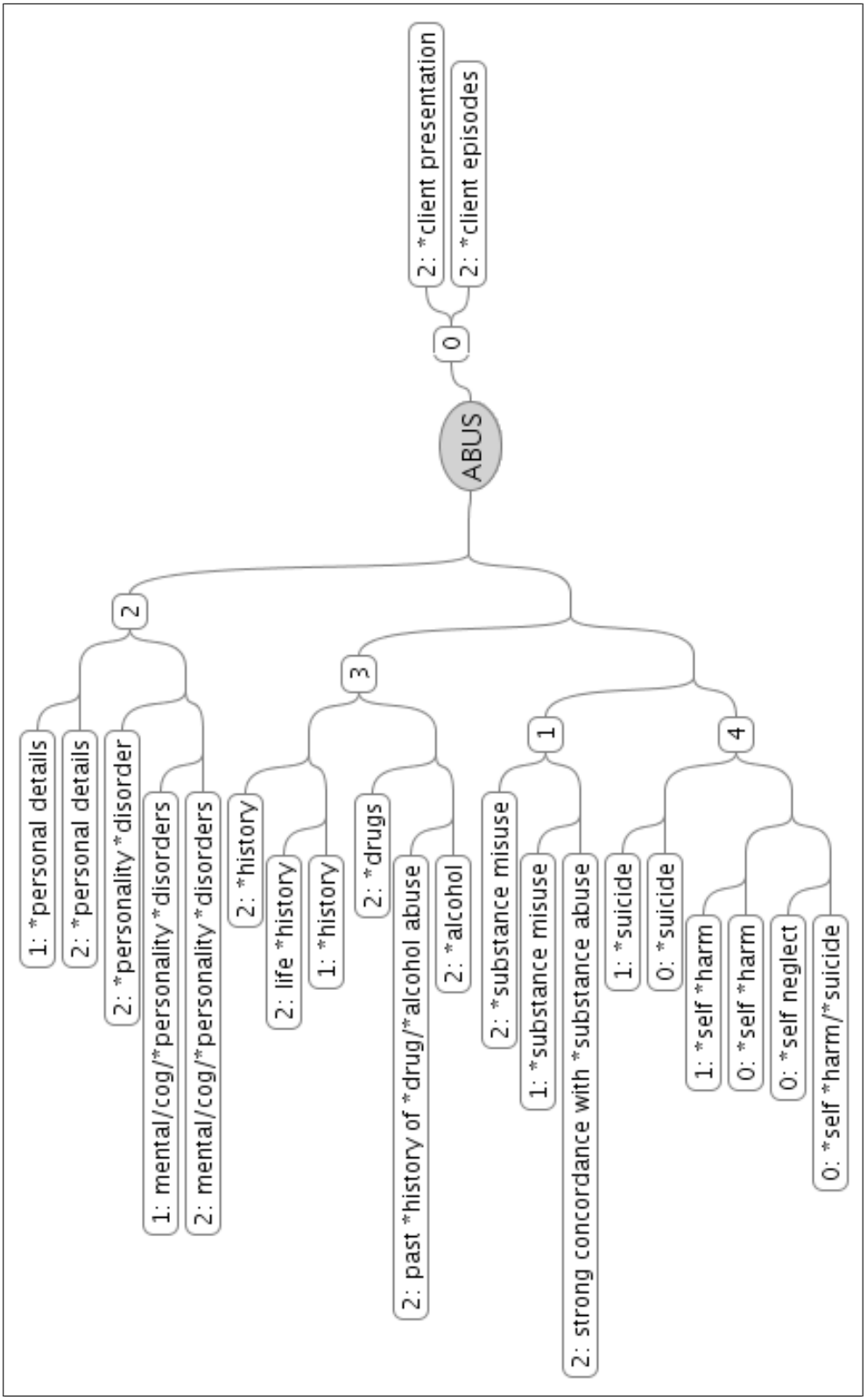

Figure 8.12: Text clusters from CA of nodes between levels 0 and 2, in paths to nodes containing 'abus'.

Figure 8.12 shows cluster number 0 for the stem 'client' as a clear outlier, depicting a major sub-division in the overall HC hierarchy. Remaining clusters under the opposing branch from the root node all show internal hierarchies to some degree. Note, for example, the sub-hierarchy for cluster number 2, in which the node `[personal details]` came from both levels 1 and 2 in respective paths. That small cluster is somewhat separate from nodes that, in addition to the stem 'person', held the further stem 'disorder'. A similar arrangement arose in cluster number 3, where 'history' was kept separate from nodes containing 'drug' and 'alcohol'. The cluster for 'history' was further differentiated by the word 'life'. Overall, though, nodes from cluster 3 are closely related, as are those from remaining clusters.

Discussion IIIa of a decimal matrix from levels 0 - 2 in paths to 'abus'

Results for an integer matrix, shown in table 8.5, included a large cluster in need of refinement; comparable results from table 8.7 for a weighted decimal matrix show that aim to have been met. Clusters numbered 1, 2 and 4 in table 8.7 did indeed result from splitting cluster 2 of table 8.5 into three smaller clusters. The lack of resolution arising from an integer matrix was reflected in the polarisation of F1 values from table 8.4. Conversely, results for a decimal matrix showed an adequately wide range from large negative F1 values to similar sized positive values. Further, gradations within that range showed sub-divisions that were absent before; a mixture of high and low negative and positive factor values from table 8.6 make better use of the axes plotted by CA.

The cluster hierarchy from Figure 8.12, though, had less suitable aspects. Take, for example, cluster 3 for nodes that contained 'history'; a better arrangement would have been the two nodes for `[history]` together, regardless of a difference in levels, with the node `[life] history]` as a further branch. In contrast, instances of the node `[personal details]` from levels 1 and 2 formed a cluster by themselves, with the nodes for `[mental/cog personality disorders]` in a related group; two stems, though, were involved in that case, rather than the sole stem 'person' in the cluster for `[personal details]`. In a similar way, cluster number 4 showed an appropriate separation of nodes for `[suicide]` and `[self harm]`, with a further small group after that. Again, that was because those sub-clusters respectively reflected one, two and three stems. Overall, though, suitable clusters resulted from levels 0 - 2 in paths to 'abus'; results presented next came from a similar CA of lower levels 3 and 4.

Results IIIb for a decimal matrix from levels 3 - 4 in paths to 'abus'

Results that follow arose from CA of paths to nodes containing the stem 'abus'; that matrix was assembled in exactly the same way as was the one just reported, except that nodes had to appear at either level 3 or level 4 in paths from any GRiST mind map. That CA produced the clusters that appear in Table 8.8:

Row Clusters			Col Clusters		Main Risk
n	i	Text	n	Stem	
0	4	**violen**t milieu	1	violen	rto
	3	**violen**ce			rto
1	4	... clearer if there is an identified **person**	3	person	rto
	3	**person**ality disorder			self harm/suicide
2	3	dramatic **event**	0	event	generic
	4	influential **event**s			self harm/suicide
3	4	tranquillisers as a **method** of abuse	2	method	self harm/suicide
	4	**method**			self harm/suicide
4	4	**voice**	5	voice	self harm/suicide
	4	**voice**s			self harm/suicide
5	4	they're not abusing **alcohol** or **drug**s...	6	alcohol drug	generic
	3	**drug**s **alcohol** abuse			self harm/suicide
	3	**alcohol** abuse			self harm/suicide
	3	**alcohol**			self harm/suicide
6	4	**history** of domestic **violen**ce or abuse	1	history, violen	rto
	3	life **history**			self harm/suicide
	4	2 kids under 5... and a **history** of abuse			rto
	3	**history** of abuse potentially important factor			self harm/suicide
7	3	**typical** example	7	client typi- cal	rto
	4	**typical client**			generic
	3	**client** episodes			self harm/suicide
	3	abuse to **client**			rto
	4	abuse to **client**			self harm/suicide

Table 8.8: Clusters from CA of nodes at levels 3 and 4, in paths to nodes containing 'abus'.

The CA for nodes from levels 3 and 4, then, led to the 8 clusters from Table 8.8. Those clusters ranged in size between 2 and 5 paths. Further, clusters 0 to 4 inclusively represented just one stem, while remaining clusters contained two stems. Note, though, that clusters 0 and 6, which were highlighted in Table 8.8, both refer to the stem 'violen'.

Discussion IIIb for a decimal matrix from levels 3 - 4 in paths to 'abus'

CA matrices that comprised weighted decimal entries, then, proved more discerning than did matrices of integer observations. That was because adjusting cells according to what column they occupied introduced additional variation into CA. The actual weighting used, though, was perhaps not quite appropriate, as the dual clusters for the stem 'violen' show. Indeed, clusters relied more on the number of separate stems conflating any group of nodes. All the same, it might have been hoped that the cluster for 'violen' would have appeared as a sub-cluster of the one for stems 'violen' and 'history'. In fact, they were completely

separated, raising a problem for any mind map based on that HC hierarchy. That said, remaining clusters were distinct in whatever stems were reflected.

In a similar way as for results from levels 0 - 2, several risk categories correspond to stems that represented cues in GRiST mind maps. Rather than branching from root nodes, then, such risks will form a discrete node structure to which other nodes might refer. The actual mechanism will be presented soon; for now, though, CA remains tied to just stems, and lacks any indication of dissimilar words that are yet related. Experiments that aimed to improve that situation are reported next.

8.3.2 Deriving Clusters of Semantically Related Nodes

So far, mind map nodes that share any stems identified in Chapter 4 have been conflated by CA. That approach, though, ignores dissimilar words that have related meanings. Fortunately, WordNet has facilities that alleviate any reliance on morphology alone. As the review in Chapter 5 described, WordNet was implemented as semantic networks composed of synsets. Pointers between those synsets allow various types of relationships to be discerned, such as hypernymy for words having similar meanings, and antonymy, for words with opposed meanings. In order to exclude more tenuous relationships, words with high number of overall senses were omitted from the analysis. Further, relationships that reflect large semantic distances across those networks will be ignored. The experiments that follow, then, forge additional links between mind map nodes by means of a CA matrix enhanced with word meanings.

Method IV for semantically related words

Accordingly, the `MindmapPathsAnalysis` class was supplemented by a further bespoke Java programme called `WordPair` that stored results from researching pairs of words in WordNet. In fact, that was done by means of the list of unique words from GRiST mind maps; each word in that list from Chapter 7 was successively paired with all remaining entries. Words from such pairs were researched in WordNet for specific relationships. For nouns, those relationships concerned hypernyms[1] and antonyms, while so-called verb-groups indicated further relationships that were not, in fact, just for verbs.

The separate Java class `AstonWordNet` performed that WordNet research, in the following way. The first step was to look up each word from any pair, storing results in an array of two `IndexWord` objects; such classes came as part of the WordNet Java package from Beckwith et al. (1993). Note that having retrieved a `Dictionary` object by using the static `getInstance()` method, the method `lookupIndexWord()` demands a specific POS.

[1]Note that synonyms are, in fact, hypernyms separated by just one level in WordNet's noun network.

Accordingly, the following code had to be repeated four times, once for each of the POS stored by WordNet for content words. That specific POS was held in a variable called `wNetPos`:

```
indexWords[0] = Dictionary.getInstance().lookupIndexWord(wNetPos, word1);
indexWords[1] = Dictionary.getInstance().lookupIndexWord(wNetPos, word2);
```

Having retrieved a pair of `IndexWord` objects, the further WordNet class `RelationshipFinder` sought out desired relationships. An instance of that class was obtained in a similar way as for the above `Dictionary` class, by using the `getInstance()` method. Subsequently, the `findRelationships()` method reported relationships between given senses of any two words, for a specified type of relationship. In that way, the following code retrieved hypernyms for the first sense of each word from a given pair, by means of the WordNet constant `PointerType.HYPERNYM`:

```
relationshipList =
RelationshipFinder.getInstance().findRelationships(indexWords[0].getSense(1),
                                                   indexWords[1].getSense(1),
                                                   PointerType.HYPERNYM);
```

The above code, then, retrieved hypernyms for particular senses of two specified words, and stored them in the variable `relationshipList`; that process was repeated for remaining senses of those words. Any relationships arising between two specific words were stored as a single `WordPair` object. In addition to the words themselves was stored the semantic distance reported by WordNet, and the familiarity in terms of the number of senses for each word. Relationships separated by semantic distances in excess of 5 were discarded. In a similar way, relationships between over-familiar word pairs were taken no further. To that end, the numbers of senses from each word in any pair was summed, and the corresponding `WordPair` object discarded should that total exceed 15 senses.

That process was applied to words conflated by stems from column headings of the matrix. For relationships deemed close enough, cells for particular node and stem combinations were recorded as if any semantically related word contained a given stem. In that way, additional rows entered the matrix for nodes that, while not referring to any given stem, reflected a word that was related to one of those stems. In the same way as for CA on stemmed words alone, nodes from levels 0 - 2 from paths to nodes containing 'abus' were analysed first. Subsequently, levels 3 and 4 were analysed in a separate CA. In all cases, though, results from HC were transformed automatically into mind maps for viewing in FreeMind.

All of the four main POS might further carry a gloss comprising a short sentence, given by WordNet as an example of actual word usage. Glosses were reduced to discrete words by the Java method `split()`; subsequently, content words from such lists were compared against, say, hypernyms of any stemmed word,

by calling the `lookupIndexWord()` method used to populate `WordPair` objects. That gave a list of related words for any word conflated by a stem from the matrix header. Those related words were compared with words from the gloss of the associated stemmed word. Should, say, a hypernym of any stemmed word appear in the corresponding list of gloss words, then a fresh `WordPair` object was created, and subjected to the process described above, to determine if such words were sufficiently close in meaning.

Results IV for semantically related words

The left-hand side of Table 8.9 lists hypernyms reported by WordNet, with relationships from verb-groups to the right-hand side. Within each of those sections appears a column for the WordNet relationship researched, followed by the POS reported. After those columns come the contents of specific `WordPair` objects, followed by corresponding semantic distances. Note, though, the need to research POS separately led to nouns and adverbs being reported in the verb-group section:

Relation	P	Word 1	Word 2	d	Relation	P	Word 1	Word 2	d
Hypernym	N	ideas	thought	0	Verb-Group	Av	likely	probably	0
		drug	medication	1		N	component	element	
		paranoid	psychotic				degree	level	
		care	treatment				harm	injury	
		frequency	incidence	5			need	want	4

Table 8.9: Accepted word-pairs related by WordNet hypernyms and verb-groups.

All of the relationships from Table 8.9 were accepted due to semantic distances of 5 or less. In a similar format, Table 8.10 lists word-pairs that were related by antonyms at acceptable semantic distances:

Relation	P	Word 1	Word 2	d	Relation	P	Word 1	Word 2	d
Antonym	N	chronic	acute	0	Antonym	N	health	illness	0
		dead	living			Aj	mental	physical	
		different	same				other	same	
		females	male			V	start	stop	

Table 8.10: Accepted word-pairs related by WordNet antonyms.

Relationships from Table 8.10 were, in fact, all noun antonyms that were accepted due to having semantic distances of zero. In contrast, comparisons between stemmed words and those from glosses yielded larger, yet acceptable, semantic distances, as depicted in Table 8.11:

Relation	P	Word 1	Word 2	d	Relation	P	Word 1	Word 2	d
Gloss	N	belief	thought	2	Gloss	N	anxiety	disorders	2
		belief	ideas				intent	thought	
		therapies	treatment				disorder	schizophrenia	5
		alcohol	substances	3			disorder	paranoia	

Table 8.11: Accepted word-pairs related by WordNet glosses.

All of the entries from Table 8.11 were, in fact, nouns that were separated by semantic distances of between 2 and 5. Regardless of any specific type of relationship, though, cases having semantic distances in excess of 5 were rejected. That can be seen in table 8.12, which depicts more distant WordNet relationships:

Relation	P	Word 1	Word 2	*d*	Relation	P	Word 1	Word 2	*d*
Hypernym	V	feel	think	7	Verb-Group	N	continuum	degree	6
		bother	getting				amphetamines	substances	
		anger	feel	9			disease	psychosis	7
		have	lead				disorders	health	8
		plan	think				calorie	degree	9

Table 8.12: Rejected word-pairs related by WordNet.

Relationships between words from Table 8.12 were, then, rejected due to excessive semantic distances of up to 9. Notably, rejected hypernyms from Table 8.12 were verbs, while verb-group entries were nouns.

Discussion IV of semantically related words

Results of experiments combining CA and WordNet, then, covered relationships from hypernyms, verb-groups, antonyms and glosses. Glosses, though, were reduced to Bags Of Words (BOW) before entering the main analysis. From such BOW, gloss words closely related to stemmed words revealed interesting relationships. Take the word pairs 'belief' and 'thoughts', 'belief' and 'ideas', and 'intent' and 'thoughts'; those words are indeed related, although just how closely is debatable. At a general level, GRiST panellists might well be interested in such nodes as a group; in fact, the concept of 'ideation' used by those experts well covers such cognitive activities. At a more detailed level, though, 'beliefs' in particular might be seen as distinct from 'ideas' and 'thoughts', while related to 'intent'. Such contrasting views might in fact, be offered to users; such is the power of intensional knowledge, in allowing analyses at whatever level of detail, without affecting existing mind maps.

More definite relationships arose between 'disorder(s)' and 'anxiety', and between 'paranoia' and 'schizophrenia'. Although anxiety might seem a light affliction compared to schizophrenia, it is more serious from a mental health perspective. Remember, though, that intensional knowledge arose from what Date (2003) termed the lossless process of normalisation; in the same way as for the example of 'belief', then, all of the disorders grouped by WordNet and afterwards by CA continue to exist separately. In practice, that might allow a series of mind maps to offer a zooming capability, ranging from an overview to a detailed representation of any specific concept. Remaining gloss relationships depicted both a very close relationship, between 'therapies' and 'treatment', and a more distant one between 'alcohol' and 'substances'.

In contrast to glosses, remaining WordNet relationships arose directly from comparing pairs of words. Hypernyms, in fact, raised a relationship between 'ideas' and 'thoughts' that could be inferred from the group resulting from glosses. Remaining examples were closely related; 'drug' is indeed a form of

'medication', while 'care' will often involve 'treatment'. A further important link arose between 'paranoid' and 'psychotic' that collectively reflect personality disorders (ONS, 2000). Further, the word 'frequency' is closely related in meaning to 'incidence', especially in a medical sense.

Verb-groups were a further source of relationships between words, for example, in drawing together the words 'component' and 'element', 'degree' and 'level' and 'harm' and 'injury'. The second word from those pairs might indeed be used in place of the first, with little affect on truth values. That might not be the case for the words 'want' and 'need', though; we are taught from childhood that those words have different meanings. Although node hierarchies from GRiST mind maps might support or negate such relationships, it is important to remember that mind mapping offers no mechanism for ensuring the veracity of such structures. While it is unlikely that Buckingham and Adams (2006) would have admitted serious misconceptions into the pruned mind maps, support from WordNet ratifies any such groups.

Whereas related words addressed so far will congregate mind map nodes, antonyms from WordNet will keep nodes apart. All the same, it will denote opposing concepts that will further enrich the knowledge base of GRiST mind maps. Experiments reported here revealed good coverage by WordNet of antonyms for verbs, nouns, and modifiers. For example, the verbs 'start' and 'stop' reflect boundaries in time, though of what is unstated. That raises a further reason for keeping such nodes apart, in that whatever starts or stops might be completely unrelated: perhaps stopping taking drugs, as opposed to starting a car. Using such knowledge to generate novel mind maps, though, remains an option, thanks to knowledge held separately. In addition to those verbs, the noun antonyms 'health' and 'illness' revealed a valuable contrast, as did the modifiers 'mental' and 'physical'. Conversely, the modifier 'same' was an antonym both for 'other' and for 'different'; despite semantic distances of zero, those words are but vaguely related.

Similarly tenuous relationships were 'feel' and 'think', 'bother' and 'getting', and 'have' and 'lead'; those were correctly rejected. Although slightly better, the words 'anger' and 'feel', and additionally 'plan' and 'think', might be worth noting. Unfortunately, excessive semantic distances led to declining informative connections between 'disease' and 'psychosis', 'disorders' and 'health', and 'amphetamines' and 'substances'. Further, the rejected relationship between 'calorie' and 'degree' suggests that senses reported by WordNet might be important; that example shows the inappropriate verb-group that arose from related senses of energy and temperature. A more suitable interpretation appeared between 'degree' and 'continuum'.

Clearly, further refinement is needed before accepting or rejecting WordNet relationships; the measures of familiarity and semantic difference are not completely adequate. All the same, the majority of such decisions were appropriate; sufficient reliable relationships arose as to record appreciable new information in CA matrices. That is assessed next by mind maps resulting from such a matrix, for nodes related both

morphologically and semantically.

8.3.3 Generating Combined Mind Maps from HC Clusters

Experiments reported next relate mainly to the stem 'abus', although examples are given of 'suicid' and for 'depress'. Due to the depth of GRiST mind map hierarchies, experiments use an incremental approach that addresses specific levels. Accordingly, results are reported for taking nodes from levels 0 - 2 inclusive, from levels 2 and 3, and from levels 3 and 4. Overall, that range of levels 0 - 4 address key concepts that Buckingham and Adams (2006) saw as formulated close to mind map root nodes.

Matrices for CA were built as for earlier experiments in this chapter. Any specific cell in a matrix, then, recorded that a particular node contained either a stemmed word, or a word having related meaning to any such stemmed word. Further, spelling corrections from Chapter 3 were applied on retrieving nodes. Subsequent to CA, novel mind maps were generated from HC clusters. Labels from such clusters were mapped back to the labels of rows in the original CA matrix, and corresponding texts used in place of HC labels.

The paths of nodes containing related words were reduced to sole representatives, which were combined automatically into novel mind maps for each stem investigated. Resulting mind maps, then, contained just a single instances of paths that might have recurred throughout individual mind maps. The first experiment, then, dealt with related nodes from levels 0 - 2 in paths to nodes having the stem 'abus'.

Method V for related nodes from levels 0 - 2 in paths to 'abus'

The first step, then, was to identify nodes that contained the stem 'abus'. Associated stems from such nodes became column headings for a CA matrix. Rows in that matrix recorded whatever stems were found in any particular node, while row labels recorded the level at which those nodes appeared within any path. Using the methods described in Sections 8.3.1 and 8.3.2, CA was run and the resulting HC structure re-coded as FreeMind nodes.

Further, stemmed words were marked automatically with a single asterisk in resulting mind maps; words related by WordNet hypernyms, verb-groups, or glosses were annotated with a double asterisk, while antonyms were given three. Mind maps were opened in FreeMind, and screen-shots taken to show any resulting node clusters. Detail from resulting mind maps are shown for the stem 'abus' from between levels 0 - 2, from levels 2 and 3, and from levels 3 and 4, in that order. A further set of results reports mind maps that arose for the stems 'depress' and 'suicid'.

Results Va for related nodes from levels 0 - 2 in paths to 'abus'

Figure 8.13 presents detail from a mind map generated automatically from an HC matrix. In fact, just clusters 2 and 3 appear, showing respective clusters of drug abuse and of personality disorders:

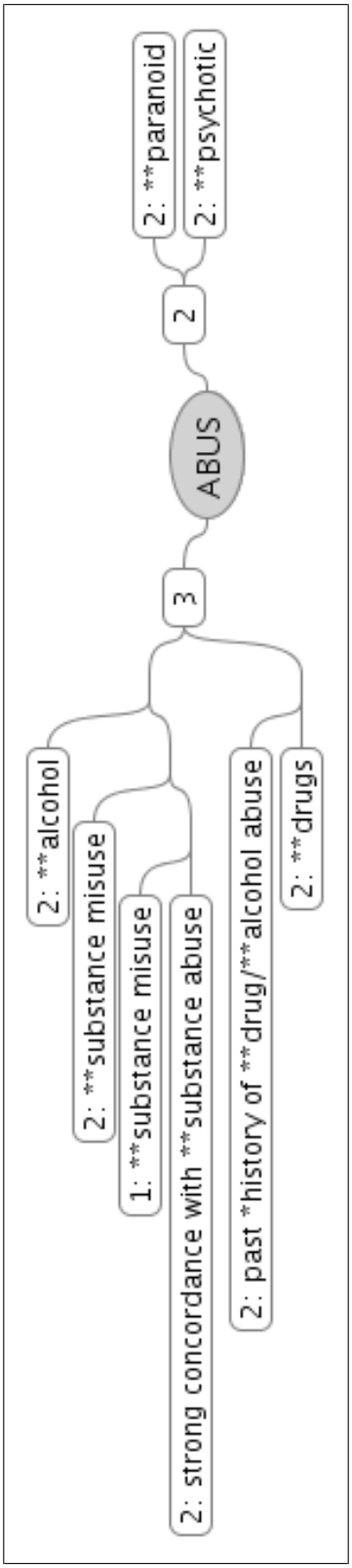

Figure 8.13: WordNet-related clusters between levels 0 and 2, in paths to nodes containing 'abus'.

The stemmed word 'history' from Figure 8.13 is annotated by a single asterisk; remaining related words, though, have two asterisks to show that they arose from WordNet hypernyms, verb-groups, or glosses. Outlier cluster number 2 shows that WordNet reported 'paranoid' and 'psychotic' to have similar meanings. In addition, nodes containing the words 'substance', 'alcohol' and 'drug(s)' were drawn together.

Discussion Va for related nodes from levels 0 - 2 in paths to 'abus'

Figure 8.13, then, shows that nodes expressing forms of drug abuse were grouped together. Although relationships between those nodes were recorded in the node hierarchies of existing GRiST mind maps, WordNet further justified such knowledge structures.

Results Vb for related nodes from levels 2 and 3 in paths to 'abus'

At slighter lower levels 2 and 3, the combination of stems, CA and WordNet produced six clusters for 'abus', of which four appear in Figure 8.14:

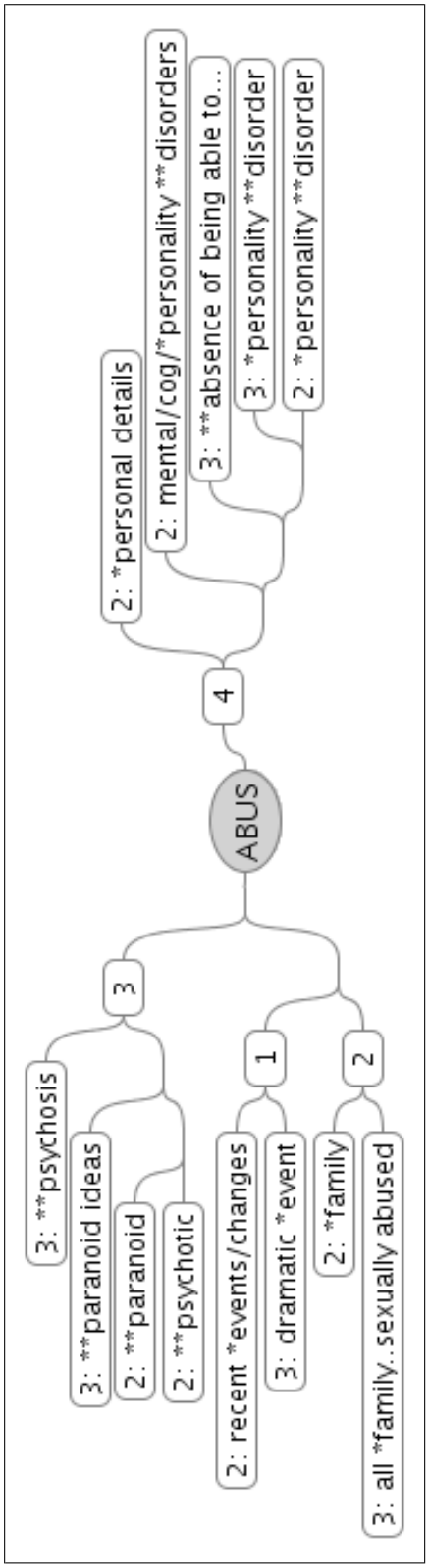

Figure 8.14: WordNet-related clusters at levels 2 and 3, in paths to nodes containing 'abus'.

Nodes for personality disorders at level 2 in Figure 8.13 were related by WordNet; those nodes were augmented in Figure 8.14 by level-3 nodes. Those novel cases, though, took different forms, for example, `[psychotic]` at level 2 was joined by the level-3 node `[psychosis]`. In a similar way, a level-2 node contained the singular 'event', while the plural 'events' appeared at level 3. The further important concept 'family' grouped together nodes from those two levels. Cluster number 4, further to grouping nodes by means of the stem-word 'person' that conflated 'personality', reflected a WordNet relationship between 'absence' and 'disorder'.

Discussion Vb of related nodes from levels 2 and 3 in paths to 'abus'

Although largely suitable nodes congregated as a result of CA on stems and WordNet relationships, the link between 'absence' and 'disorder' was actually inappropriate. Remaining clusters, though, are useful both to humans and to machines. In those respects, generated mind maps respectively constitute useful end products in themselves, and further as structures that contribute to a normalised information base.

In fact, clusters expressing similar concepts arose in a mind map representing levels 3 and 4 to those at levels 2 and 3. Those mind maps, then, are not reproduced here; all the same, a graded approach to node paths from GRiST mind maps yielded structures that might further be fused into a single hierarchy, if needed. Mind maps presented so far, though, arose from trawling mind maps for the stem 'abus'; the next set of results present similar experiments for the stems 'depress' and 'suicid', further important concepts in assessing mental health risks.

Results Vc for related nodes in paths to 'depress' and 'suicid'

Results for this experiment arose from processing nodes that contained the stem 'depress'. That analysis was limited to levels 0 - 2 of any resulting paths. Detail of clusters 2 and 3 from the ensuing mind map are shown in Figure 8.15:

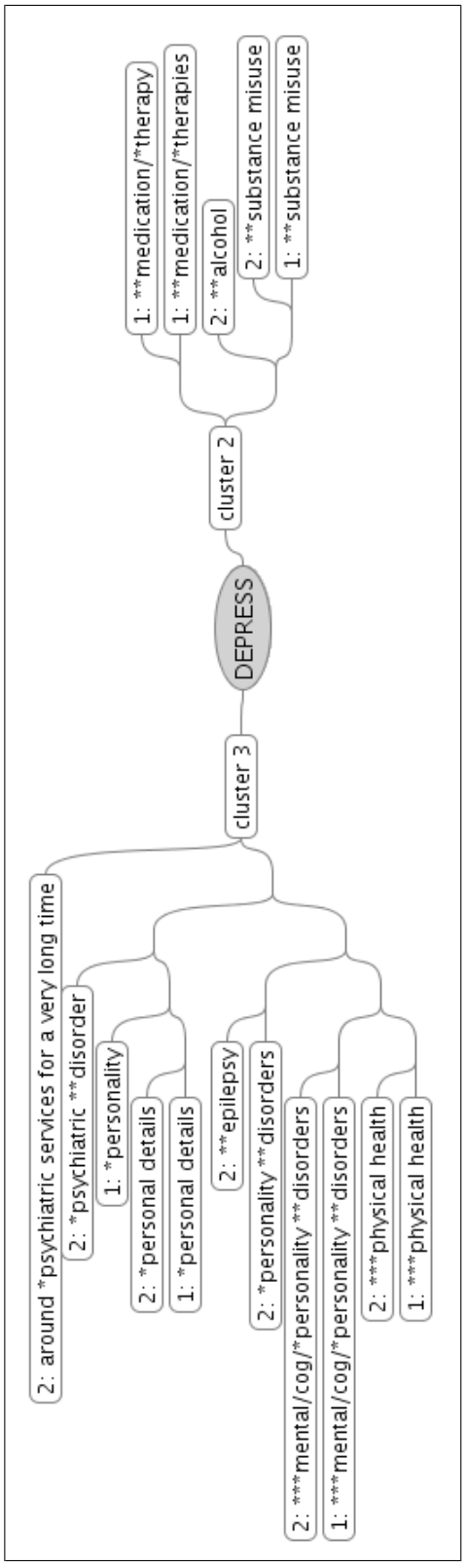

Figure 8.15: WordNet-related clusters at levels 0-2, in paths to nodes containing 'depress'.

Cluster number 3 from Figure 8.15 shows nodes conflated by the stem 'psych', in the form of 'psychiatric', and by 'person' for 'personal' and 'personality'. WordNet further contributed relationships between 'epilepsy' and 'disorders', marked by two asterisks. Further, the antonyms 'mental' and 'physical', which were annotated by three asterisks, marked mental disorders as opposed to physical health. Although all contained within cluster 3, HC separated those nodes into two distinct clusters. Cluster number 2, on the other hand, was the outlier from that CA; in addition to nodes expressing 'therapy' and 'therapies' that were conflated by the stem 'therap', WordNet determined that the words 'medication', 'alcohol' and 'substance' had closely related meanings.

Moving on to results for the stem 'suicid', outlier cluster number 2 and the further cluster number 13 are presented in Figure 8.16:

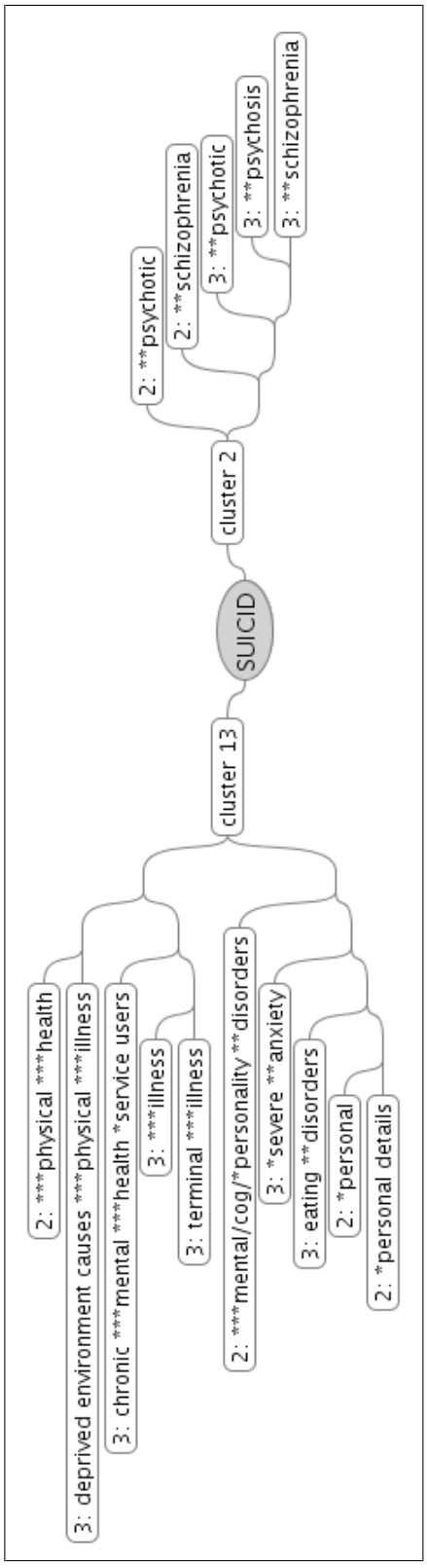

Figure 8.16: WordNet-related clusters at levels 3 and 4, in paths to nodes containing 'suicid'.

In addition to the opposing concepts of 'mental' and 'physical' from the mind map for 'depress', Figure 8.16 shows the further antonym pair 'illness' and 'health' within cluster number 13. In fact, those nodes formed an identifiable sub-cluster, while remaining nodes congregated due to the stem 'personal', and to the related words 'anxiety' and 'disorder'. In addition, the stems 'servic' and 'severe' were detected, though just in single nodes. Outlier cluster number 2, on the other hand, drew together the WordNet-related words 'psychosis', 'psychotic' and 'schizophrenia'.

Discussion Vc of related nodes in paths to 'depress' and 'suicid'

Results from nodes expressing forms of depression and suicide re-stated the validity of clustering nodes by means of weighted CA matrices. That one of the clusters from CA of nodes containing 'suicid' was numbered 13 means that many more clusters arose compared with the mere three from an integer matrix for 'abus'. That, in turn, underlines the improvement made by decimal matrices, which introduced a more useful amount of variation. That was reflected in the 12 sizeable differences in F1 values detected by the algorithm for determining optimum numbers of clusters, which, when added to the default of two, gave a total of fourteen.

Although the word 'substance' might be seen as a word well suited to the upper echelons of the Tree of Porphyry discussed by Sowa (1992), it has a more precise meaning in terms of medication and illegal drugs taken by users of mental health services. Further, antonyms were particularly evident in the mind map for 'suicid'. That relationship failed to arise for the other two stems reported here, because relevant nodes did not appear in paths to the corresponding nodes. Although those antonyms were combined into the generated mind map, they might just as well be omitted, and referenced in the way that will shortly be shown for the main risk categories. Thanks to the way that intensional knowledge is stored separately to the GRiST mind maps themselves, both options remain available.

Of further note is that resulting mind maps for those two stems resembled the one for 'abus'; that was due to way that risk factors repeated across the main categories specified by Buckingham and Adams (2006). Indeed, it is that repetition of risk factors that led to the main categories being omitted from any generated mind maps. For example, the concept of 'abuse' was recorded as a precursor to various forms of self-harm, including suicide, in addition to posing a Risk To Others (RTO) and further, a generic risk factor. Rather, generated mind maps put the stem of any concept of interest at the root, which obviates the need to duplicate nodes across those main categories. The approach taken to indicating relevant risk categories while avoiding duplication is addressed by the final experiments in this chapter.

8.3.4 Generating Normalised Mind Maps

The first part of this Chapter showed that a combination of stemming, WordNet, and Correspondence Analysis generated clusters of related nodes. In that way, representative paths were combined into a single

knowledge structure in FreeMind format. Those mind maps, though, were just the first step towards a normalised information base; nodes were reassembled as they existed in GRiST mind maps, with no attempt made at refinement. The next step, then, transforms clusters from CA into normalised mind maps, to give a single, idealised node hierarchy for any concept under investigation. While succinct nodes might be used as they stand, the process of normalisation raised in Chapter 2 means splitting complex nodes into more definite ideas. As a result, novel nodes might arise; further, nodes might contain words that did not participate in CA, yet must be accommodated in any normalised hierarchy.

A further problem of representation concerns risks such as self harm and suicide that appeared directly under the root nodes of GRiST mind maps, in line with the template that Buckingham and Adams (2006) provided. Along with further categories such as RTO and a generic risk, that template led to panellists repeating lower-level concepts under such top-level categories. Normalisation, though, aims to remove such duplication; it follows, then, that sole instances of mind map concepts must reflect associations with several possible risk categories. Further, resulting mind maps must be amenable to interpretation both humans and by machines.

In fact, an existing facility of FreeMind helps to resolve those difficulties, by means of arrows between nodes. Links may be unidirectional or bidirectional, with the latter having arrow heads at both ends. In short, arrows links act as pointers between mind map nodes in a way that resembles links between WordNet's synsets; pointers in WordNet, though, are for use by machines, whereas arrows in mind maps further denote related concepts for human viewers. As a result, risk categories will be held as a separate group of nodes branching from the root of any generated mind map. All the same, correspondence with lower-level risks will be maintained by means of FreeMind's arrow pointers.

The problem remains, though, that top-level risk factors sometimes comprised more than just a sole idea; such was the case for the nodes `[self harm/suicide]` and `[mental/cog/personality disorders]`. Indeed, discrete nodes already exist in GRiST mind maps for `[self harm]` and `[suicide]` as separate risks. The node `[self neglect]` further augmented that group. Although previous experiments grouped those nodes together, the problem remains of determining optimal node hierarchies for such conjoined and partially matching concepts. While not wishing to reduce nodes to bags-of-words, nodes that share important words might be re-written with that linking word branching into nodes for associated words.

For a similar reason, lower-level nodes such `[client episodes]` and `[client presentation]` might be split into a node for `[client]` that branches both to `[episodes]` and to `[presentation]`. There were, though, nodes that already expressed such atomic concepts; all the same, a problem arose due to such nodes appearing at various levels in paths. Take, for example, the node `[history]` that occurred at both levels 1 and 2 in paths to nodes conflated by 'abus'; that arrangement exemplified the structural variation

described in Section 6.1 of this chapter. The first normal form, though, requires that node to occur just once; in such cases, the node at the highest level superseded those at lower levels.

Further knowledge available for any idealised node structures came from relationships reported by WordNet. Nodes related by hypernyms, say, might introduce novel words that force existing configurations to be re-assessed. Antonyms, though, will not be used, here; rather, they will be presented by means of an interface to be revealed in Chapter 10. It is vital, though, that generated mind maps reflect any WordNet relationships in a way that makes clear their origin in the experiments described here. A solution lies in so-called anonymous nodes that carry no text. Such empty nodes should not, though, be confused with blank nodes that currently exist in GRiST mind maps; rather, anonymous nodes are formal components of semantic networks.

Anonymous nodes, according to Berners-Lee (1998), act as conjunctions. In normalised mind maps, then, the children of any such node are deemed to be related by an implicit conjunction; as Berners-Lee (1998) puts it, any anonymous node

> '... implies an implicit variable existentially qualified in the scope of the conjunction.' (p.1.)

In other words, children of anonymous nodes express related sub-concepts by means of 'and'. Although mind map nodes of whatever type fulfil that purpose, anonymous nodes allow a more formal representation of any derived relationships. Whereas blank nodes from GRiST mind maps reflected just that mind map authors neglected to add text when creating nodes in FreeMind, formal anonymous nodes offer an explicit way of representing nodes related by knowledge from WordNet, both to human viewers, and to machines treating those mind maps as an information base. The origin of such arrangements, though, will be retained as justification; machines and humans alike will be able to draw on such knowledge, rather than assuming any empty node to be formally anonymous.

Further assistance in remodelling GRiST mind map nodes comes from experiments on prepositions reported in Chapter 7. In particular, the word 'of' revealed an interesting property, in that words to either side of that preposition were sometimes found reversed and contiguously. These final experiments, then, show the culmination of experiments in spelling correction, stemming, and the CA of both prepositions and of mind map paths. In fact, two mind maps are generated for any given stem; the first displays node paths exactly as found in GRiST mind maps, but combined into a single hierarchy, while the second type results from applying knowledge gathered here. Such pairs of mind maps, then, constitute 'before' and 'after' versions in the process of generating normalised mind maps about mental health risks.

Method VI for generating combined mind maps

This experiment shows the creation of a combined mind map from paths to nodes that contained 'abus'. As previously in this chapter, representative paths were obtained by loading CSV strings into a Java `TreeSet` object; the resulting unique paths from mind maps were processed sequentially. Novel nodes were created on the first encounter of any key; subsequently, child nodes were created or appended at appropriate levels.

Results VI for a combined mind map of paths to 'abus'

Figure 8.17 shows the result of combining the top three levels of all forty six GRiST mind maps, for paths that led to nodes expressing concepts related to 'abuse':

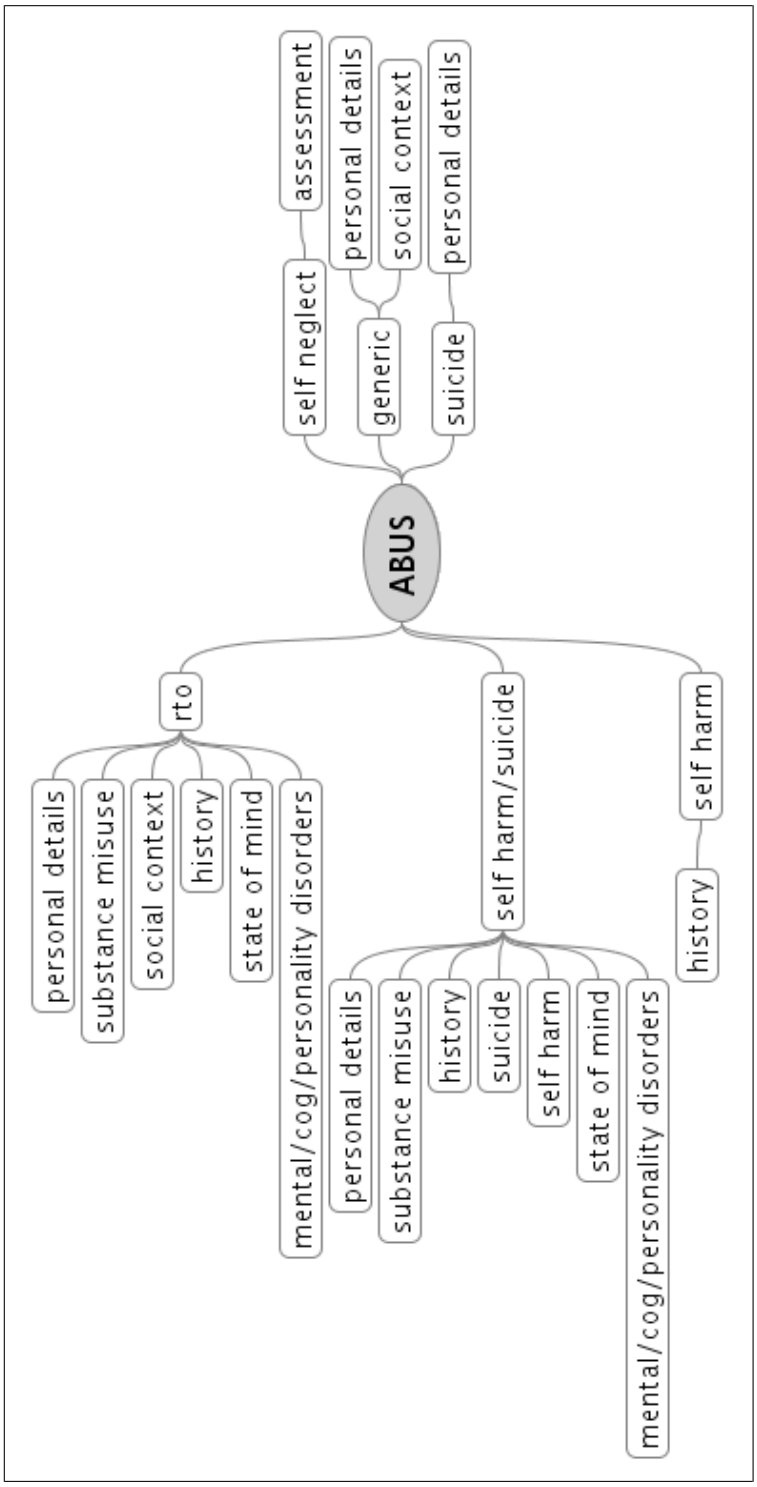

Figure 8.17: Summary of paths between levels 0 and 2, to nodes containing 'abus'.

The generated mind map from Figure 8.17 shows paths that led ultimately to nodes which contained the stem 'abus'. Main risk nodes branch from the root node, which was labelled to reflect the stem under investigation. For each of those main risks stems a further level of nodes that indicate related narrower mental health risk factors.

Discussion VI of a combined mind map of paths to 'abus'

The combined mind map in Figure 8.17 is primarily meant for human use, in that it depicts the domain of knowledge to be normalised for the concept 'abuse'. That in itself, though, might prove a useful end to these experiments: GRiST researchers and panellists could absorb such summaries at a glance. Further, the problem of nodes repeating under various main risks is well demonstrated by the mind map from Figure 8.17. For example, the node `[mental/cog/personality disorders]` appears under both `[rto]` and `[self harm/suicide]`. In addition, the node `[personal details]` occurs under all main risks except `[self harm]` and `[self neglect]`.

Indeed, text from nodes expressing various forms of self-harm overlapped; in addition, the related, though separate, concept of suicide was embedded in a longer node. That was further the case for `[mental/cog/personality disorders]`; one of those words, 'personality' was conflated by the stem 'personal' that appeared as an actual word in the node `[personal details]`. That suggests breaking up that longer node, and moving 'personality' to a novel node group centred on the stem 'personal'. The normalisation process imposed here alleviates that problem, and generates mind maps that act both as a useful aid to human interpretation of mental health risks, and further as an information base for machines. Generating such combined and normalised mind maps is addressed by the following experiments.

Method VII for generating normalised mind maps

The first step was to identify fully and partially contained concepts in mind map nodes. That was done by means of a bespoke Java class called `NormalisedMindmap` that was responsible for building the knowledge structures reported here. The main input to that programme was a list of clusters derived from the HC matrix produced by CA. Rather than just re-combining paths, though, clusters from CA were inspected for nodes having words in common, and results stored in a further novel Java class called `SplitNode`.

Individual `SplitNode` objects, then, were compiled as follows. First of all, words from any two nodes under comparison were reduced to individual words by means of Java's `Split()` method, as was done in previous experiments. Within those lists, the positions of prepositions were noted, as were those of any additional stems to the one under scrutiny. Should any node-pair have one or more stems in common, the positions of those stemmed words were recorded, and from there, the indices of words that preceded or followed them. Further, the keys of the nodes concerned were stored, with the first identifying the longer node of any pair, and the second key for the shorter node.

In addition, `SplitNode` objects were made sortable by implementing Java's `Comparable` interface, and by further overriding the `compareTo()` method; that allowed the static method `Arrays.sort()` to arrange lists of `SplitNode` objects into the following order:

1. the concept that was common to any pair of nodes,
2. the number of words in any shared concept,
3. any words that preceded such shared concepts, and
4. any words that followed them.

Sorting lists of `SplitNode` objects allowed subsets of results to be identified; nodes from such subsets held a stemmed concept in common. In addition, sorting on the number of words per concept forced shorter ones to the top of each subset. In that way, shorter or atomic concepts were processed first.

Results VII for a normalised mind map of paths to 'abus'

Figure 8.18 presents the `SplitNode` objects that arose from comparing nodes 11176, 11616, 10209 and 11156, that is, `[suicide]`, `[self harm]`, `[self neglect]` and `[self harm/suicide]` respectively:

```
2 full match(es) by containing ID:

   11156 (11176) --> [self, harm] [SUICIDE] (via [SUICID] )

   11156 (11616) --> [SELF, HARM] [suicide] (via [SELF, HARM] )

2 partial match(es) by concept length:

   11616 (10209) --> [SELF] [harm] (via [SELF] )
   10209 (11616) --> [SELF] [neglect] (via [SELF] )

2 separate node(s) fully contained in 11156 [self, harm, suicide]:
   11176: [suicide]
   11616: [self, harm]
```

Figure 8.18: `SplitNode` objects for related nodes.

The first `SplitNode` object in Figure 8.18 arose from comparing the nodes `[suicide]` and `[self harm/suicide]`. Bracketed entries at the end of the first two objects show what stems conflated the concepts of 'suicide' and of 'self-harm'. In such cases, no words remained after accounting for those embedded concepts: shorter nodes were fully contained in a combined version. In contrast, the partially overlapping nodes `[self harm]` and `[self neglect]` came next in Figure 8.18. The final entries in that figure show the two existing concepts that were further found combined into a single node; remember, though that the second of those, `[self harm]`, partially overlapped with `[self neglect]`.

Sorted `SplitNode` objects from the top three levels of all forty six GRiST mind maps were combined into the mind map shown as Figure 8.19:

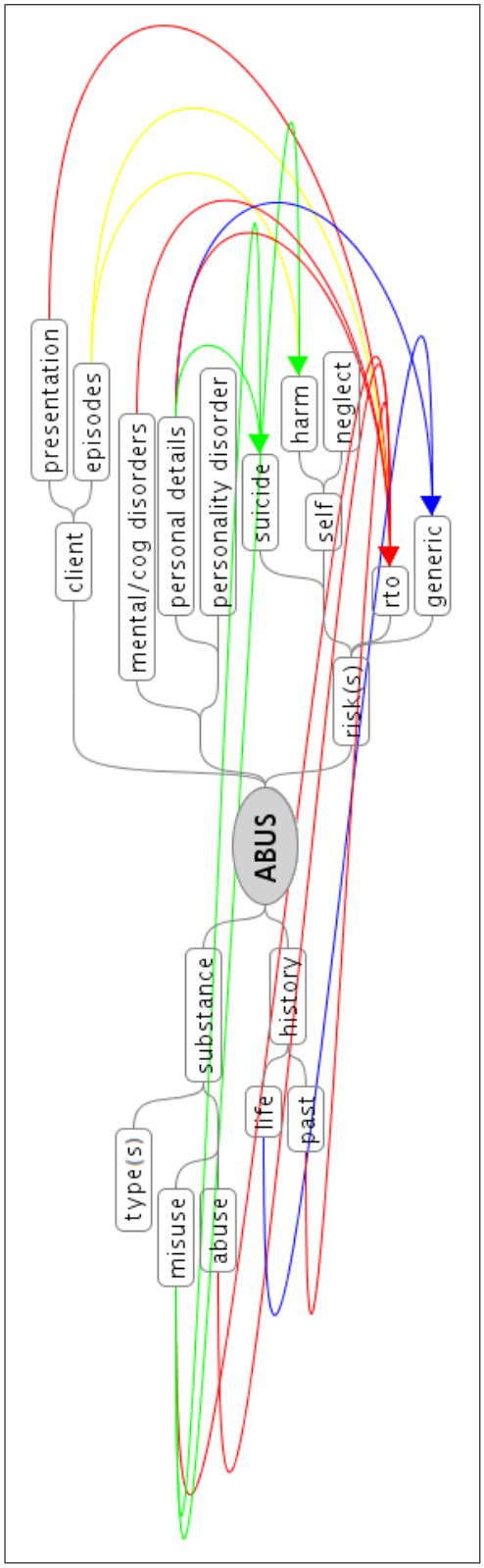

Figure 8.19: Normalised mind map for paths between levels 0 and 2, to nodes containing 'abus'.

The nodes [substance abuse] and [substance misuse] from the top-left of Figure 8.19 gave a common parent of [substance] branching into the child nodes [abuse] and [misuse]. An intervening anonymous node depicted a reliable WordNet relationship between hypernyms. Although anonymous nodes are hard to discern in FreeMind, the one in question occurred at the same level as did the branch from [substance] to [type(s)]. Conversely, the [history] node branched directly into [life] and [past] in the absence of any dependable WordNet relationship between those child nodes.

In fact, the arrangement of a parent node [substance] branching into the child [type(s)] arose from the node [types of substances]; an heuristic gleaned from experiments in Chapter 7 allowed that node to be automatically reconfigured. Specifically, words to either side of the preposition 'of' might be reversed, and the preposition itself removed. In that way, the phrase 'types of substances' was rewritten as 'substance type(s)', using the singular forms conflated elsewhere in GRiST mind maps by corresponding stems. That was further justified by the suffix '-s' that arose from experiments reported in Chapter 4. As a result, the word 'type' was automatically appended with that suffix enclosed in parentheses. The resulting format bore comparison with existing nodes that started with the word 'substance', giving the final sub-hierarchy.

At the top-right of Figure 8.19 appears a group headed by the node [client] that reflected structures from separate GRiST mind maps. On one hand, [client] had the child [assessment], while on the other, [history] appeared as the child. The resulting hierarchy arose from simply joining those paths, and eliminating the duplicated [client] node. The next group down the right-hand side reflected the morphological relationship between 'personal' and 'personality' found in Chapter 4, which resulted in a further anonymous node. As a whole, that sub-hierarchy arose from automatically stripping the word 'personality' from [mental/cog/personality disorders] to leave [mental/cog disorders]. The removed word was joined with 'disorder', the stem of 'disorders' from the parent node's original text, to give the novel node [personality disorder].

The bottom-right of Figure 8.19 further shows a single hierarchy of risk categories, with an anonymous node above the nodes [suicide] and [self harm] to show their origin in splitting the longer node [self harm/suicide]. In turn, the nodes [self harm] and [self neglect] were rationalised in the same was as was the group headed by [client]. The risks [rto] and [generic], on the other hand, branch directly from the [risk(s)] node, and were otherwise unaltered. Relationships between top-level risk factors and those at lower levels were shown by arrow pointers, generated automatically. For human viewers, coloured arrows reflected particular main risk categories; in that way, related risks of substance abuse and misuse were related to the top-level [rto] node by means of red arrows; that main risk factor was at the top of corresponding paths which yielded the novel group.

In a similar way, both children of [history] pointed to [rto]. Of child nodes for [client], though, just [presentation] pointed to [rto]; that pointer is shown in yellow to identify a group that related to multiple main risk factors. The remaining child [episodes] was related to the structure for 'self harm' by means a further yellow arrow. Additional arrows from Figure 8.19 were coloured green to indicate related main risks of suicide and self-harm, while remaining pointers to a generic risk factor were shown in blue. Note, though, that the risk of self-neglect has no arrows leading to it; that was in order to improve clarity by showing just selected pointers.

Discussion VII of a normalised mind map of paths to 'abus'

A combination of spelling correction, stemming, CA and WordNet, then, culminated in a combined mind map for the concept of 'abuse'; indeed, any of the stems identified in Chapter 4 might drive that process. For the stem in question, additional stems such as 'alcohol' and 'suicid' arose in nodes from paths to those containing that driving stem. Further, WordNet indicated relationships between nodes by means of shared meanings, instead of by conflated stems. Rather than discrete nodes, though, entire paths were related in that way; further reducing such paths to CSV strings of representative keys eliminated duplication, in the spirit of the second normal form, 2NF.

Although the aim was not to reduce mind map concepts to a BOW, certain nodes were reconfigured in that way. That happened, though, just when shorter nodes shared a particular word, which justified moving any remaining words to child nodes. Importantly, the combined node of [self harm/suicide] was reduced to separate nodes [self harm] and [suicide] as children of the inserted [risk(s)] node. The resulting [self harm] node was further decomposed into the parent node [self] and its children [harm] and [neglect]. That was further the case for the word 'personality' in the combined concept of [mental/cog/personality disorders]. Rather than splitting words, though, the removed word in that case was combined with 'disorder' to yield a novel node. Although the stem 'person' conflated 'personal' and 'personality', those differing words were kept in their respective nodes, rather than generating further children as was done for the connecting word 'self'. By means of splitting multiple concepts, then, resulting nodes conformed to the first normal form, 1NF, that eliminates repeating groups from tuples, rather than from relations.

Normalising GRiST mind maps further demanded a way of representing lower level risks that repeated across several main categories. That problem was overcome by means of arrow pointers that are allowed by FreeMind. In addition to colour-coding such links for human viewers, arrows are encoded in a format amenable to machines, details of which will appear in Chapter 9; for now, it suffices that machines have a way to interpret such links, and in turn, any related nodes from separate sub-hierarchies. Indeed, that facility provides the cross-links that Novak and Cañas (2006) claimed to be lacking from mind maps. In

addition to providing a visual aid for humans, though, pointers here furthered the use of GRiST mind maps as a normalised information base[1].

8.4 Chapter Discussion

The first experiment in this chapter involved retrieving representative paths to nodes from GRiST mind maps; the resulting subset of paths reflected sole occurrences of particular paths. Those unique paths fed the subsequent experiment that applied CA to GRiST mind maps. To that end, clusters of related paths were revealed by using a matrix of columns representing any stems from those paths, and of node identifiers as rows. That analysis, though, proved just partly successful; although clusters were produced, humans viewing those results would see one large cluster in particular as comprising several smaller ones. That was due to a lack of variation in the underlying matrix, which was overcome by using weighted decimal values instead. As a result, that large cluster was successfully split into three separate, smaller clusters.

In practice, a slight problem arose from the actual weightings employed. The approach was to adjust raw observations by a proportion derived from the column index of any cell; that, though, led to clusters that reflected the numbers of stems contained in participating nodes, in addition to the identity of any particular stem. Further research, then, is needed in order to refine weightings in a way that emphasises the values of stems, rather than the numbers of them. All the same, just a sole unsuitable cluster arose in the examples reported; that was for the stem 'violen', which appeared in two clusters from the same CA. Rather than too little variation, that case reflected excessive amounts introduced by weightings.

Although GRiST mind maps contained deep paths in excess of 10 levels, just the top three levels were addressed by those first two experiments; that was to restrict analyses to any key ideas that Buckingham and Adams (2006) saw as formulated close to mind maps' root nodes. A further experiment, though, performed CA on the lower levels 3 and 4. That analysis revealed nodes that repeated at both of those levels; in such cases, the highest-level instance contributed to the resulting normalised mind map. In that way, hierarchical variations between individual mind maps were resolved. Future research will combine overlapping groups from successively lower levels into a single, normalised mind map. A possible approach might be to run further CA on levels to either side of each main group, to reveal normalised structures by which to join consecutive sub-hierarchies from the main analyses.

In addition to paths to nodes that were conflated by stems, WordNet supplied further related words for which no stem existed. While that approach successfully identified additional related nodes, the means of

[1] In future research, problems caused by colour-blindness in humans suggest using arrows of more suitable hues.

recording such knowledge in CA matrices might be improved. Experiments reported here treated words related by WordNet as if they were conflated by a corresponding stem, which came from one of the words of any pair under comparison. In the future, it would be preferable to seek a separate representation of such additional knowledge; that, though, is hindered by CA's restriction to just two variables. Existing and new research has shown that extra variables might be encoded in row and column labels; even so, any more than one such variable might raise difficulties in interpreting CA results.

Results from CA augmented by WordNet showed a need for further refinement, in that large semantic distances led to declining informative relationships that would have contributed greatly to the resulting mind maps. Conversely, occasional inappropriate relationships would better have been rejected. The two measures applied, familiarity and semantic difference, were not completely adequate in such cases. Paying more attention to particular WordNet senses suggests one additional source of information; further, a later version of WordNet might offer more current word usage.

In addition to variations on the concept of 'abuse', the stems 'depress' and 'suicid' yielded further combined mind maps. In fact, those generated mind maps closely resembled the one for 'abus', due to all three concepts repeating across the main risk categories. There were, though, slight differences between results from those three stems. For example, the antonyms 'illness' and 'health' arose just from analysing 'suicid'; results from this approach, then, depended on the particular nodes, and from there, the paths that were processed for any given stem. That, in turn, suggests pooling knowledge gleaned for individual stems, once those analyses are complete.

Having combined CA with stemming and WordNet, further experiments created normalised knowledge structures as mind maps. The first such experiment, though, produced a combined mind map directly from representative paths isolated in previous analyses; that un-normalised mind map gave an overall view of the concept 'abuse'. Particular concepts, though, continued to spread across the top-level risk categories, in the same way as in individual mind maps. In contrast, a normalised form of that mind map removed those main risks to a separate group, referenced by FreeMind's pointers. The resulting cross-links enriched the emerging information base with knowledge over and above local hierarchical associations. In that sense, rather flat mind map structures evolved into semantic networks more resembling WordNet's verb network, which Fellbaum (1990) reports to be richer in such sideways connections than is the network of nouns. Rather than POS, though, pointers here depict related mind map concepts.

Of particular importance in this approach was the role of HC in providing a precursor to mind maps encored in FreeMind format. Although that entailed a degree of justified reconfiguration, node hierarchies in the resulting combined mind maps largely reflected those from HC. In fact, more detailed consideration of successive sub-clusters merits further research, as does accounting for the correlations

and contributions reported by CA. For examples, correlations indicate the degree of association between paths and corresponding concepts, which might be reflected in, say, the colour or style of nodes. Such gradations might further help in identifying dominant concepts within any cluster, and from there, in deriving appropriate novel nodes from combined concepts.

Overall, then, the process that culminated here started in Chapter 3 by correcting spelling errors, and identifying valid novel words that augmented the list of GRiST concepts. After that, Chapter 4 identified stems in GRiST mind maps in a way that depended solely on morphological similarities, rather than on linguistic rules. Subsequently, Chapter 7 revealed POS that congregated around certain prepositions, which helped here in reconfiguring nodes into a simpler, standardised format. In that respect, the way in which FreeMind stores mind maps was helpful; accordingly, that is the topic of the next chapter.

8.5 Chapter Summary

The introduction to this chapter described variations between forty-six GRiST mind maps that recorded interviews with mental-health experts. Those individual mind maps were subsequently integrated into a single, combined version; that demanded great manual effort, due to variations in representing key ideas. The chapter went on to describe CA and WordNet as the tools for refining those GRiST mind maps automatically, by means of Hierarchical Clustering (HC) from which novel, combined mind maps were generated.

To that end, CA matrices reflected nodes that contained stems from Chapter 4. Subsequently, such CA matrices were supplemented by relationships suggested by WordNet. From that combination of HC and WordNet arose mind maps that reflected paths to related concepts. That was demonstrated by two sets of experiments in refining mind map structures, starting with nodes related in terms of words conflated by stems. Simple contingency tables, though, failed to yield appropriate clusters; that was rectified by introducing further variation by means of weightings and decimal observations. Subsequent experiments showed that approach to be more discerning.

Paths to nodes that contained the stem 'abus' were analysed between various levels in paths from GRiST mind maps. Although main risk categories were noted, they did not appear in any generated mind maps, so as to satisfy normal forms that weed out duplication. Supplementing matrices by means of WordNet relationships such hypernymy and antonymy yielded mind maps that drew together paths to concepts related by meaning rather than by any stems. In addition, glosses identified further relationships.

The last experiments reported in this chapter involved generating normalised mind maps from such combined versions. Relationships between nodes from those mind maps were indicated by FreeMind's arrow pointers; main risk nodes appeared as a separate sub-hierarchy of nodes, to which remaining nodes referenced by means of such pointers. Resulting knowledge structures revealed an overall picture of the concept of 'abuse' in those mind maps. In addition to benefiting human mental health experts and GRiST researchers, those mind maps constituted a normalised information base of cues and risk factors. Finally, an overall discussion of experiments reported here was offered, before this summary closed the chapter.

Part III

Implementation, Summary and Conclusions

9

Mind Maps, Metadata and an XML Database

9.1 Introduction

The thread for this chapter is the eXtensible Mark-up Language (XML) that FreeMind uses to hold mind maps. In fact, that format has become a standard way of storing and transmitting digital information; any machine able to process XML can, in turn, communicate easily with other machines. Indeed, XML offers a very straight forward route to such inter-operability. That benefit, and any further advantages, are presented first section of this chapter. Following that comes a discussion of 'metadata' that comprise data about data, rather than raw information in its own right[1].

Following that overview, the problem of storing knowledge as XML is addressed; specifically, a so-called native XML Database permits an approach based almost entirely on XML. That database will hold both GRiST mind maps and any metadata collected during experiments reported earlier. The mind maps partition of the database will enable single GRiST nodes or even complete paths to be retrieved by machines. Metadata held in a separate area of the database will, in a similar way, be conveniently stored and readily queried and updated, without the need for a 'traditional' relational database. Subsequently, the implementation of such a sub-system for GRiST is considered, before the chapter closes with a summary. Firstly, then. to an overview of XML and of its role in GRiST.

[1]Note that the singular form of 'data' is reluctantly used, for compatibility with references to that plural noun.

9.2 The Advantages of XML

XML has emerged as a standard for holding information in a machine-readable format, which has reconciled diverse computer platforms. XML promotes such interoperability by allowing machines to readily exchange information, in the form of *elements* that provide a standard format for holding data. Files of such elements allow machines to share information freely, particularly over the Internet. Indeed, the word 'extensible' refers to *tags* that might be defined for any particular dataset; novel tags would be created for whatever elements are required. In addition to elements themselves, tags specify names for attributes attached to such elements.

Together, elements and attributes within so-called XML documents describe a flat-file database. However, the very flexibility of XML hinders attempts any automatic modelling of ideas expressed by such elements. Even in simple record-orientated files, differing element names often obscure relationships that are obvious to humans. Document type descriptions (DTDs) and XML Schemata remedy that problem by specifying valid element names and combinations (Fischer, 1998). In other words, DTDs and schemas provide intentional knowledge about what form records might take in any given XML document. FreeMind does not apply such restrictions to mind maps, other than to ensure conformity to just a sole XML element, which is allowed a small number of attribute types. In fact, Polansky and Foltin (2010) do provide a DTD for FreeMind; that DTD, though, governs largely the display of mind maps, rather than any content.

XML further provides a means of storing the variety of data found, say, on the Internet. That poses a problem for highly-structured formats; relational tables, for example, are strict in their specification of data types and column names. The Internet, though, lacks such a regular structure; formats in which data are stored might differ greatly, although with a degree of overlap. Such differences in the structure of information mean that XML represents semi-structured data (Thuraisingham, 2002).

The Format of FreeMind XML

In fact, FreeMind stores mind maps as XML. Accordingly, mind maps can be parsed by programs other that FreeMind, in order to extract data structures. Although FreeMind stores node hierarchies in files suffixed '.mm', changing that suffix to '.xml' allows mind map to be processed as pure XML. In fact, GRiST did not use such XML directly, due to FreeMind's focus on ways of displaying mind maps. Instead, various programs stripped away any formatting instructions, with the resulting XML files reflecting just the tree structure of GRiST mind maps. Indeed, any base XML file might be displayed in various ways, depending on the particular requirements of human viewers. In that way, interoperability was achieved between components of GRiST (Buckingham & Adams, 2006; Buckingham, Ahmed, & Adams, 2007).

FreeMind, though, stores text not in elements themselves, as is the norm, but in attributes; that format does not directly reflect mind map concepts and sub-concepts, as would a hierarchy of differentiated, meaningful elements. Rather, a loose structure of 'node' elements store short phrases, or even sentences, as attributes. There is, then, little differentiation between records, as can be seen from the FreeMind XML in Figure 9.1. Rather than the refined XML by which Buckingham and Adams (2006) demonstrated the tags employed by FreeMind, the approach here is to use XML directly from GRiST mind maps. Accordingly, the left-hand side of Figure 9.1 shows a mind map created by GRiST expert number 2, while the right-hand side depicts the corresponding XML, as viewed in the Mozilla web browser:

```
−<map version="0.7.1">
 −<node TEXT="expert 02">
  +<node TEXT="Suicide"></node>
  −<node TEXT="Self Harm">
     <node TEXT="history"/>
     <node TEXT="state of mind"/>
     <node TEXT="personal details"/>
     <node TEXT="intention"/>
   </node>
  +<node TEXT="RTO"></node>
  +<node TEXT="Self Neglect "></node>
  +<node TEXT="Generic"></node>
  </node>
 </map>
```

Figure 9.1: Detail from a GRiST mind map, with the corresponding XML.

Although the XML from Figure 9.1 came directly from FreeMind, GRiST researchers' description of their converted form largely applies. The `<node/>` tag, then, depicts individual nodes from a mind map hierarchy[1]. FreeMind nodes from Figure 9.1 start with an angled bracket '`<`' followed by the name of the XML element, 'node'. An oblique stroke followed by a closing angled bracket means that any such node has no children; should that stroke be absent, nodes remain 'open', with further nodes nested inside them. Accordingly, open nodes of the form `<node>` are closed by corresponding `</node>` tags later on in the XML document (Buckingham, Ahmed, & Adams, 2007).

Node hierarchies from FreeMind, then, correspond to nested XML elements having the `<node>` tag; on any given element, '`TEXT`' attributes specify nodes' contents. Further, nodes might be 'open' or 'closed', or in FreeMind parlance, 'folded' or 'unfolded'. For example, the root node `[expert 02]` from Figure 9.1 has an open `<node>` tag, having no oblique, but just a closing angled bracket. The same goes for immediate children of that root; XML for main risk nodes such as `[suicide]` and `[self harm]` contain sub-hierarchies of increasingly specific cues and risk factors. In fact, the `[self harm]` node was unfolded to one level in Figure 9.1 to depict leaf nodes such as `[history]`, which has a terminating oblique to close it. The penultimate line showed the root node `[expert 02]` closed by an explicit '`</node>`' tag, before a '`</map>`' tag closed the entire mind map, by balancing the '`<map>`' tag from the first line.

An Approach Centred on XML

Figure 9.1, then, showed nested nodes of identical type that define hierarchical mind map structures; information about those nodes was further held as attributes. In fact, XML allows any number of different attributes, which further allowed unique expert IDs held in nodes to be recoded as discrete attributes. Programs subsequently retrieve information about any given node from the associated attributes (Buckingham & Adams, 2006). To that end, web browsers have so-called parsers that allow programmers to develop software around XML (Wilde, 2006; Buckingham, Ahmed, & Adams, 2007).

The success of XML arises from its independence of particular platforms and languages; in recent years, XML has increasingly comprised the core of applications, particularly with regard to database systems. XML, by means of appropriate element tags, is able to represent structures having a relational structure, in addition to the noted facility for representing semi-structured information. That power, along with the growth of the Internet, has given rise to an approach called 'XML-centric development', which refers the ubiquity of XML throughout such systems. Although XML technologies are not yet mature, machines might rely on XML as the sole mechanism for storing and transporting data (Wilde, 2006). An XML-centric approach, then, will allow components from this thesis to integrate seamlessly into the existing GRiST web site. Further, XML is ideally suited to storing additional information that

[1] Note that XML nodes are depicted in the same font as have been mind map nodes in earlier chapters.

explains any raw data; such extra knowledge comprises 'metadata', to which attention turns next.

9.3 Metadata: Data about Data

The word 'metadata' comes from joining the two words 'meta' and 'data'. In fact, 'data' refers to information of any form, whether on paper or held electronically, while the Greek word 'meta' means 'transcending' and 'going above and beyond'. Accordingly, 'metadata' means data that describes other data ("Computer Dictionary", 2010). Indeed, the future of information systems might depend on such data about data. Although metadata is far from a recent idea, it has generally been used for specialist, one-off purposes, in human-readable formats. The rise of the Internet, though, has led to a more formal approach to metadata as a means of finding relevant information, and for integrating sources from varying computer platforms and database management systems. In that respect, metadata resembles the Rosetta Stone that translated between the Greek, Demotic and Hieroglyphics languages (Jeffery, 2000).

Types of Metadata

Metadata, then, describes data sources such as files and relational databases, and in addition any records within those repositories. Attaching metadata provides a means whereby machines might interpret any associated data. In practice, metadata reflects the structure of underlying information, and assists in distilling knowledge from data. In addition, metadata helps to refine queries so that they return what users actually request; metadata might further explain to users the processing involved in answering queries. There are, though, three types of metadata. The first type is schema metadata that dictates what can be held in any given information base. Associative metadata, on the other hand, links records together, even though such connections might be unclear in any underlying data. The last type of metadata is the navigational variety, which describes routes between schema and associative metadata (Jeffery, 2000).

The term 'schema metadata' well describes the intensional knowledge introduced in Chapter 2; as was noted, though, that form of knowledge is usually created before any actual records. Due to mind mapping's lack of schema metadata, such intensional knowledge was generated from extensional knowledge encoded as GRiST mind maps. Corresponding metadata views mind maps as relations, and nodes as tuples, which gave rise to a structure based on tuples such as $nodeID \rightarrow nodeID$ and $nodeID \rightarrow concept$. By means of such tuples, metadata arising from in GRiST mind maps will indicate specific instances of related nodes and concepts.

Such metadata reflect any results retained from CA and WordNet; viewed as associative metadata, such knowledge served to links records together, as Chapter 8 showed in novel mind maps created from

HC clusters. Further associative metadata for spelling corrections Chapters 3 allows corrected words to be retrieved on demand, rather than applied to existing nodes. In addition, stems from 4 reflect differing forms of general concepts contained in GRiST mind maps. As Jeffery (2000) noted, associative metadata further helps users to visualise and explain information. In that spirit, FreeMind's arrow pointers, introduced in Chapter 8, depicted relationships across nodes, and augmented the strict hierarchical nature of mind maps as demanded by Buzan (2003). As well as drawing together mind map concepts, associative metadata from experiments reported earlier will allow users to review evidence for automated decisions, by means of metadata about metadata.

Although the name 'navigational' suggests actively helping users to find resources, that type of metadata in fact serves just to map between schema metadata and the associative type. Experiments reported earlier have shown that role to be filled, in part, by bespoke Java classes; such objects mediated between intensional knowledge held as metadata and extensional knowledge in GRiST mind maps. In that way, key concepts from GRiST mind maps will be mapped while individual nodes are left unchanged. Such mappings, then, will be held separately as metadata. Accordingly, the next section addresses a so-called native XML database that will hold such knowledge in discrete compartments.

9.4 Xindice: a Native XML Database

Treating XML documents themselves as a flat-file database suffers a major drawback: it lacks any innate facility for accessing such data. In fact, the importance of XML has led to databases that store data in that format; Xindice is one such native XML database[1]. In addition to offering a convenient repository for XML documents, Xindice provides a means of searching and retrieving such data (The Apache XML Project, 2007).

Despite holding just semi-structured data, any XML database must support the basic functions of a Data Base Management System (DBMS), such as query management, and processing updates. Any XML database should further provide a means of managing both actual data and metadata (Thuraisingham, 2002). Xindice meets those demands by providing a basic command-line interface that works under both Windows and Linux. The first step in creating a database is to define *collections* through that command line. Collections are the basic unit of storage within Xindice; in that respect, Xindice collections correspond to tablespaces in relational databases. All collections are treated as isolated resources, as are any sub-collections they might contain. Standard backup and recovery procedures are applied to collections by simply copying files that constitute any database (The Apache XML Project, 2007).

In addition to such basic management functions, Xindice fulfils further requirements of a DBMS, one of which concerns retrieving data by means of queries. In Xindice, queries are couched in a format called XQuery; that, in turn, uses a further XML tool, XPath, to identify XML elements. By itself, though, XPath does actually return any information: that is done by Xindice. XPath further underpins the XUpdate format that mediates database changes. Final support for Xindice as a DBMS comes from a facility for indexing data, which enhances retrieval in a similar way as in relational databases (The Apache XML Project, 2007). All of those DBMS-like features will be addressed shortly; for now, though, attention turns to the actual XML held in Xindice collections. After that, sections on querying and updating the database will make use of that XML repository.

[1] Xindice may be pronounced with an Italian lilt, 'Xeen-DEE-chay', if so desired (The Apache XML Project, 2007).

9.5 A Database of GRiST Mind Maps

An XML-centric approach, then, suggests using a database in that same format; the one chosen here, Xindice, will hold both mind maps and metadata. To that end, the first part of this section describes the creation the mind map portion of that database. Having created and loaded that partition, the process of retrieving mind map nodes is described, after which comes a way of retrieving entire paths of nodes from Xindice. Subsequently, a similar process is described for holding metadata in a separate part of that database. In addition to metadata that denotes related nodes, a further type depicts nodes that are better kept apart, due to antonymy in WordNet or to over-stemming conflating too large a range of words. The first of those tasks, then, is to create the required partitions in Xindice.

Creating A Xindice Collection for Mind Maps and Metadata

The Xindice command line comprises a shell script called `xindice.sh`. All commands from that interface follow a basic format, and use various parameters to express any required actions and resources. The first such parameter is the operation required; for example, the argument `ac` indicates that a new collection is to be created. Following that first argument, the `-c` argument always specifies the desired collection, in terms of the Hyper-Text Transfer Protocol (HTTP). Rather than an external web site, though, developers use a local service at HTTP port 8080; that port accesses the Tomcat Java Servlet container at the URL `localhost` provided by Tomcat (The Apache Software Foundation, 2010).

Command-Line Access to Xindice, by means of Tomcat

Tomcat, then, interprets and executes commands from Java classes that comprise Xindice, routing requests and delivering results over the Internet. In addition to any desired actions and resource locations, commands that manage collections further take the name of the desired collection, within the top-level collection given by the `-c` parameter. In that way, Figure 9.2 shows the creation of a new collection for holding GRiST mind maps; the `ac` command requires Xindice to 'add collection' within the database specified by the `-c` parameter, `xmldb:xindice://localhost:8080/db`. The name of that new collection, `mindmaps`, is given by the `-n` parameter:

```
> xindice.sh ac -c xmldb:xindice://localhost:8080/db -n mindmaps
trying to register database
Created : xmldb:xindice://localhost:8080/db/mindmaps
```

Figure 9.2: Creating a Xindice collection for GRiST mind maps

The collection specified by `-c` in Figure 9.2 was actually the top level of the database, `db`. After 'trying to

register database', Xindice announced success in creating the required collection[1]. In a similar way, the 'list collections' command `lc` provides information about the database. That command gave the top-level collections listed in Figure 9.3 for the main `db` collection:

```
> xindice.sh lc -c xmldb:xindice://localhost:8080/db
          mindmaps
          meta
          system
Total collections: 3
```

Figure 9.3: Main collections within Xindice.

The `lc` command, then, retrieves 'child' collections from within the one named in the `-c` argument. Of the three resulting collections, 'system' and 'meta' already existed, for Xindice's own use. The new 'mindmaps' collection, in contrast, was created to hold user-data in the form of GRiST mind maps. Subsequently, that new collection was populated by a variant of the 'add document' command, `ad`. In fact, mind maps from a single local folder were loaded by the `addmultiple` command. Figure 9.4 shows the `-f` argument used to specify that directory:

```
> xindice.sh addmultiple -c xmldb:xindice://localhost:8080/db/mindmaps
             -f /home/keith/mindmaps

Reading files from: /home/keith/mindmaps
Added document xmldb:xindice://localhost:8080/db/mindmaps/02-condensed.mm...
Added document xmldb:xindice://localhost:8080/db/mindmaps/73-condensed.mm
```

Figure 9.4: Loading GRiST mind maps into a Xindice collection.

Just the first and last XML files to be loaded appear in Figure 9.4, for brevity[1]. In that way, novel mind maps might be added from any source, given the required permission. Proof that mind maps from Figure 9.4 were loaded successfully came from the 'list documents' command `ld` shown in Figure 9.5; in the same was as in the preceding example, just Xindice's first and last responses are shown:

```
> xindice.sh ld -c xmldb:xindice://localhost:8080/db/mindmaps
    02-condensed.mm...
    73-condensed.mm
Total documents: 46
```

Figure 9.5: Mind maps loaded into Xindice as XML documents.

Figure 9.5 shows that all forty-six GRiST mind maps were loaded successfully. Although mind maps are pure XML, Xindice added a header to each mind map, in order to explicitly declare the XML content in the form `<?xml version=1.0?>`.

[1]The message 'trying to register database' is hard-coded into Xindice, and is omitted from subsequent examples.
[1]Note that GRiST mind maps were not numbered sequentially; there were, in fact, just 46 GRiST mind maps.

Accessing Xindice by means of Web Browsers

Rather than viewing mind map XML directly in Mozilla, the so-called 'dirty debug tool' that comes with Xindice was used; all the same, that tool uses Mozilla or any other web browser, such as Internet Explorer™. Figure 9.6 shows the contents of Xindice's 'mind maps' collection; individual mind map documents are listed on the left-hand side, while the right-hand side shows detail of the condensed mind map from GRiST expert 02:

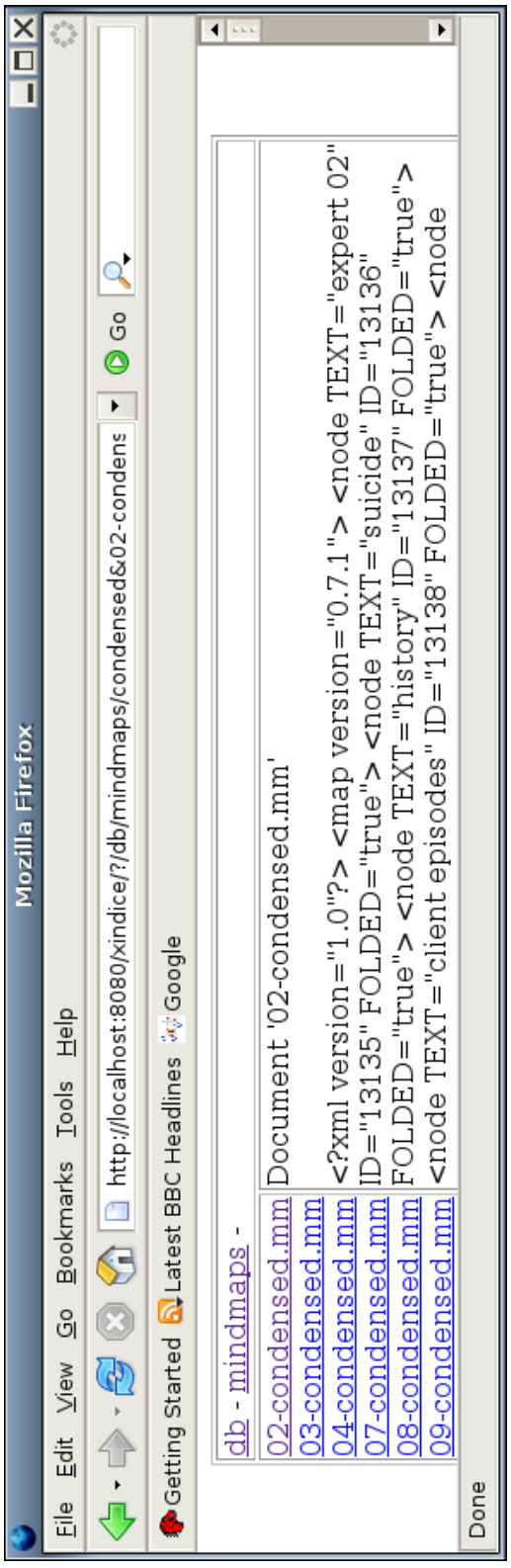

Figure 9.6: Detail from the 'mindmaps' collection of the Xindice database.

The browser's address bar from Figure 9.6 shows the full URL of the mind map in the main data window. The first line of that window shows the `mindmaps` collection as a sub-collection of the main one, `db`. Underneath that entry, to the left, the mind map displayed in the right-hand panel appears highlighted, and takes its name from the particular XML file, `02-condensed.mm`, that was originally loaded into Xindice. That file name is reflected in the corresponding XML document, in the right-hand panel.

The second line of the panel from Figure 9.6 shows the XML header added by Xindice, while remaining XML in that that right-hand panel is pure FreeMind. That XML shows node elements to have two main attributes: a unique numeric `ID`, and an associated `TEXT` attribute. Further attributes, such as `FOLDED`, control the FreeMind GUI. Although subsequent queries concerned with modelling knowledge ignored such attributes, they were retained so that mind maps might be loaded directly from Xindice; that will be shown shortly, in chapter 10.

Adding an Index to the Xindice Database

The unique `ID` attribute allowed XPath to identify specific nodes; in that sense, `ID` acted in the same way as do primary keys for relational databases. Indeed, such keys arise from the process of abstraction and normalisation described in Chapter 2; for GRiST mind maps, though, that was applied to existing mind maps, rather than acting as schema metadata. All the same, a further similarity arose between the two types of DBMS, namely, an index on such unique IDs.

Figure 9.5 shows Xindice's `add_indexer` command building an index called `index-1`, with the `-p` option specifying that it should be compiled from the `ID` attributes of corresponding `node` elements, expressed as `node@ID`:

```
> xindice.sh add_indexer -c xmldb:xindice://localhost:8080/db/mindmaps
             -n index-1 -p node@ID
CREATED : index-1
```

Figure 9.7: A unique index based on `ID` attributes.

As a result of the index built in Figure 9.5, XML elements are identified more quickly; more importantly, a sole representative for several identical nodes is easily identified within the `mindmaps` collection. That becomes clear in the next section, which deals with retrieving XML elements from Xindice.

Retrieving Mind Map Nodes from Xindice

Queries to Xindice are framed as XPath expressions that identify parts of an XML document. XPath expressions are composed of *location paths* that comprise one or more *location steps*; individual steps are separated by the character '/'. Each step has a *predicate* that filters the results from any preceding steps; predicates use functions such as `contains()` to identify any required XML elements. Single-step location paths suffice to find nodes having a specific text string, while more complex XPath expressions arise from combining location steps into a composite location path (Clark & Derose, 1999).

Figure 9.8 shows the result of running a simple query against the `mindmaps` collection, by means of the `xpath` command. Following the standard `-c` flag that specified the required collection, the XPath expression itself appears after the `-q` option; in this case, the expression `[@ID='13137')]` requested a specific node whose `ID` attribute was set to the unique key '13137'. Following that query comes a single result, as an instance of the element `xq:result`:

```
> xindice.sh xpath -c xmldb:xindice://localhost:8080/db/mindmaps
             -q "//node[@ID='13137')]/attribute::TEXT"
<xq:result TEXT="history"
 xq:col="/db/mindmaps"
 xq:key="02-condensed.mm">
```

Figure 9.8: Retrieving a specific node from the `mindmaps` collection.

The second line of the expression from Figure 9.8 uses `//node` to instruct XPath to process `node` elements from the root downwards, in search of the key '13137'. The particular node under consideration at any instant is called the context node, to which functions such as `contains()` are applied. The second step subsequently acts on any nodes identified by that first step, and uses, `attribute::TEXT` to request just

the `TEXT` attributes of nodes from the first step[1]. That expression further demonstrates the use of 'axes' within XPath; the `attribute` axis contains just attributes of nodes from preceding steps, while, say, the `child` axis contains just nodes contained within any nodes identified. (Clark & Derose, 1999).

Output from Queries that Research Xindice

Output from the query in Figure 9.8 comprised a `xq:result` element; the prefix `xq:` refers to a *namespace* that lists XML elements defined for XQuery[2]. Elements from that namespace dictate that any `xq:result` element must contain the results of a query, along with two other `xq:` attributes. Those extra attributes show the source of any result: `xq:col` names the originating collection, while `xq:key` gives the XML document within that collection. A similar format appears in Figure 9.9, where the XPath expression `[contains(@TEXT,'abus')]` specified nodes having `TEXT` attributes that contained the stem 'abus'. Because no particular node was requested by means of a unique key, seventy-three `xq:result` elements arose for nodes containing 'abus', of which just one is shown as an example:

```
> xindice.sh xpath -c xmldb:xindice://localhost:8080/db/mindmaps
             -q "//node[contains(@TEXT,'abus')]/attribute::TEXT"
<xq:result TEXT="abuse to client"
 xq:col="/db/mindmaps"
 xq:key="02-condensed.mm">
```

Figure 9.9: Retrieving nodes conflated by 'abus' from the `mindmaps` collection.

As was the case for queries specifying particular node keys, the query from Figure 9.9 returned just single nodes from Xindice. A second facility of XPath, though, was useful in retrieving the full paths of nodes from GRiST mind maps, as is discussed next.

Retrieving Node Paths from Xindice

Chapter 8 combined the full paths of mind map nodes into more representative structures; such paths were, in fact, retrieved by extending the basic XPath expression from Figure 9.8. Rather than the full Xindice query, just the XPath component appears next, first in full, and subsequently as successive steps:

```
xpath //node[@ID="10238"]/ancestor-or-self::node/attribute::ID
```

Step 1:	`//node[@ID="10238"]`	Identify the node having an ID attribute set to '10238'.
2:	`/ancestor-or-self::node`	Node 10238 itself, plus ancestors up to the root node.
3:	`/attribute::ID`	Select just `ID` attributes of nodes from step 2.

[1] Without that `attribute::TEXT` axis, Xindice returns the full sub-hierarchy above any matching node.
[2] Although not displayed, the namespace in Figure 9.8 was `xmlns:xq="http://xml.apache.org/xindice/Query"`.

Step 1 used a unique ID to identify node key 10238; having selected that context node, step 2 ascended the node hierarchy back towards the root, by means of the `ancestor-or-self` axis. That axis worked by first identifying the parent of node 10238 from step 1, followed by the parent of that immediate parent, and so on. The final element was the root node of the mind map that contained node 10238. Finally, step 3 used the `attribute` axis to strip away all but the `ID` attributes of nodes from such paths. Table 9.1 shows nodes from the resulting path, with the required node 10238 as the final entry:

Level	ID	Text
Root	10047	condensed 15
0	10209	self neglect
1	10231	assessment
2	10232	client presentation
3	10237	bruising
4	10238	suggests someone abusing them

Table 9.1: The full path to node 10238, from the XPath expression in Figure 9.9.

Retrieving just `ID` attributes minimised the amount of information processed by the intense recursive processing required to identify ancestor nodes. Node texts themselves were subsequently fetched separately for each key in any path; that was done by efficient queries that specified such unique keys. In fact, texts were reattached to respective nodes by means of Java classes, although it might well be done in XPath itself.

In fact, searching just for keys mimics an aspect of relational databases, in that the database proper need not be searched. Rather, the single field required was held in the index defined earlier; such 'index-only' retrieval is very efficient, as response times demonstrate. Take, for example, nodes containing the stem 'abus' that were retrieved by the XPath expression from Figure 9.9. Without a unique index on the `ID` attribute, retrieving paths for those nodes took 64,028 milliseconds, or just over a minute. Creating the index brought that down to 1,856 milliseconds: under two seconds. While such improved response times are not the primary aim here, they permit forms of analysis that make the database do most of the work.

9.6 A Database of GRiST Metadata

Having identified metadata, a repository is needed to store it. In fact, metadata can be encoded as XML (Thuraisingham, 2002; Jeffery, 2000), suggesting the Xindice database as just such a repository, although using a separate partition to the GRiST mind maps. Schema metadata used here dictate fields within tuples, for example as $nodeID \rightarrow concept$, where *concept* holds values such as 'abuse'. Field names in that tuple are schema metadata, while actual values are associative metadata; in that way, metadata relate specific values together. Those first two types of metadata in turn act as navigational metadata,

by which Chapter 10 extends the FreeMind GUI. Having identified metadata, then, a medium is needed in which to store it; the Xindice database offers just such a repository, as shown next.

Creating Xindice Collections for Metadata

Alongside the 'mindmaps' collection at the top level of the database, a separate main collection was named 'metadata'. Three types of metadata were stored in that collection. The first type was for stems that identified related word forms; in fact, stems were called 'keys' within the database, reflecting their role as an access method. The second type of metadata comprised words that should not be conflated by a given stem, to avoid grouping too wide a range of words; the third type further stored spelling corrections that were applied dynamically to nodes, on reading them from the database. To accommodate those three types of metadata, sub-collections were created within the main 'metadata' collection. Figure 9.10 shows the resulting child collections by means of the 'list collections' command, `lc`:

```
> xindice.sh lc -c xmldb:xindice://localhost:8080/db/metadata
   keys
   exclusive-keys
   corrections
Total collections: 3
```

Figure 9.10: Sub-collections of the main 'metadata' collection

Those three sub-collections each contained a single document for storing metadata elements. Figure 9.11 shows one such document under the 'keys' metadata collection; that information came from the 'list documents' command, `ld`, for the specified collection:

```
> xindice.sh ld -c xmldb:xindice://localhost:8080/db/metadata/keys
   keys-root
Total documents: 1
```

Figure 9.11: The 'keys-root' XML document in the 'metadata/keys' collection.

Figure 9.11 shows that the document `keys-root` was created in the `keys` collection, which was contained in turn within the top-level `metadata` collection. That new document, though, was empty; accordingly, a sole XML element called `keys` was added. That element constituted a root node for any subsequent queries or updates, which require a context node on which to operate.

The result, then, was a sub-collection that stored stems, which comprised secondary keys in addition to the main access route of `nodeID`. That new XML document contained a sole element, further called `keys`. Actual metadata were stored as child nodes of that document-level element, as Figure 9.12 shows for metadata about stem keys:

```
−<keys>
    <key TEXT="abilit"/>
    <key TEXT="young"/>
  </keys>
```

Figure 9.12: Example of XML for holding keys metadata.

Individual `key` elements from Figure 9.12 were, then, nested within the overall `keys` element at the root of a document. In that way, the collection for a specific type of metadata was primed with a single-element document; actual stems comprised `key` elements nested within the document's root element. In a similar way, the remaining two metadata collections were primed with documents called `corrections-root` and `exclusive-keys-root`, with respective XML elements called `corrections` and `exclusive-keys`.

A similar situation existed for spelling metadata, but with two attributes instead of just one. Figure 9.13 shows that the first attribute, `WORD1`, reflected a spelling error, while `WORD2` supplied a correction:

```
−<corrections>
    <correction WORD1="abiity" WORD2="ability"/>
    <correction WORD1="worring" WORD2="worrying"/>
  </corrections>
```

Figure 9.13: Example of XML for holding spelling correction metadata.

Whereas spelling metadata from Figure 9.13 required just pairs of `WORD` attributes, the third type of metadata had to cater for any number of words that should be processed separately; that is done by means of `STEM` elements. Figure 9.14 shows such metadata for two examples. The first is for the words 'male' and 'female' that were conflated by 'male'. WordNet reported those words to be antonyms, and should be treated separately; all the same, they are marked as related by the `TYPE` attribute. The second example involves the key 'form' that identified a variety of words that should be handled in isolation:

```
−<exclusive-stems>
    <exclusive TYPE="related" STEM1="male" STEM2="female"/>
    <exclusive TYPE="exclusive" STEM1="form" STEM2="formal"
                                STEM3="inform" STEM4="informal"/>
  </exclusive-stems>
```

Figure 9.14: Example of XML for holding exclusive-keys metadata.

Metadata from Figure 9.14 allows software to process words separately, or together; regardless, the distinction can be made clear to users. Having determined what metadata must be stored, then, the next section describes the exact mechanism for creating metadata in Xindice.

Creating Metadata in Xindice

XML elements for metadata contain one, two, or several attributes, depending on the type; all types, thought, are treated in a similar fashion. In fact, metadata for spelling corrections is used to demonstrate how new elements were added. That type took just two attributes: one for any misspelling, and another for the corresponding correction. For any such pair, XUpdate appended a new `correction` element to the root element, `corrections`. The syntax of XUpdate was similar to that of XQuery, and used XPath to identify an appropriate root element. Figure 9.15 gives the XML for adding a single new correction:

```
−<xu:modifications version="1.0">
 −<xu:append select="/corrections">
  −<xu:element name="correction">
     <xu:attribute name="WORD1">aclcohol</xu:attribute>
     <xu:attribute name="WORD2">alcohol</xu:attribute>
   </xu:element>
  </xu:append>
 </xu:modifications>
```

Figure 9.15: XUpdate XML for appending a new correction element

The short name `xu:` is from the XUpdate namespace, which defines the required elements. The `xu:append` element on the second line indicates where to add the new child element; that is under the `corrections` root element, as specified by the `select` attribute. Line 3 shows that a new `correction` element is to be added. Lines 4 and 5 give the required `WORD` attributes, before remaining entries close all higher level elements. The resulting XML is sent to Xindice for processing.

Retrieving Metadata

In fact, metadata about stem keys is the most easily retrieved: the XPath expression for that was simply `"/keys"`, to list all elements in the `keys-root` document. The resulting list of `key` elements provided the required stems as `TEXT` attributes. Further types of metadata, though, required more processing. In particular, exclusive stems might have multiple `STEM` attributes; for any given stem, then, bespoke Java classes interrogated the database for exclusions. Such cases arose from grouping too broad a variety of words, for example, the words ‘inform’ and ‘formal’ that were conflated by ‘form’. Metadata, though, allowed XPath to keep those words separate.

To that end, XPath had to identify `exclusive-key` elements corresponding to any given word. Figure 9.16 shows how the Java class `MindmapMetadata` used those entries. The filter `@*='inform'` in the first line matches any XML attribute that contains the word 'inform', on whatever element. The next entry shows the resulting metadata, which had three `STEM` attributes, of which `STEM3` was set to 'inform'. The line after that shows the automated decision to treat those stems separately, due to the spread of L′ for corresponding words. Following that is an XPath expression that identifies words closely related to 'inform', by excluding words conflated by 'formal'. The final entry shows the six words actually conflated by the stem 'inform':

```
xPathExpression = //exclusive[@*='inform']

<exclusive STEM1="form" STEM2="formal" STEM3="inform"/>
keep INFORM separate from: FORM FORMAL

required xPathExpression =
//node[contains(@TEXT,'inform')][not(contains(@TEXT,'formal'))]

6 filtered word(s) for "inform":  inform, informs, informed,
informing, informants, information
```

Figure 9.16: XPath expression for selecting exclusive-key metadata

The third entry in the XPath expression from Figure 9.16 used the `contains()` function to identify the desired nodes bearing the stem 'inform', while the `not()` function removed unrelated words stemmed by 'formal' from any results. Combining two conditions in that way identified nodes that contained 'inform', but not the word 'formal'; because 'informal' contains both, it was ignored. No check was needed for the word 'form', as it was shorter than either 'inform' or 'formal', and would conflated a wider group of words. All the same, concepts might be shown as related by setting the corresponding `TYPE` attribute.

Because `TEXT` attributes are treated as entire strings, though, a problem would arise for any node containing both 'inform' *and* 'informal'; without further analysis by Java code, version 1.1 of XPath used in Xindice would fail. In fact, version 2 of XPath allows for smaller sub-strings, called tokens (Clark & Derose, 1999). Tokenising text would permit XPath to check individual words within nodes; for that reason, the adoption of version 2 by Xindice would be welcomed.

9.7 Implementation Considerations

FreeMind stores mind maps as XML, which can be parsed by software to produce data structures for analysis; in developing GRiST, that involved software written ion Lisp. That programming language was well suited to representing hierarchical structures, and to recursive analyses that extracted information from such trees. In fact, those Lisp programmes converted FreeMind XML into a more general format, for display in web browsers and accompanying client-side software created in Macromedia Flash. In that way, participants in GRiST were able to review their own interview data, and further to gain an overall view of collected experts' ideas (Buckingham & Adams, 2006).

Changing and Augmenting FreeMind

In contrast, the approach taken here was to process FreeMind XML directly, and to present any results by means of that same GUI. In fact, FreeMind was written in a freely available programming language called Java (Sun Microsystems Inc., n.d.), which is based on the technique of Object Orientated Programming (OOP); that approach stresses the reuse of existing programmes, known as *classes*. Reusing existing code requires the key word *extends*, as Figure 9.17 shows for a class called `AstonFreeMind`:

```
public class AstonFreeMind extends FreeMind {
}
```

Figure 9.17: Extending the `FreeMind` Java class, to create `AstonFreeMind`.

The resulting `AstonFreeMind` class from Figure 9.17 provided all the functionality of the parent class, `FreeMind`; additional code was needed just for altering that functionality, and for adding facilities. The first such amendment was to remove the first two navigation buttons from the main tool bar; those two buttons respectively comprised the 'next' and 'previous' controls. To that end, Figure 9.18 shows Java code that retrieved an array of panel sub-components, from a `FreeMind` instance referenced by the reserved word 'this':

```
Component[] components = this.getContentPane().getComponents();
MainToolBar mainToolBar = (MainToolBar)components[0];

mainToolBar.remove(0);
mainToolBar.remove(0);
```

Figure 9.18: Removing unwanted `FreeMind` options from the novel `AstonFreeMind` class.

In fact, that main tool bar retrieved in Figure 9.18 was the first sub-component within any FreeMind instance, at index '0', from which the first two entries were removed. The 'cast' denoted by parentheses forced retrieved components to take the form of `MainToolBar` objects. Subsequently, the first two entries were removed by their successive positions at the front of the list supplied by FreeMind. Although that Java code has repeated references to array index 0, the first occurrence removes the entry at the beginning of that list, promoting what was previously the second element to the front, for the second invocation of `remove()`.

Restricting FreeMind's File Operations

The second change made to FreeMind's standard behaviour was to read mind maps from the `mindmaps` collection within Xindice; further, any changes or novel mind maps were written to separate XML files, in order to ensure the database's integrity. To that end, FreeMind's options for opening and saving mind maps as XML documents were deactivated; first of all, though, the existing 'Save' menu was retrieved from the FreeMind menu bar by means of the following bespoke Java method, `getMenu()` presented next

That method takes as arguments the name of the desired menu, and the actual FreeMind menu bar object on which it lies. Successive menus are retrieved by means of Java's native `menuBar.getItem()` method, for as many as are reported by the variable `menuBar.getMenuCount()`. Each menu's name is checked, and the position of the required menu stored in the variable `menuIndex`, which terminates the search:

```
private int getMenu(String menuName, MenuBar menuBar) {
      int menuIndex = -1;
      for (int i = 0; menuIndex < 0 && i < menuBar.getMenuCount();++i; {
            JMenuItem menuItem = menuBar.getItem(i);
            if (jMenu.getText().equals(menuName)) {
                  menuIndex = i;
            }
      }
      return menuIndex;
}
```

Figure 9.19: The bespoke Java method `getMenu()` from the `AstonFreeMind` class.

Passing the name 'Save' to the `getMenu()` method from 9.19 retrieved the position of that menu item on the main FreeMind menu bar, which appears at the top of the GUI. That, in fact, was a drawback of Java: lists of components on GUIs have to be searched for any desired object. Although more direct access might be allowed by novel code, Java lacks any direct mappings between components and their position on the GUI.

Having provided a means of retrieving menu objects, then, code in Figure 9.20 retrieves of the FreeMind menu bar, from which it subsequently removes of the 'Save' menu item at the corresponding index, should it be an acceptable value. Subsequently, a bespoke 'Save As' option is added back to the menu bar in the form of Java's built-in `JMenuItem` class. To that new item is added a `MouseListener` object that detects mouse movements and clicks, which are handled by means of an 'new' instance of the bespoke Java class `SaveAsProcessor()` to produce the required changes in FreeMind's file handling:

```
MenuBar menuBar = this.getFreeMindMenuBar();
int menuIndex = getMenu("Save", menuBar);

if menuIndex > -1) {
      mainMenu.remove(menuIndex);
      JMenuItem saveAs = new JMenuItem();
      saveAs.addMouseListener(new SaveAsProcessor());
      mainMenu.add(saveAs);
}
```

Figure 9.20: Replacing the 'Save As' dialogue in the `AstonFreeMind` class.

By means of the `SaveAsProcessor()` class from Figure 9.20, then, a new 'save as' option committed updated GRiST mind maps as novel XML documents, outside Xindice. Further, generated combined and normalised maps might be stored for later retrieval. That, in turn, meant replacing various 'listeners' that invoked custom code according to whatever key was pressed, or mouse button clicked. Particular actions, then, were specified by attaching independent listeners to each button of interest. In a similar way as for the existing 'Save' menu, FreeMind's 'New' and 'Open' options were removed from the main menu; those options, though, were not reintroduced in any form.

Additional Options for FreeMind's Node Menus

In addition to the main FreeMind menu bar, right-clicking on individual nodes yields pop-up menus of available options; such node-level menus were further modified to deactivate options that might not apply, say, due a lack of relevant metadata for any given node.

The Java code from Figure 9.22 started by calling the `getMenu()` method. That method, though, was not the one from Figure 9.19; rather, a further method of that name took just the required menu as an argument, and processed FreeMind's 'node' menu without needing that object as a parameter. Such 'overriding' allows Java to recognise methods that, while having the same name, have differing 'signatures' of arguments. The first two statements, then, derive the dimensions of the new pop-up in terms of the current FreeMind GUI; the pop-up is half the width and height of the containing panel. The third statement creates the actual pop-up as a new instance of FreeMind, while the succeeding three lines

```
int AstonIndex = getMenu("Aston Options");

for (int i = 0; AstonIndex > -1 && i < 4
&& i < jToolBar.getComponentCount(); ++i; {
      Component component = jToolBar.getComponent(i);
      component.setEnabled(false);
}
```

Figure 9.21: Enabling the menu of 'Aston Options' in the `AstonFreeMind` class.

respectively hide the standard toolbar, sets the pop-up dimensions, and further sets the 'zoom' level to adjust the size of the resulting mind map. Lastly, the new pop-up is added to the main FreeMind GUI:

```
Double width = this.getSize().getWidth() / 2;
Double height = this.getSize().getHeight() / 2;

FreeMind popUp = new FreeMind();

popUp.getController().setToolbarVisible(false);
popUp.setPreferredSize(width, height);
popUp.getView().setZoom((float) 0.75);

this.getContentPane().getComponents().add(popUp);
```

Figure 9.22: Creating a pop-up window within the FreeMind GUI of the `AstonFreeMind` class.

The code from Figure 9.22 shows the OOP nature of FreeMind's style of Java; by means of the variable `this`, an entire FreeMind GUI was made available for manipulation. In that way, various new menus were added to FreeMind, while certain standard options were removed or redirected. That ability to retrieve entire FreeMind objects further helped to identify the keys of whatever nodes were selected. The example given in Figure 9.23 shows how individual nodes were obtained from FreeMind; that process starts by retrieving the `MapView` object, provided by the standard `FreeMind` class, that holds any mind map node hierarchy. Subsequently, an associated `NodeView` object returns whatever nodes were selected, with the actual nodes appearing from the `getModel()` method:

```
MapView mapView = this.getView();
NodeView nodeView = mapView.getSelected();

MindMapNode node = nodeView.getModel();
```

Figure 9.23: Creating a pop-up window within the FreeMind GUI of the `AstonFreeMind` class.

Java code from Figures 9.18 to 9.23, then, showed ways in which control was taken from FreeMind itself, and passed to novel code in the `AstomFreeMind` class. In order to put those separate components together, attention turns next to an overview of a proposed system having, at its heart, the forty-six GRiST mind maps held in Xindice, and the metadata arising from this thesis.

An Overview of the Proposed System.

The proposed system, then, relies on GRiST mind maps and any corresponding metadata held in Xindice. Along with Java's facility with XML, that promoted the XML-centric development encouraged by Wilde (2006). In addition to actual Xindice data, XML provided a mechanism for querying the database, and for transporting results back to users. As a result, XML elements both from mind maps and from metadata collections were identified by XPath, and returned by way of HTTP, making Xindice available both from a command line and from Java classes. In fact, that was done by means of a new class called `MindMapMetadata`, which relied on a further new class, `XindiceCollection`, to connect to the required Xindice collection. For easier command line usage, both of those classes were invoked by a third, called `XRun`, that required two arguments: the last part of a collection name, and a XPath expression.

Part One: Extracting Knowledge from GRiST Mind Maps.

Such communication with Xindice allowed GRiST mind maps to be readily accessed, and further stored additional knowledge held as metadata. The first of those types arose from spelling correction; that process appears in the top part of Figure 9.24, where FreeMind XML from GRiST mind maps checked for spelling errors. Suggested replacements from Jazzy were refined by means of L', and any accepted corrections stored in Xindice. In a similar way, stems identified in the bottom part of Figure 9.24 are written to the database; Xindice itself appears to the right-hand side, adjacent to the Tomcat servlet container that mediates communication between Java objects and Xindice:

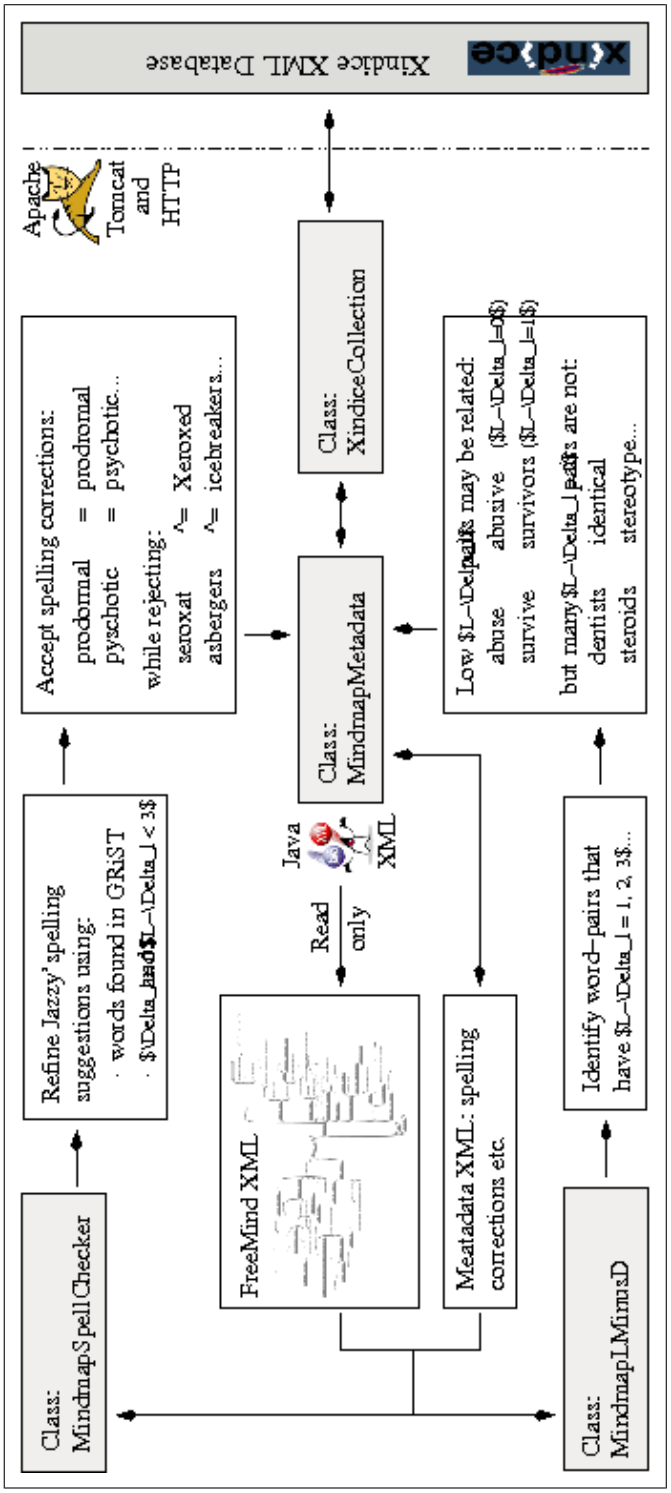

Figure 9.24: Implementation Diagram I

The two novel classes from Figure 9.24, then, mediated communication with Xindice; those classes, `MindmapMetadata` and `XindiceCollection`, might in fact store any XML data in a web-orientated format. That suggests a more general use of Xindice, as an alternative to the SQL databases that currently support dynamic web sites. Developing such systems is assisted by the Tomcat container, which eases development by mimicking web site behaviour. In fact, the `MindmapMetadata` class handled all transfers of data. On receiving a request for data from Xindice, a new `XindiceCollection` object established a connection; on completion, `MindmapMetadata` returned lists of XML elements from `XindiceCollection` and closed the corresponding connection.

Part Two: Applying Knowledge to GRiST Mind Maps.

Having extracted spelling corrections and stems from GRiST mind maps, the second phase involved WordNet, and three new Java classes. The first of those was `MindmapPOSAnalysis` that performed CA on prepositions and meta-type pairs, while the second, `MindmapPathsAnalysis`, provided metadata for generating novel mind maps. In turn, those classes invoked a class called `AutoCA` to perform CA and to interpret any results. Accordingly, Figure 9.24 shows just those three new classes, and omits the `MindmapMetadata` and `XindiceCollection` classes introduced previously. Further, the Xindice database appears at the top of Figure 9.24; Java classes on the left-hand side used CA to disambiguate WordNet entries, while classes to the right used WordNet to conflate nodes by meaning, in addition to any morphological similarities:

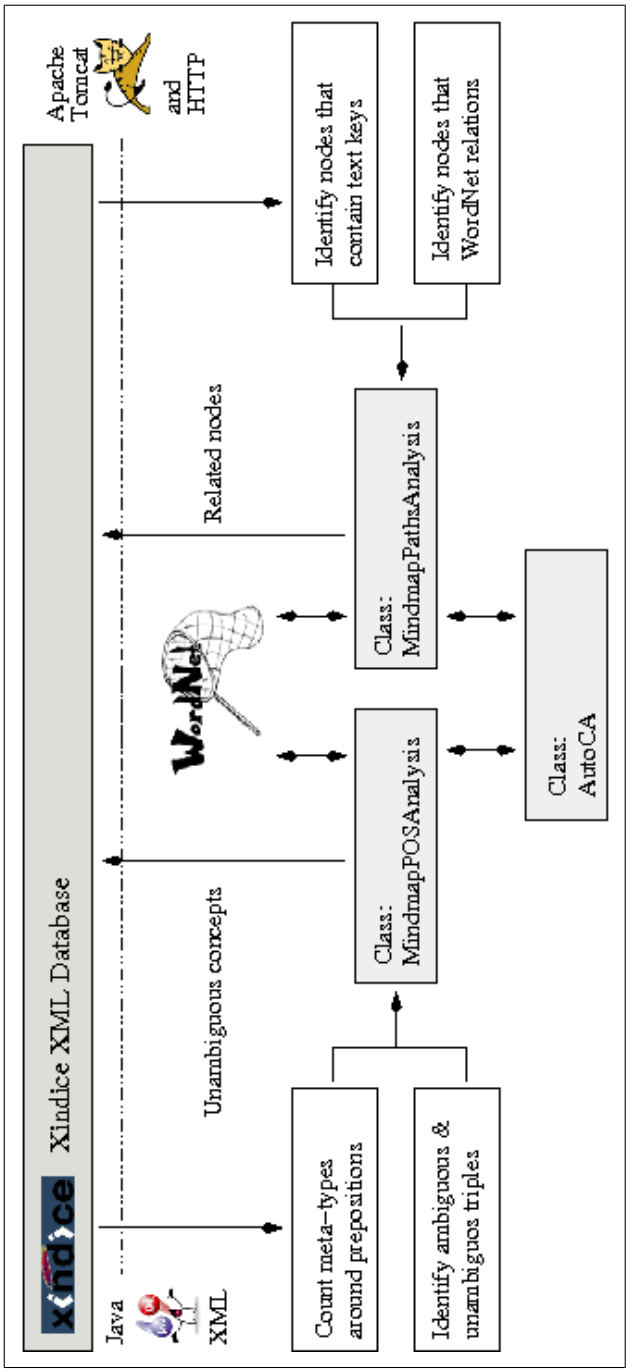

Figure 9.25: Implementation Diagram II

The combination of Figures 9.24 and 9.25, then, yielded a system for interacting with knowledge from GRiST mind maps. By means of the Xindice XML database, and Java's facility for handling XML, the resulting system followed the XML-centric approach espoused by Wilde (2006). The resulting information base of expert mental health knowledge stored unambiguous concepts and related nodes, which formed the basis for the final phase; that comes after a summary of this chapter.

9.8 Chapter Summary

This chapter introduced XML as a flexible way of holding data. Indeed, FreeMind uses XML to represent mind maps. In contrast to, say, relational databases, XML represents just semi-structured data; that makes metadata all the more important, so as to impose more structure on such data. Indeed, the system described here adopted the approach of XML-centric development', which reflected the use of end-to-end XML in this thesis.

Following that introduction came an overview of the Xindice XML database, which provided a convenient repository for both mind maps and metadata. Xindice was shown to posses important features of any DBMS, including support for separate collections, query management, and update processing; in addition, XPath was introduced as vital to those activities. XML from FreeMind node hierarchies was loaded directly into a Xindice collection, with text held as attributes of generic `node` elements. The Xindice command line was used to create and populate the database; subsequently, queries of varying complexity showed how mind map nodes were retrieved by XQuery and XPath. Notably, a unique index greatly speeded up some of those queries.

Following that overview of Xindice, metadata were described as vital to future information systems. Having identified metadata, Xindice offered an ideal repository for any resulting XML, in that separate collections kept metadata apart from GRiST mind maps. Within the corresponding metadata collection were created three sub-collections, one for each type of metadata; that was done by means of XUpdate, which added new metadata elements to the database. One particular type of metadata stored exclusive stems, which were best treated separately; that allowed a flexible approach to handling words conflated by such stems. To that end, adoption of version 2 of XPath for Xindice was strongly encouraged.

Lastly, various Java classes were shown in relation to the Xindice database for mind maps and metadata. Two particular classes mediated all communication with Xindice. By means of those helper classes, mind map nodes and associated metadata were retrieved precisely, and swiftly. Indeed, such databases are seen as useful in wider applications for the Internet, which might well become the next important generation of databases. Certainly, XML technologies performed well in the approach reported here; the culmination of that work forms the penultimate chapter of this thesis.

10

Benefits to GRiST

10.1 Introduction

In Chapter 9, the `FreeMind` Java class was extended to create a novel class called `AstonFreeMind`, which was augmented by means of new facilities. The first such amendment was to remove certain navigation buttons from the main tool bar; those buttons offered options that processed mind maps outside Xindice. The second change to FreeMind involved reading mind maps from Xindice's `mindmaps` collection, with any changes or novel mind maps written to separate XML files. To that end, FreeMind's options for opening and saving mind maps as XML documents were deactivated, and a bespoke 'Save' used instead.

In addition to the main FreeMind menu bar, pop-up menus from the 'nodes' menu further gave access to metadata gathered during this thesis. The FreeMind interface, then, was tailored to maintain the integrity of the Xindice database, and to display 'Aston Options' that present mind maps created from metadata stored in Xindice. Indeed, raw metadata should be presented in a way that is amenable to humans (Berners-Lee, 1997); such a facility is provided by the Graphic User Interface (GUI) described next. That GUI comprises parts, the first being a work space for GRiST researchers and panellists, which allows easy access to the mind maps that underpin the project. Following that come novel menu options that operate on entire mind maps, and further options available for individual nodes.

10.2 A Work Space for GRiST

The panel shown in Figure 10.1 is the initial display of the new `AstonFreeMind` class, which retrieves the names of GRiST mind maps from Xindice, and generates the required XML to a temporary file. That file is subsequently loaded by the existing 'Open' functionality in FreeMind. The resulting GUI, then, provides a convenient interface through which to handle the forty six GRiST mind maps:

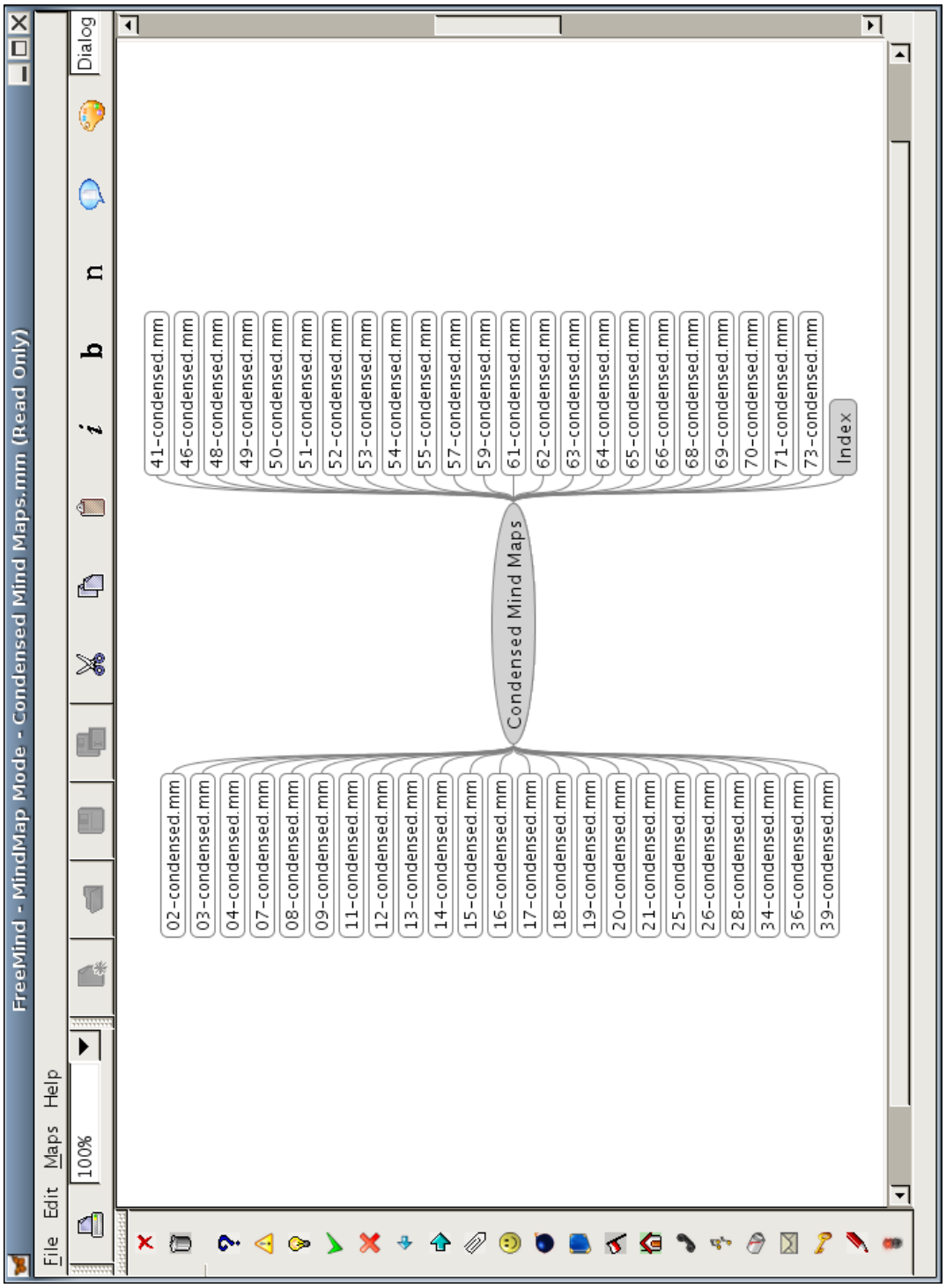

Figure 10.1: Initial Screen from the AstonFreeMind GUI

Mind maps represented by nodes in Figure 10.1 can now be opened either individually, in groups, or all forty-six in one go. Further, selected mind maps may be merged into a single entity by the process described in Chapter 8.

In addition to nodes carrying the names of mind maps in Figure 10.1, an `[Index]` node was added at the bottom right-hand corner, as can be seen in Figure 10.1. Branches from that node allow users to display metadata gleaned during this thesis. Some of the options provided by this new panel are handled by the parent `FreeMind` class, while others reflected bespoke code from `AstonFreeMind`. Available options fall into two categories: those that act on whole mind maps, and others that work at the level of individual nodes. Both types will now be presented in more detail.

10.3 Menu Options that Operate on Mind Maps

One of the new facilities provided by `AstonFreeMind` appears on the panel in Figure 10.2, and allows users to open several GRiST mind maps simultaneously; a further option generates a single entity from paths in the selected mind maps. Those actions are available from a new 'Aston Options' entry on the pop-up menu arising from right-clicking on selected nodes:

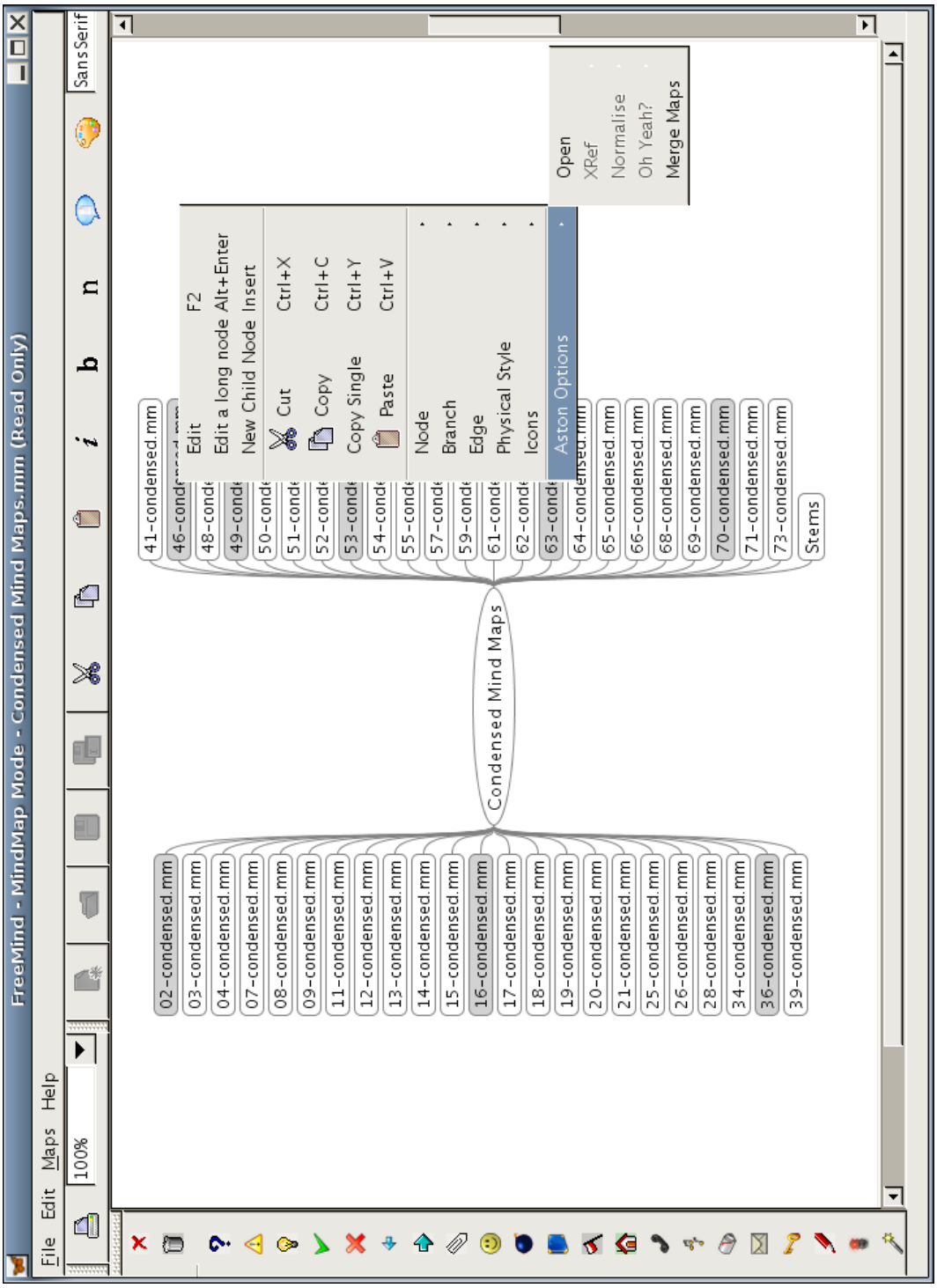

Figure 10.2: Map Options Menu on the AstonFreeMind GUI

Note that just two options are available in Figure 10.2; the remainder are 'greyed out' by means of retrieving that menu from the `FreeMind` class, and disabling options that do not operate on entire mind maps, as described in Chapter 9. Clicking on the 'Open' in the new Aston menu, then, opens several mind maps in separate panels within the main FreeMind GUI. In contrast, the 'Merge Maps' option combines any selected mind maps into a single, un-normalised version, in a sole new panel. Remaining Aston options are disabled because they operate on particular nodes or on atomic concepts; those options will be presented in full after having presented mind maps arising from the options that were available.

Multiple panels resulting from the 'Open' option need no further explanation; that option opened mind maps exactly as would FreeMind, except that multiple clicks on the standard 'Open' dialogue would be avoided. Mind maps arising from the 'Merge' option, though, will not have been seen before; accordingly, the result of merging the mind maps selected in Figure 10.2 are presented as a single, un-normalised mind map in Figure 10.3. That combined version presents the summed knowledge about mental health risks in the selected mind maps, whose GRiST identifiers appear in the root node:

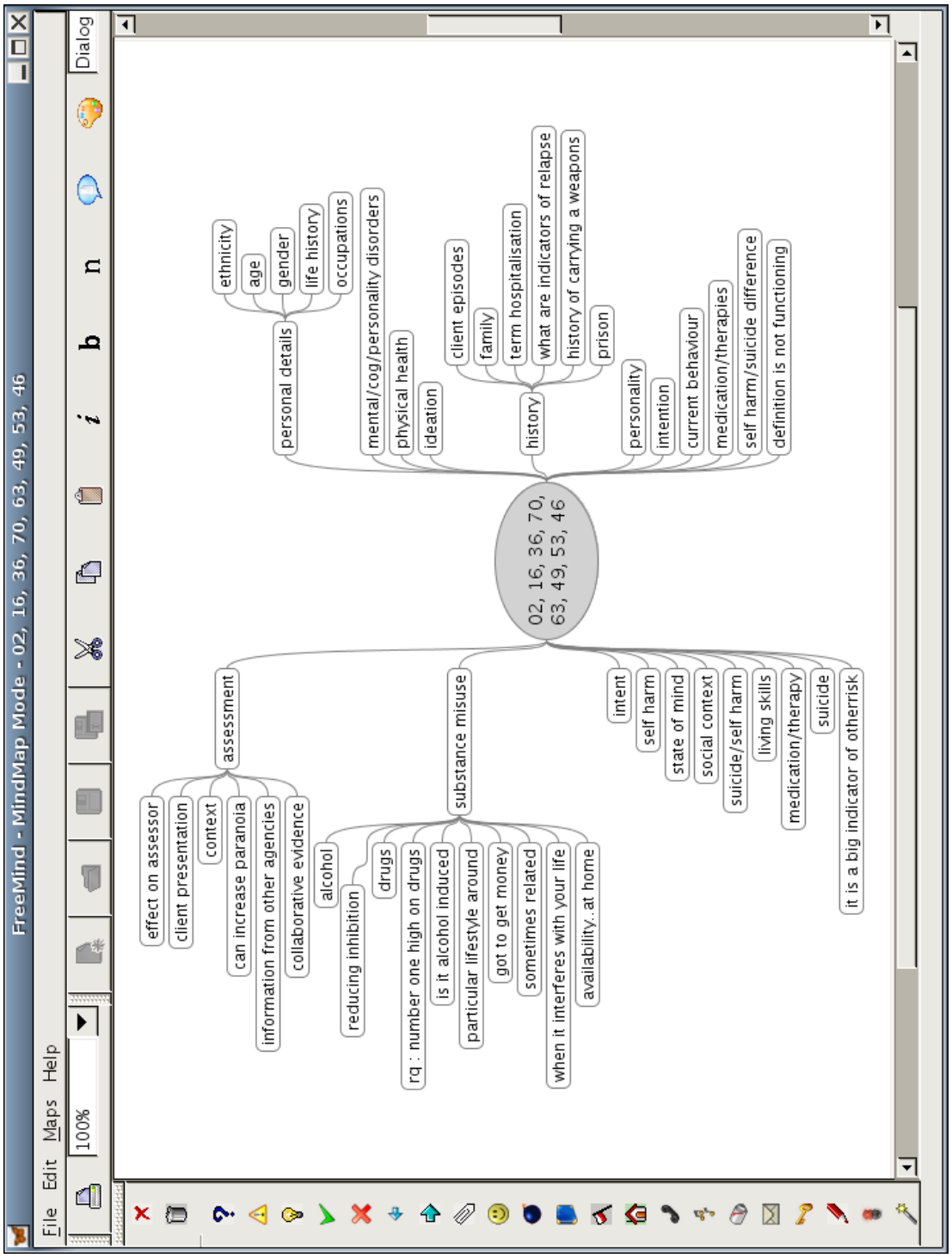

Figure 10.3: Combined mind map from the 'merge' option in Figure 10.2

In fact, the mind map from Figure 10.3 reflects paths just from the upper levels of the combined mind map hierarchy; in fact, did not go much further, as just the first three levels were merged. Chapter 8, though, proposed using structures from overlapping levels as a means of joining separate analyses into complete, combined mind maps. Although not normalised, such merged mind maps distil knowledge from selected experts into a graded summary of risk factors. Having, then, shown novel options that act on entire mind maps, attention turns next to facilities that operate on individual nodes.

10.4 Menu Options that Operate on Mind Map Nodes

Returning to the initial screen from the AstonFreeMind GUI, a further node was added that did not, in fact, refer to any GRiST mind map. That was the `[Index]` node from Figure 10.2, which was shown 'folded', that is, with child nodes hidden from view; that node is shown unfolded in Figure 10.4 to reveal a node hierarchy. The stems of various concepts identified in experiments by means of the Levenshtein Distance, reported in Chapter 4, are presented in alphabetical groups; in this case, shaded entries 'J' and 'N' have further been unfolded to reveal associated stems:

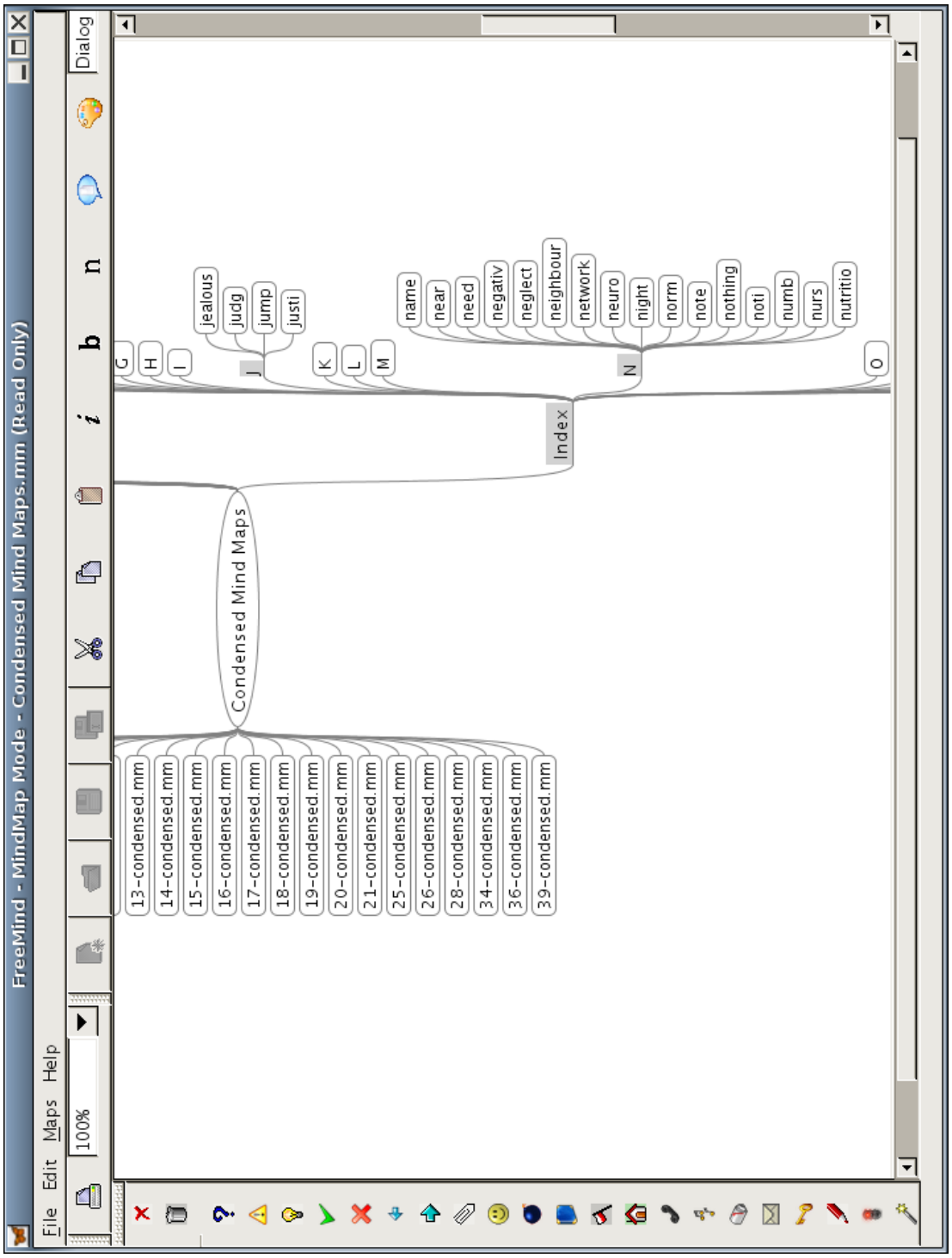

Figure 10.4: Detail from the 'Index' node in Figure 10.2

The index node for `[N]` from Figure 10.4, say, lists stems such as 'judg' for 'judging', 'judged' and 'judgemental'. A right mouse click while positioned over any such index entry identifies paths to nodes containing the corresponding stem. Those paths can then be combined into two ways; the first is a cross-reference that displays paths exactly as found in GRiST mind maps, but combined into a single hierarchy. The second option, in contrast, generates a normalised node hierarchy of more atomic concepts, and further reorganises correspondences to top-level risk categories. In order to illustrate those options, Figure 10.5 shows the pop-up menu arising from right-clicking the index node for 'depress'. On that menu, options that operate on entire mind maps are unavailable for the `[Index]` node hierarchy. Instead, the `[XRef]` and `[Normalise]` nodes offer facilities for generating mind maps from paths just to nodes containing the required stem, to form novel, distinct overviews of particular concepts:

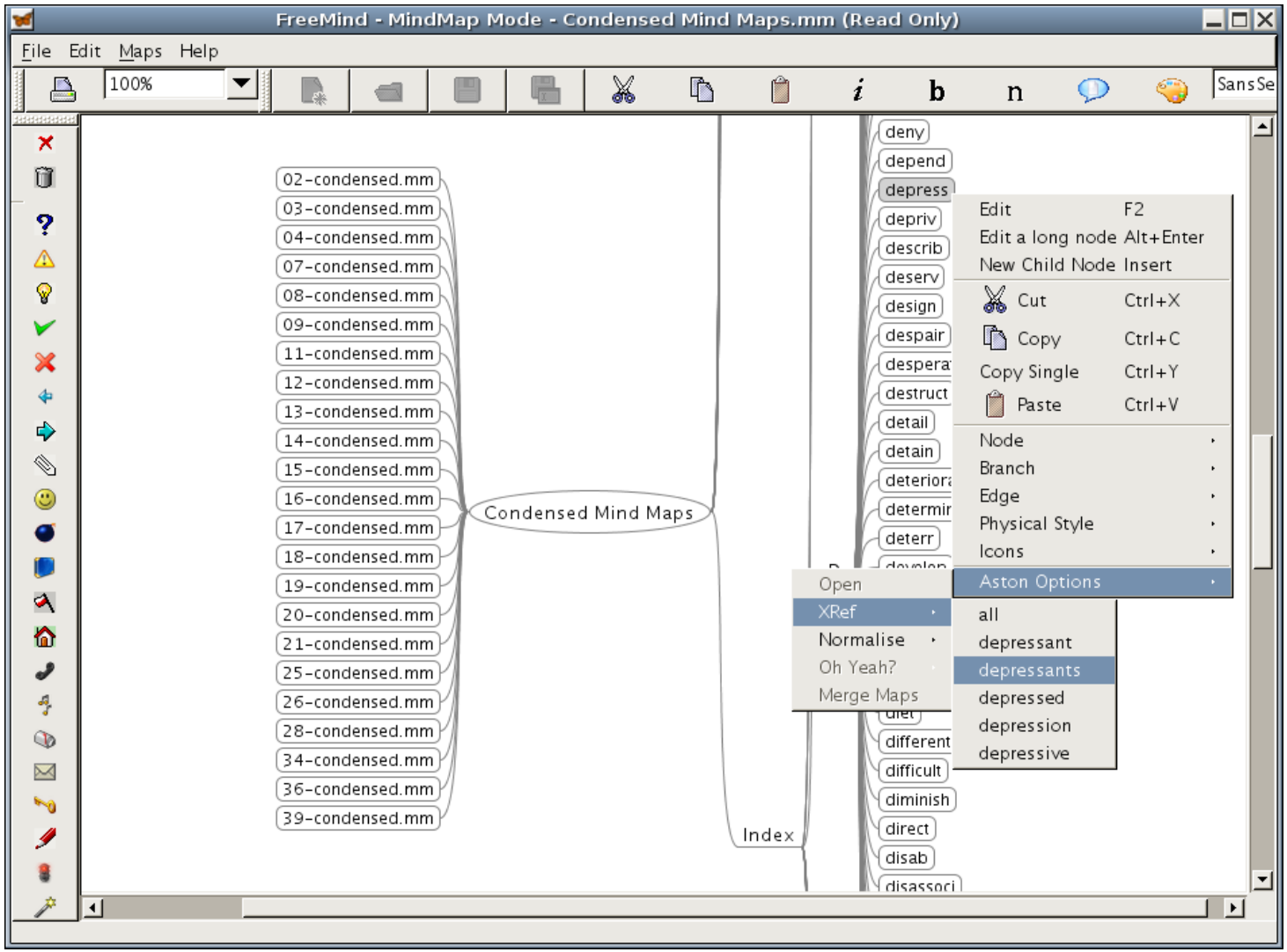

Figure 10.5: Details from the 'XRef' option in Figure 10.2

The selected node from Figure 10.5, then, identified concepts conflated by the stem 'depress'; the corresponding right-click menu was modified to reflect available options. The selected cross-reference option gave a further sub-menu of possible search strings, to allow searching either for all conflated nodes or for specific morphological forms. In the letter case, that would have meant using the overall stem 'depress', which did not appear as an actual word in any GRiST mind map. The resulting mind map might be displayed in two ways: as a pop-up window, or as a full-screen mind map in which all permitted FreeMind options are available. Indeed, the 'Normalise' option works in a similar way; accordingly, Figure 10.3 overleaf shows such a normalised mind map in a pop-up window.

The pop-up shown in Figure 10.6, then, presents a normalised version of further paths to nodes conflated by 'abus'. Although a separate instance of the novel `AstonFreeMind` class, that pop-up in fact allowed just a small sub-set of FreeMind options; apart from moving the new mind map within its smaller panel, actions are restricted to folding and unfolding nodes:

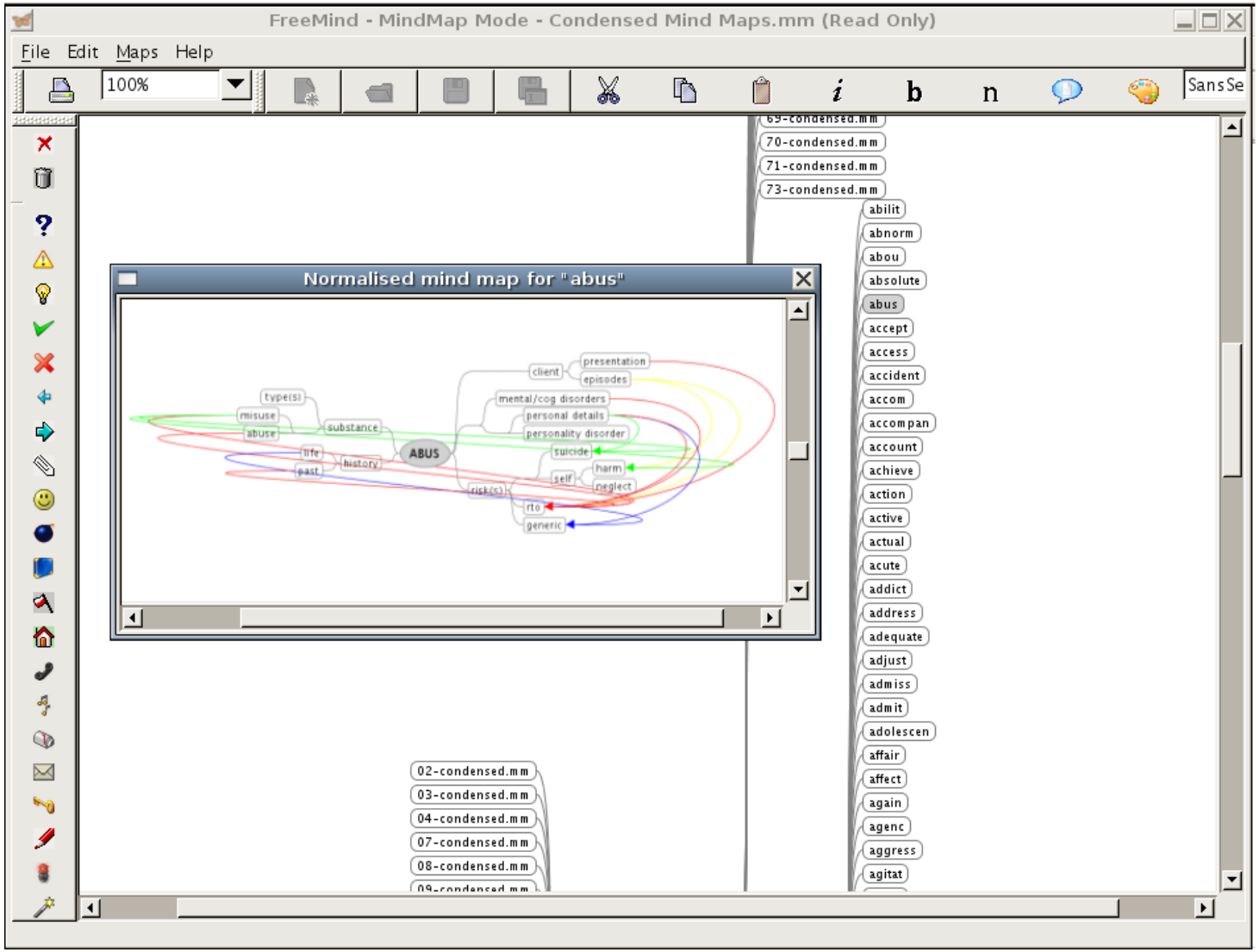

Figure 10.6: Details from the 'Normalise' option in Figure 10.2

The pop-up in Figure 10.6 resulted, in fact, from extra functionality encoded in the novel `AstonFreeMind` class; in this case, imbuing FreeMind with a capability for showing nested mind map panels. The pop-up from Figure 10.6, then, summarised paths to nodes expressing forms of 'abuse', with arrows showing any relationships to a separate group of corresponding risk categories. Using pop-up panels further avoided cluttering the workspace with multiple open mind maps; all the same, full mind maps may be generated on demand. When generating normalised mind maps, though, machines relied on metadata from spelling, CA, and stemming; it is important that such metadata are available for inspection, as shown next.

Berners-Lee (1997) suggests that machines might have a button marked 'Oh Yeah' that, when clicked, would display audit information that justifies decisions made automatically. In that spirit, such a button was implemented in `AstonFreeMind` to display metadata relevant to any generated node hierarchy. Figure 10.7 shows the menu that results from clicking the right mouse button on the node `[substance misuse]`; the existence of metadata for that node made the 'Oh Yeah' option available. A further sub-menu listed just a sole source of metadata, from WordNet:

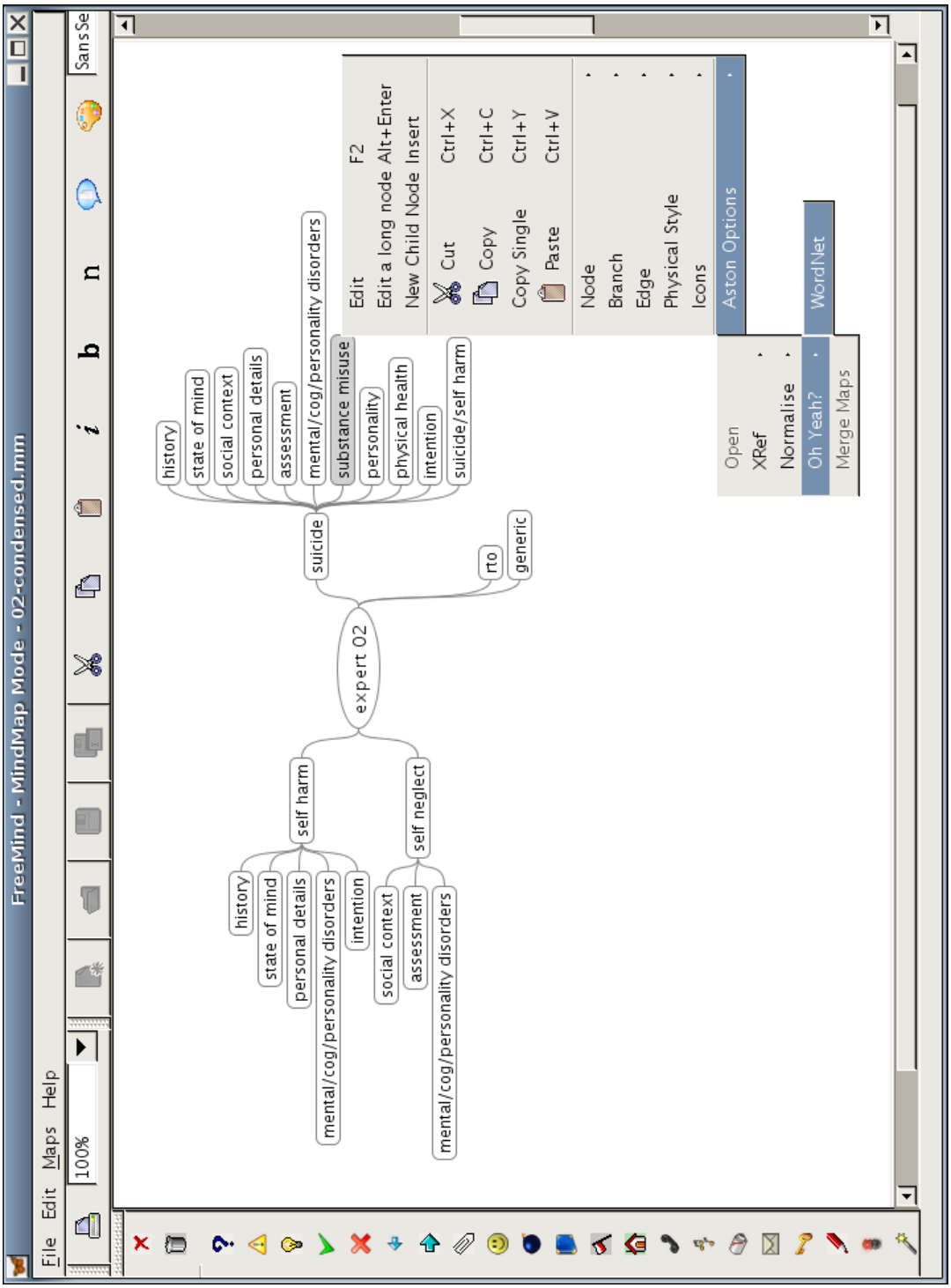

Figure 10.7: Details from the 'Normalise' option in Figure 10.7

The pop-up menu in Figure 10.7, then, showed WordNet metadata to be available for the node `[substance misuse]`. Selecting that option displayed another pop-up window populated with metadata from Xindice, so as to address the problem of trust in knowledge-based systems.

A further extension to FreeMind attempted to answer that question, by means of metadata from the Xindice database. Structures of intensional knowledge held as XML were described in Chapter 9; retrieving those elements by means of XPath queries allowed FreeMind pop-ups to display metadata associated with specific nodes. Although such keys are not visible in FreeMind itself, the underlying XML does store them; in that way, the key of any node selected in FreeMind was passed as an argument to XPath. Metadata retrieved from the XML database were subsequently formatted into a pop-up window, and displayed in FreeMind.

The means of displaying the 'Oh Yeah' pop-up window resembled that described for normalised mind maps. Accordingly, just detail of the pop-up resulting from clicking Figure 10.7 is reproduced as in Figure 10.8, which shows WordNet metadata for node number 13199, `[substance misuse]`. Two groups of WordNet results are evident: one for words related to 'substance', with further one noting the relationship between 'abuse' and 'misuse':

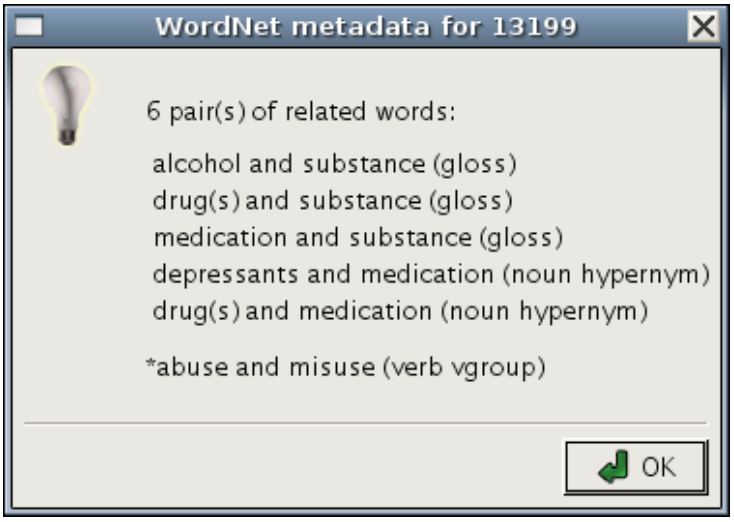

Figure 10.8: Detail of the 'Oh Yeah?' pop-up arising from Figure 10.7

The first three entries in Figure 10.8 show words that appeared in glosses, which constitute examples of word usage. The next two results come from mutual hypernym pointers in WordNet. Lastly, the relationship between 'abuse' and 'misuse' arose from a WordNet verb group, even though those words are, in fact, nouns; that result is marked with an asterisk to indicate that verb groups were less reliable sources of WordNet metadata.

The first group of words from Figure 10.8, then, arose from glosses for the word 'substance'. In that

way, the related words ‘alcohol’, ‘drug(s)’, and ‘medication’ were found, and used in drawing together nodes in generated mind maps. That word ‘medication’, though, further had acceptably close hypernyms ‘depressants’ and ‘drug(s)’. The first of those results was due to stripping the prefix ‘anti-’, while the latter provided further support for the corresponding gloss result. In addition, appending the plural suffix ‘(s)’ to the singular form showed that both words were detected in GRiST mind maps.

10.5 Chapter Summary

The introduction in Section 10.1 described 'Aston Options' that were provided by the novel `AstonFreeMind` Java class. Those additional facilities were available on the work space for GRiST that was shown in Section 10.2; certain of those options operated on entire mind maps, as was shown in Section 10.3. Such facilities included an ability to open several mind maps simultaneously, and further to generate merged versions of selected mind maps.

Further options available on the new GRiST workspace were described in Section 10.4; those options acted on individual nodes, rather than on whole mind maps. Pop-up menus added to FreeMind's standard 'nodes' menu further gave access to metadata gathered during this thesis. Those options comprised a means of generating cross-references of specified concepts, and for creating normalised knowledge structures; both options resulted in novel mind maps shown either in pop-up windows allowed by the `AstonFreeMind` class, or as additional FreeMind windows.

This new user interface, then, implements the work of this thesis by means of object-orientated programming techniques. The resulting GUI acts both as a convenient way of interacting with GRiST mind maps, and as a vehicle for presenting newly discovered knowledge. It further permits the interrogation of metadata, in order to justify decisions taken automatically by machines. All that remains is to summarise the chapters of this thesis, and to close with some conclusions and proposals for future research.

11

Summary and Conclusions

This chapter summarises the approach taken by this thesis to automating analyses of GRiST mind maps, and of treating them as an information base of mental health knowledge. Part I, then, introduced problems that arise from mental health disorders, and of the need for accurate risk assessments. Knowledge extracted from mind maps recoded by experts during Part I was subsequently applied in Part II, in order to normalise any knowledge they contained. Finally, Part III showed the implementation of the proposed approach, and benefits that it offered to GRiST. Those parts are now addressed in more detail, before and some conclusions close the chapter, and the thesis.

11.1 Summary

Part I: Extracting Knowledge from GRiST Mind Maps.
Part I of this thesis, then, started by introducing the role of risk analysis in mental health care. In that respect, the GRiST project provided important, expert knowledge to help front-line services in making assessments. Further, a pre-processing stage of data cleaning and abstraction was applied, which resulted in spelling corrections, and stems that identified related word forms. Corresponding chapters are now reviewed in more detail.

Chapter 1, then, introduced difficulties arising from mental health problems in the United Kingdom. Such problems harm sufferers themselves, in addition to family members and to the public at large. In recent years, though, problems have been exacerbated by so-called care in the community, rather than in institutions such as hospitals. As a result, workers in front-line services, although lacking any formal training in risk analysis, must evaluate any behavioural cues exhibited and decide whether to refer cases to better qualified experts in mental health.

To that end, the GRiST encapsulated experts' ideas about assessing mental health risks, and supports from-line services by means of a web-based interface; in fact, GRiST captured such knowledge by means of mind maps that recorded interviews with specialists. Because of the importance of mind mapping to GRiST, that technique was reviewed in terms of ideas espoused by its inventor. GRiST mind maps, though, justifiably strayed from those rigid guidelines; all the same, the ensuing richness of those mind maps hindered any automated interpretation by machines. Further, mind mapping lacks any intrinsic means of restricting the expression of knowledge; that allowed GRiST panellists to stray from the template that was provided.

After Chapter 1 closed with an overview of this thesis, Chapter 2 addressed mind maps in the wider context of 'semantic networks' such as concept maps, relational databases and knowledge bases. A review of mind mapping in knowledge engineering, though, showed mind maps to be restricted to brainstorming,

which contributed knowledge to more formal semantic networks. In sharp contrast, viewing GRiST mind maps as an information base of metal health knowledge made them amenable to abstraction, by which related formats are refined. Applying abstraction to GRiST mind maps yielded a structure for storing knowledge that augmented any inherent hierarchical associations.

Chapter 3 began that process of abstraction by dealing with spelling mistakes, which detracted from isolating key concepts in GRiST mind maps. A review of automated spelling correction, though, revealed that 'non-words' might be valid words that happened to be missing from standard dictionaries. In fact, support for taking words as presented, or for determining acceptable suggestions, came from words encoded in GRiST mind maps themselves.

Spelling corrections were further refined by means the Levenshtein Distance, L. Although that algorithm was designed to correct transmission errors arising from noisy channels, it has become widely used for comparing textual information. After reviewing approaches to using and refining L, an adjusted version was proposed for improving spelling correction in GRiST mind maps, in addition to revealing valid, novel terms. Experiments showed that approach to be largely successful, in that unsuitable candidate corrections were declined, sometimes in favour of taking words as they were entered by GRiST experts.

Abstraction was furthered in Chapter 4 by means a 'stemming', which identified invariant portions of related word forms. Rather than relying on linguistic rules, though, the Levenshtein Distance that Chapter 3 applied to spelling correction was employed. In that way, stems were extracted from mind map nodes purely by identifying morphological similarities. Experiments subsequently showed that approach to successfully extract key concepts from GRiST mind maps, regardless of particular word forms. That approach further identified reliable pre- and suffixes, in addition to words that should be excluded from conflation, in order to prevent over-stemming.

Part II: Applying Knowledge to GRiST Mind Maps.

Having successfully identified novel concepts, corrected genuine spelling mistakes, and extracted stems from GRiST mind maps, Part II applied any such knowledge to discerning related nodes; that further involved the popular WordNet tool, by which Chapter 5 revealed related meanings between words from GRiST mind maps, rather than related forms. There were, though, problems in using that tool, namely, a lack of coverage for stop words, and an innate ambiguity between certain entries. Whereas stop words are commonly ignored, or removed prior to any main analysis, the solution proposed here lay in existing lists of stop words. The first challenge posed by WordNet, then, was met by an extension to WordNet that mimicked synsets reported for content words.

The more difficult problem of ambiguity, though, was shown as amenable to a technique called 'clus-

tering'; the specific form of clustering used here was CA, which Chapter 6 introduced as a means of resolving ambiguity in WordNet. After describing the underlying mechanisms of CA, which draws on classical mechanics, existing approaches were reviewed that applied CA to researching plain text. Those studies suggested a way to determine patterns of word usage around prepositions, which might subsequently resolve ambiguous cases in GRiST mind maps. Although interpreting CA results is generally a human activity, automating CA for GRiST mind maps allowed machines to derive such patterns unaided.

Subsequently, Chapter 7 presented results from overcoming the problems posed by WordNet, described in Chapter 5. Experiments started by identifying 'triples' of words, each triple comprising a word, followed by a preposition, with a further word after that. CA on triples composed of just unambiguous words revealed reliable patterns of usage, which subsequently suggested most likely interpretations of ambiguous words from further, novel triples. Instead of words themselves, though, CA involved so-called 'meta types' that reflected roles that words play in phrases, as 'things', as 'actions', or as 'modifiers'. That knowledge further augmented the hierarchical associations inherent in mind maps, enriching the emerging information base of mental health knowledge.

Chapter 8 was concerned with generating idealised, combined versions of the collected GRiST mind maps, from nodes identified by an amalgamation of CA, stemming, and WordNet; the ensuing mind maps provided overall knowledge structures for particular concepts. After drawing related meanings from WordNet, that chapter described the hierarchical nature of CA clusters that contributed nodes to any combined mind map. Experiments in refining mind map structures first addressed just nodes grouped by stems; subsequently, those mind maps were augmented by semantically related nodes identified by WordNet.

Part III: Implementation, Summary and Conclusions.

The last part of this thesis looked at the system that arose from preceding parts. Chapter 9 introduced the XML format in which FreeMind encodes mind maps, and further introduced a native XML database; that database stored both GRiST mind maps, and any knowledge gathered by experiments in earlier chapters; such additional knowledge constituted 'metadata' that described existing data encoded in mind maps. The Xindice database further responded to queries from both machines and humans, about either type of data. The resulting XML-centric system was ideally suited to Internet applications.

Following that discussion of implementation issues, Chapter 10 demonstrated the gains that accrued from this thesis. Such benefits included extensions to FreeMind that communicated with the XML database, and provided a work space for GRiST researchers. One such extension generated refined mind maps on demand, while a further one aided navigation between related nodes. Importantly, metadata that supported any automated decisions were made available to humans by means of an 'Oh Yeah?' button

inspired by Berners-Lee (1997). That chapter closed by considering the implementation of this approach in Java and XML. That, then, brought the thesis to this point: Chapter 11, which summarised the approach taken here to normalising and enriching GRiST mind maps. Accordingly, the final conclusion follows next.

11.2 Success in Answering Research Questions

Chapter 1 raised research questions that are re-stated here, with an indication of what success was achieved in answering them:

- *Can a theoretical framework be provided for mind mapping?* Chapter 2 showed the answer to that question to a resounding "yes". The family of related representations known as semantic networks readily accepts mind maps as its latest addition. Many similarities exist between members of that group, with mind maps differing mainly in a lack of control over content. That was overcome by extracting intensional knowledge after having created mind maps, rather than beforehand in other types of semantic network. As a result, mind maps might be queried in a similar way to relational databases, or knowledge bases held as ontologies such as concept maps, or structures in OntoEdit. That was made possible by, as Date (1975, 2003) says, "putting one thing in one place", just as in those more formal representations currently employed by the research community.
- *Is it possible to improve on existing spelling correction algorithms?* Again, the answer is "yes", as was shown in Chapter 3. By referring to the actual corpus of text under investigation, appropriate corrections were accepted, and novel yet valid words retained as used by mind map authors. Further, an adjusted form of the Levenshtein Distance promoted acceptance of suggestions that were further down Jazzy's lists. In short, evaluating suggestions in that way led to great improvements in spelling correction for GRiST mind maps. Medical terminology found in those mind maps was treated as such, instead of being replaced by inappropriate candidates from Jazzy's dictionaries.
- *Is it possible to improve on existing stemming techniques?* Results from experiments that used an adapted Levenshtein Distance showed that, indeed, stemming need not rely on linguistic rules; in some cases, such knowledge actually impaired stemming. Although the Porter stemmer (Porter, 2006) was rejected primarily for its reliance on linguistic rules, results show a purely text-based approach to work better, especially when any corpus is considered as a source of knowledge in itself.

(continued overleaf)

- *What can be done to resolve the ambiguity evident in WordNet?* Ambiguity in WordNet obscured exact meanings of words used by human authors. That problem was overcome here by means of CA's clustering facility. Ambiguous words that appeared to either side of various prepositions were disambiguated by considering that context. In addition, words determined to be nouns yielded subjects and objects for actions suggested by words best treated as verbs. By those means, then, deficiencies in WordNet were successfully mitigated.
- *How might the structure of mind maps be refined automatically?* Mind mapping differs from related digital formats for representing knowledge, in exerting no control over the structure of mind maps. However, in deriving such rules after the event, mind maps have been shown amenable to normalisation. Normalised paths of nodes, when passed through CA, yielded an idealised version of whatever concept was under consideration. In that way, variation in expressing ideas was overcome, allowing mind mapping is to be treated as a formal representation of knowledge.

11.3 Wider Applications and Future Work

In addition to the GRiST mind maps, mind mapping in general would benefit from discoveries made in this thesis. In fact, the inventor of mind mapping lists five key uses of mind mapping. Those uses can be paraphrased as:

1. Self-Analysis: review past achievements and project future goals.
2. Problem Solving: gain an in-depth understanding of a problem by mind mapping it.
3. Thinking: free the thinking process by creating branches from a central image.
4. Teaching: mind mapping encourages flexibility in delivering talks.
5. Management: improve communication, and help focus marketing and promotion (Buzan, 1996, 2003).

GRiST mind maps largely reflected item 3 from the above list, namely, thinking. In education, though, Buzan (1996, 2003) considers just teachers' use of mind mapping as an aid to preparing and giving lessons. That goes against personal experience of teaching in English and French Secondary schools, where mind maps proved more popular with pupils than with teachers, for uses $1-3$ inclusively: self-analysis, problem solving, and thinking. Benefits to management, number 5 in the list, seem to reflect uses $1-4$, rather than any specific advantage.

Whatever use mind maps finds, techniques invented here would help in discerning patterns that are not immediately evident. Notably, a single mind map might be generated to give an overall view of a particular domain. Further, adjuncts to FreeMind would allow users to use mind maps as an information

base that responds to queries. In that way, sales managers, for example, could more easily assimilate ideas from their sales force by having them create mind maps. In Education, teachers might obtain a combined view of pupils' thoughts on, say, on-line abuse, or of problems facing children having Special Educational Needs (SEN). For general mind mapping by individuals, ideas from successive mind maps made for usages 1 and 3, self-analysis and thinking, might be compared or combined on a home PC.

This thesis further contributed novel algorithms for spelling correction and stemming. Those, in turn, relied on an adapted Levenshtein Distance, and an automated form of Correspondence Analysis. Both of those analyses might find applications outside mind mapping. At the time of writing, people in general are becoming aware of the potential for finding patterns in data, for example through television series that depict mathematicians assisting the police, or by IBM's advertising campaign. Echoing the uses of mind mapping listed by Buzan (1996, 2003), managers might find trends in sales data, while teachers might detect patterns in factors that affect pupils' academic and social well-being. Of particular interest are ways in which clustering might assist the Police in identifying patterns in criminal activities.

A further application of this research might be in analysing DNA sequences. Although L and CA are already used by molecular biologists (Tylera, Hortona, & Krause, 1991), work from this thesis might help in interpreting whatever results arise. For the former, identifying repeated strings within strings would turn stemming to a non-linguistic purpose, while for the latter, automated CA could help humans in interpreting whatever clusters and outliers emerge.

Future work, then, aims to bring techniques developed here to the public at large. That will be done, in part, by a web site for KnowWare (Priscott, 2011), which will feature "knowledge-based wares". In addition to a separate domain for mobile devices, an Android "app" is planned to fully integrate knowledge-based offerings on hand-held devices. As well as CA and Edit Distance options. that new site will host a 'mind map farm', which will allow users to query and reconfigure mind maps. Rather than requiring FreeMind to be installed on local machines, KnowWare will offer a centralised service based on a Xindice XML database. A Java servlet reads various configuration files from the server, and prepares a HTML page. FreeMind itself runs from a Java archive downloaded by an Applet, rather than as an executable on client machines. The word 'farm' reflects that mind map authors might share, view and analyse mind maps as a community.

11.4 Conclusions

This thesis opened with an overview of the problems caused in the U.K. by mental health problems, and of the GRiST project that disseminates expert knowledge to front-line services. There was, though, a great deal of knowledge left untouched by GRiST, particularly of knowledge of a non-hierarchical nature. This thesis, then, aimed to automate the processing of those mind maps, mimicking the process performed manually by Buckingham and Adams (2006). To that end, early chapters pre-processed those mind maps by correcting spelling errors, and by extracting stems that identify key concepts.

The chosen tool for spelling correction and stemming was the Levenshtein Distance L. After adjustment for string lengths, L-based algorithms showed both qualitative and quantitative improvements over the Jazzy spelling checker and the Porter stemmer. Importantly, eschewing any built-in linguistic knowledge actually yielded better results. That is not to argue for ignoring such linguistic knowledge entirely, but to apply it as needed rather than as a fundamental aspect of any algorithm.

Armed with spelling corrections and stems, the second phase of this thesis applied knowledge to GRiST mind maps, rather than extracting it from them. In that respect, the WordNet tool proved very useful, but bore the dual disadvantages of a lack of widely used words such as prepositions, and of ambiguous entries for many words. Both of those problems were overcome, by means of an adjunct to WordNet, and by performing CA on matrices of prepositions and pairs of meta-types. As a result, WordNet was proved to be a more useful and discerning tool. Further, automating CA allowed machines to successfully determine clusters of 'actions', 'things' and 'modifiers' that revealed more precisely what GRiST authors meant when using ambiguous words in conjunction with prepositions.

As a result, the third part of this thesis described a XML-centric system that managed mental health knowledge, and made it available in novel way that benefit GRiST as a whole. Rather than using the LISP language for modelling knowledge structures, the system described here relied purely on Java and XML, and used an experimental d6database called Xindice at its heart. That combination of emerging Internet technologies, in addition to benefiting GRiST, would be useful in any web context; indeed, once mature, such XML databases are set to displace relational databases as the most appropriate way of storing, analysing, and transporting knowledge. GRiST, then, will benefit from this research, as will the application of mind mapping in general, and of the use of XML technologies in web-based settings. I therefore offer this thesis as worthy of being conferred the degree of Doctor of Philosophy.

Keith Priscott, August 2011.

References

Aas, K., & Eikvil, L. (1999). *Text Categorisation: A Survey.* Norwegian Computing Centr-e. (From http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.2236&rep=rep1&type=pdf)
(pages 98, 99, 128, and 145)

Ackroyd, M. (1980). Isolated Word Recognition using the Weighted Levenshtein Distance. *Acoustics, Speech, and Signal Processing*, *28*(2), 243–244. (From http://ieeexplore.ieee.org/ielx6/29/26145/01163382.pdf?tp=&arnumber=1163382&isnumber=26145)
(pages 10, 71, 74, 76, 77, and 79)

Adamchik, V. (2009). Homework Assignment 4 - Word Ladder. (URL: http://www.cs.cmu.edu/afs/cs/Web/People/adamchik/15-121/labs/HW-4%20Word%20Ladder/lab.html)
(pages 10 and 64)

Albert, D., & Steiner, C. (2005). Representing Domain Knowledge by Concept Maps: How to Validate Them? In T. Okamoto, D. Albert, T. Honda, & F. Hesse (Eds.), *The 2nd Joint Workshop of Cognition and Learning through Media* (pp. 169–174). (From http://hal.archives-ouvertes.fr/docs/00/19/04/01/PDF/Albert-Dietrich-2005.pdf)
(pages 53 and 56)

Basili, R., Marziali, A., Pazienza, M., & Velardi, P. (1996). Unsupervised Learning of Syntactic Knowledge: Methods and Measures. In *Conference on Empirical Methods in Natural Language Processing* (p. 23-32). (From http://www.aclweb.org/anthology-new/W/W96/W96-0203.pdf)
(pages 142, 143, 144, and 148)

Beckwith, R., Miller, G., & Tengi, R. (1993). Design and Implementation of the WordNet Lexical Database and Searching Software. *Technical Report*. (URL: http://www.cogsci.princeton.edu/~wn/5papers.pdf)
(pages 141, 177, 228, and 243)

Bekkerman, R., & Allan, J. (2005). *Using Bigrams in Text Categorization* (Internal Report No. 408). Department of Computer Science: University of Massachusetts. (URL: http://maroo.cs.umass.edu/pub/web/getpdf.php?id=552)
(pages 32, 133, 142, 143, 144, 145, 149, and 153)

Benzécri, J. P. (1992). *Correspondence Analysis Handbook*. New York: Marcel Dekker.
(pages 11, 153, 154, 155, 160, and 164)

Berners-Lee, T. (1997). Cleaning up the User Interface. World Wide Web Consortium. (URL: http://www.w3.org/DesignIssues/UI.html)
(pages 302, 315, and 324)

Berners-Lee, T. (1998). RDF Anonymous Nodes and Quantification. World Wide Web Consortium (W3C). (URL: http://www.w3.org/DesignIssues/Anonymous.html)
(page 258)

Biplab, K., Wallace, P., Wallace, S., & Gill, W. (2008). Some observations on Mind Map and Ontology Building Tools for Knowledge Management. *Ubiquity*, *9*(9), 1–9. (From http://www.acm.org/ubiquity/volume_9/pf/v9i9_sarker.pdf)
(pages 47, 51, and 53)

Blake, C. (2007). The Role of Sentence Structure in Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing* (pp. 101–106). Association for Computational Linguistics. (From http://acl.ldc.upenn.edu/W/W07/W07-1417.pdf)
(pages 11, 146, 147, and 151)

Booth, B. (1979). Assessing Volcanic Risk. *Journal of the Geological Society*, *136*(3), 331–340.
(page 20)

Brants, T. (2003). Natural Language Processing in Information Retrieval. In *Computational Linguistics in the Netherlands (CLIN).* University of Antwerp. (From http://www.cnts.ua.ac.be/clin2003/proc/03Brants.pdf)
(pages 33, 142, 144, and 145)

Brill, E., & Moore, R. (2008). *Spell Checker with Arbitrary Length String-to-String Transformations* (U.S. Patent No. 7366983). (From http://www.freepatentsonline.com/7290209.pdf)
(pages 64, 66, 68, 69, 70, 75, 76, 94, and 95)

British Psychological Society. (2002). *Understanding Mental Illness - Recent Advances in Understanding Mental Illness and Psychotic Experiences.* BPS Division of Clinical Psychology. (Extract from http://www.mentalhealthcare.org.uk/content/?id=60)
(pages 19 and 20)

Buckingham, C. (2007). Improving Mental Health Risk Assessment using Web-Based Decision Support. *Health Care Risk Report*, *13*(3), 17–18. (From https://www.grist.galassify.org/docs/hcrr-published.pdf)
(pages 22, 23, and 26)

Buckingham, C., & Adams, A. (2005). Can You Help Us Develop a Mental-Health Decision-Support System?
(URL: https://www.grist.galassify.org/docs/bjhc-letter.pdf)
(page 22)

Buckingham, C., & Adams, A. (2006). Employing XML-Based Technologies to Elicit Mental-Health Risk Knowledge. In J. Bryant (Ed.), *Current Perspectives in Healthcare Computing 2006* (pp. 284–291). BCS HIF: Swindon. (From http://www.galassify.org/grist/development/docs/cdb-hc2006.pdf)
(pages 22, 28, 32, 34, 42, 47, 49, 51, 53, 54, 57, 58, 94, 129, 130, 145, 219, 229, 247, 248, 256, 257, 266, 273, 274, 276, 292, and 327)

Buckingham, C., Adams, A., & Mace, C. (2007). Cues And Knowledge Structures Used By Mental-Health Professionals When Making Risk Assessments. *Journal of Mental Health*, 1–16. (From `https://www.grist.galassify.org/docs/jmh-published.pdf`)
(pages 21 and 215)

Buckingham, C., Ahmed, A., & Adams, A. (2007). Using XML and XSLT for Flexible Elicitation of Mental-Health Risk Knowledge. *Medical Informatics and the Internet in Medicine*, *32*(1), 65–81. (From `https://www.grist.galassify.org/docs/cdb-hc2006.pdf`)
(pages 10, 23, 24, 25, 33, 219, 220, 273, and 276)

Buckingham, C., Kearns, G., Brockie, S., Adams, A., & Nabney, I. (2004). Developing a Computer Decision Support System for Mental Health Risk Screening and Assessment. In J. Bryant (Ed.), *Current Perspectives in Healthcare Computing 2004* (pp. 189–194). BCS HIF: Swindon. (From `http://www.galassify.org/grist/development/docs/grist-hc2004.pdf`)
(pages 10, 20, 21, 22, 41, 42, 52, 133, 160, 215, and 227)

Bullen, R., Cornford, D., & Nabney, I. (2003). Outlier Detection in Scatterometer Data: Neural Network Approaches. *Neural Networks*, *16*(3–4), 419–426.
(pages 150 and 172)

Buzan, T. (1974). *Use Your Head*. BBC Books.
(pages 26, 28, 45, and 139)

Buzan, T. (1996). *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Plume Books.
(pages 26, 28, 45, 56, 133, 139, 325, and 326)

Buzan, T. (2003). *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. Plume Books.
(pages 26, 28, 139, 278, 325, and 326)

Buzan, T. (2008). iMindMap™ version 3.
(URL: `http://www.imindmap.com/`)
(pages 10, 26, 27, and 32)

Cañas, A. J., & Carvalho, M. (2004). Concept Maps and AI: An Unlikely Marriage? *Revista Brasileira de Informticana Educação*. (From `http://www.ihmc.us/users/acanas/Publications/ConceptMapsAI/Canas-CmapsAI-Sbie2004.pdf`)
(pages 10, 43, 44, 45, and 216)

Clark, J., & Derose, S. (1999). XML Path Language (XPath). The World Wide Web Consortium (W3C).
(URL: `http://www.w3.org/TR/xpath/`)
(pages 285, 286, and 291)

Collins, A., & Quillian, R. (1969). Retrieval Time from Semantic Memory. *Journal of Verbal Learning and Verbal Behavior*, *8*(2), 240–247. (From `love.psy.utexas.edu/~love/concepts/CollinsQuillan1969.pdf`)
(pages 10, 134, 135, and 136)

Computer Dictionary. (2010). LoveToKnow, Corp. (URL: http://computer.yourdictionary.com/) (page 277)

Crowell, J., Zeng, Q., & Kogan, S. (2003). A Technique to Improve the Spelling Suggestion Rank in Medical Queries. In *AMIA 2003 Symposium Proceedings.* (From http://ukpmc.ac.uk/ukpmc/ncbi/articles/PMC1480059/pdf/amia2003_0823.pdf) (page 64)

Date, C. J. (1975). *An Introduction to Database Systems.* Addison-Wesley. (pages 54, 55, 56, 232, and 324)

Date, C. J. (2003). *An Introduction to Database Systems.* O'Reilly. (pages 54, 55, 56, 57, 246, and 324)

Debenham, J. (1996). Integrating Knowledge Base and Database. In *SAC '96: Proceedings of the 1996 ACM symposium on Applied Computing* (pp. 28–32). ACM. (From http://doi.acm.org/10.1145/331119.331135) (pages 52 and 56)

Debenham, J. (1998). Representing Knowledge Normalisation. In *Tenth International Conference on Software Engineering and Knowledge Engineering, SEKE'98* (pp. 132–135). (From https://eprints.kfupm.edu.sa/62279/1/62279.pdf) (page 56)

Dolamic, L., & Savoy, J. (2008). Stemming Approaches for East European Languages. In *Advances in Multilingual and Multimodal Information Retrieval* (pp. 37–44). Springer-Verlag. (From http://www.springerlink.com/content/582xh2q7v5t2gw68/fulltext.pdf) (page 145)

Eastman, N. (1997, June). The Mental Health (Patients in the Community) Act 1995: A Clinical Analysis. *The British Journal of Psychiatry*, *170*(6), 492–496. (From http://bjp.rcpsych.org/cgi/reprint/170/6/492?ck=nck) (page 20)

Felder, R., & Spurlin, J. (2005). Applications, Reliability and Validity of the Index of Learning Styles. *International Journal of Engineering Education*, *21*(1), 103–112. (From http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/ILS_Validation(IJEE).pdf) (page 26)

Fellbaum, C. (1990). English Verbs as a Semantic Net. *International Journal of Lexicography*, *3*(4), 278–301. (Contained in http://wordnetcode.princeton.edu/5papers.pdf) (pages 136, 138, and 267)

Fellbaum, C., Gross, D., & Miller, K. (1990). Adjectives in WordNet. *International Journal of Lexicography*, *3*(4), 265–277. (Contained in http://wordnetcode.princeton.edu/5papers.pdf) (pages 136, 138, 151, and 182)

Fischer, E. (1998). The Many Uses of XML.
(URL: http://www.sis.pitt.edu/~mbsclass/standards/fischer/XMLUses.html)

(page 273)

Gilleland, M. (2003). Levenshtein Distance, in Three Flavors.
(From http://www.merriampark.com/ld.htm)
(page 74)

Gruber, T. R. (1995). Towards Principles for the Design of Ontologies used for Knowledge Sharing. *International Journal of Human-Computer Studies*, *43*, 907–928.
(page 49)

Hefke, M., & Stojanovic, L. (2004). An Ontology-based Approach for Competence Bundling and Composition of Ad-hoc Teams in an Organisation. In *Proceedings of I-KNOW 04 Graz, Austria* (pp. 126–134). (From http://i-know.know-center.tugraz.at/previous/i-know04/papers/hefke.pdf)
(pages 10, 49, 50, 51, and 53)

Hegazy, S., & Buckingham, C. (2008). An Algorithm for Robust Relative Influence Values Elicitation (ARRIVE). In *Proceedings of the 2008 Third International Multi-Conference on Computing in the Global Information Technology* (pp. 91–96). IEEE Computer Society. (From http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04591351)
(pages 42 and 133)

Higuera, C. de la, & Micó, L. (2008). A Contextual Normalised Edit Distance. In E. Chávez & G. Navarro (Eds.), *1st International Workshop on Similarity Search and Applications* (pp. 61–68). (From http://www.sisap.org/2008/talks/A-Contextual-Normalised-Edit-Distance.pdf)
(pages 10 and 78)

Horrocks, I., Li, L., Turi, D., & Bechhofer, S. (2004). The Instance Store: DL Reasoning with Large Numbers of Individuals. In *Proceedings of the 2004 description logic workshop (dl 2004)* (pp. 31–40). (From http://www.cs.man.ac.uk/~horrocks/Publications/download/2004/HLTB04a.pdf)
(page 52)

Horrocks, I., Sattler, U., & Tobies, S. (2000). Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, *8*(3), 239–264. Available from http://web.comlab.ox.ac.uk/people/Ian.Horrocks/Publications/download/2000/HoST00.pdf
(page 145)

Idzelis, M. (2005). Jazzy - Java Spell Check API.
(URL: http://jazzy.sourceforge.net/)
(pages 69 and 80)

Izumi, E., Uchimoto, K., & Isahara, H. (2007). Vocabulary Usage and Intelligibility in Learner Language. In M. Davies, P. Rayson, S. Hunston, & P. Danielsson (Eds.), *Proceedings of the Corpus Linguistics Conference (CL2007).* University of Birmingham. (From http://ucrel.lancs.ac.uk/publications/CL2007/paper/159_Paper.pdf)
(pages 11, 168, 169, and 214)

Jeffery, K. (2000). METADATA: The Future of Information Systems. In B. S., E. Lindencrona, & A. Slvberg (Eds.), *Information Systems Engineering: State of the Art and Research Themes* (pp. 183–195). London: Springer. (From http://www.wmo.ch/pages/prog/www/WDM/ET-IDM/Doc-2-3.pdf)
(pages 277, 278, and 287)

Kernighan, M., Church, K., & Gale, W. (1990). A Spelling Correction Program Based on a Noisy Channel Model. In H. Karlgren (Ed.), *International Conference On Computational Linguistics, Morristown* (Vol. 2, pp. 205–210). (From http://www.ldc.upenn.edu/acl/C/C90/C90-2036.pdf)
(pages 64, 65, 66, 67, 68, 70, 94, and 95)

Kukich, K. (1993). Techniques for Automatically Correcting Words in Text. In *Proceedings of the 1993 ACM Conference on Computer Science* (pp. 377–439). ACM. (From http://dc-pubs.dbs.uni-leipzig.de/files/Kukich1992Techniqueforautomatically.pdf)
(pages 62, 63, 64, 65, 66, 67, 68, 83, 84, 87, 94, and 95)

Larkey, L., Ballesteros, L., & Connell, M. (2002). Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. In *SIGIR 2002* (pp. 275–282). ACM. (From http://portal.acm.org/citation.cfm?id=860528)
(pages 103, 104, 105, and 106)

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, *10*(8), 707–710. (Request from store at Birmingham University, shelf ID Q 60.A5D61)
(pages 69 and 70)

Lin, Y., Wei, J., Lee, G., & Lee, D. (2005). A Visualization Tool for the Sitemap of a Knowledge Portal and the Concept Map of Group Knowledge. In *Proceedings of I-KNOW 05 Conference.* (From http://i-know.know-center.tugraz.at/content/download/393/1556/file/Lin_paper.pdf)
(pages 51 and 89)

Lyras, D., Sgarbas, K., & Fakotakis, N. (2007). Using the Levenshtein Edit Distance for Automatic Lemmatization: A Case Study for Modern Greek and English. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence* (pp. 428–435). IEEE Computer Society. (From http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4410417)
(pages 75 and 76)

Mayfield, J., & McNamee, P. (2003). Single N-Gram Stemming. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 415–416). ACM. (From http://portal.acm.org/citation.cfm?id=860528)
(pages 98, 102, 105, 128, and 130)

Mcnamee, P., Mayfield, J., & Piatko, C. (2001). The HAIRCUT System at TREC-9. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)* (pp. 273–279). (From http://comminfo.rutgers.edu/~muresan/IR/TREC/Proceedings/t9_proceedings/papers/jhuapl.pdf)

(pages 67, 68, and 94)

Mihalcea, R., & Moldovan, D. (2000). Semantic Indexing using WordNet Senses. In *Proceedings of ACL Workshop on IR & NLP, Hong Kong.* (From `http://acl.ldc.upenn.edu/W/W00/W00-1104.pdf`)

(pages 148 and 182)

Miller, G. (1990). Nouns in WordNet: A Lexical Inheritance System. *International Journal of Lexicography*, *3*(4), 245–264. (Contained in `http://wordnetcode.princeton.edu/5papers.pdf`)
(pages 10, 134, 137, 138, and 145)

Miller, G., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1990). Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, *3*(4), 235–244. (Contained in `http://wordnetcode.princeton.edu/5papers.pdf`)
(pages 133, 134, 136, 137, 138, 148, and 150)

Mueser, K., Drake, R., & Resnic, S. (1997). Models of Community Care for Severe Mental Illness: A Review of Research on Case Management. *Schizophrenia Bulletin*, *24*(1), 37–74. (From `http://schizophreniabulletin.oxfordjournals.org/cgi/reprint/24/1/37.pdf`)
(page 20)

Murtagh, F. (n.d.). Input Data Coding in Correspondence Analysis.
(From `http://www.cs.rhul.ac.uk/home/fionn/papers/fm25.pdf`)
(pages 160 and 164)

Murtagh, F. (2005). *Correspondence Analysis and Data Coding with R and Java.* Chapman & Hall/CRC.
(From `http://astro.u-strasbg.fr/~fmurtagh/mda-sw/correspondances/`)
(pages 153, 171, 188, 190, 222, 223, 227, 230, and 235)

Murtagh, F. (2008). The Correspondence Analysis Platform for Uncovering Deep Structure in Data and Information.
(From `http://arxiv.org/pdf/0807.0908v2`)
(page 153)

Murtagh, F., Ganz, A., & McKie, S. (2008). The Structure of Narrative: the Case of Film Scripts.
(From `http://arxiv.org/pdf/0805.3799`)
(pages 153 and 164)

Murtagh, F., Mothe, J., & Englmeier, K. (2007). Ontology from Local Hierarchical Structure in Text. *CoRR*, *abs/cs/0701180*. (From `http://arxiv.org/pdf/cs/0701180v1`)
(pages 149, 153, 164, and 224)

Mylopoulos, J. (1998). Information Modelling In The Time Of The Revolution. *Information Systems*, *23*(3–4), 127–155. Available from `http://www.cs.toronto.edu/~jm/2507S/Readings/Survey.pdf`
(pages 33, 52, 53, 54, 55, and 56)

Nagypál, G. (2007). Ontology Development. In R. Studer, S. Grimm, & A. Abecker (Eds.), *Semantic Web Services* (pp. 107–134). Springer.

(page 46)

Nardi, D., & Brachman, R. (2002). An Introduction to Description Logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, & P. Patel-Schneider (Eds.), *The Description Logic Handbook* (pp. 1–40). Cambridge University Press. (From `http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-01.pdf`)
(pages 46, 52, 54, 133, and 145)

Navarro, G. (2001). A Guided Tour to Approximate String Matching. *ACM Computer Surveys*, *33*(1), 31–88. (From `http://security.riit.tsinghua.edu.cn/seminar/2006_3_9/2001-A_Guided_Tour_to_Approximate_String_Matching.pdf`)
(pages 10, 64, 69, 70, 71, 72, 74, 76, 77, and 92)

Nishina, Y. (2007). A Corpus-Driven Approach to Genre Analysis: The Reinvestigation of Academic, Newspaper and Literary Texts. *Empirical Language Research (ELR) Journal*, *1*(2). (From `http://ejournals.org.uk/ELR/article/2007/2`)
(pages 11, 15, 161, 162, 163, 164, 169, 213, 222, and 223)

Novak, J. D., & Cañas, A. J. (2006). The Origins of the Concept Mapping Tool and the Continuing Evolution of the Tool. *Information Visualization*, *5*(3), 175–184. (From `http://cmap.ihmc.us/Publications/ResearchPapers/OriginsOfConceptMappingTool.pdf`)
(pages 45, 216, and 265)

Object Refinery Limited. (2005–2009). JFreeChart.
(URL: `http://www.jfree.org/jfreechart/`)
(page 188)

ONS. (2000). *Psychiatric Morbidity among Adults living in Private Households 2000: Main Report*. The Stationery Office. (From `http://www.data-archive.ac.uk/doc/4653uide2.pdf`)
(pages 19 and 247)

Orengo, V., & Huyck, C. (2001). A Porter-Like Stemming Algorithm for the Portuguese Language. In *International Symposium on String Processing and Information Retrieval* (pp. 186–193). IEEE Computer Society. (From `www.cwa.mdx.ac.uk/chris/Search/stemmer.doc`)
(pages 101, 104, 105, and 129)

Paleo, B. W. (2007). An Approximate Gazetteer for GATE based on Levenshtein Distance. In V. Nurmi & D. Sustretov (Eds.), *Proceedings of the Twelfth ESSLLI Student Session (European Summer School in Logic, Language and Information)* (pp. 197–208). (From `http://citeseerx.ist.psu.edu/viewdoc/download`?doi=10.1.1.100.1787&`rep=rep1&type=pdf`#`page=206`)
(pages 76 and 78)

Pedersen, T. (2001). A WordNet Stop List.
(URL: `http://www.d.umn.edu/~tpederse/Group01/WordNet/wordnet-stoplist.html`)
(pages 15, 149, 176, and 177)

Petch, E. (2001). Risk Management in UK Mental Health Services - an overvalued Idea? *Psychiatric Bulletin*, *25*, 203–205. (From `http://pb.rcpsych.org/cgi/content/full/25/6/203`)

(page 21)

Petkovic, T., Kostanjcar, Z., & Pale, P. (2005). E-Mail System for Automatic Hoax Recognition. In L. Budin & S. Ribaric (Eds.), *MIPRO 2005 XXVII. International Convention* (pp. 117–121). (From http://crosbi.znanstvenici.hr/datoteka/203908.CIS05.pdf)
(pages 92 and 116)

Polansky, D., & Foltin, C. (2010). FreeMind - Free Mind Mapping Software. (URL: http://freemind.sourceforge.net/wiki/index.php/Main_Page)
(pages 28 and 273)

Porter, M. (1980). An Algorithm for Suffix Stripping. *Program*, *14*(3), 130–137. (From http://tartarus.org/~martin/PorterStemmer/def.txt)
(pages 14, 98, 99, 100, 101, 104, 106, and 128)

Porter, M. (2006). The Porter Stemming Algorithm. Tartarus.Org. (URL: http://tartarus.org/~martin/PorterStemmer/)
(pages 127 and 324)

Priscott, K. (2011). On the Road to KnowWare. KnowWare U.K. (URL: http://p.knowwareuk.com/)
(page 326)

Reich, J. R., Brockhausen, P., Lau, T., & Reimer, U. (2002). Ontology-Based Skills Management: Goals, Opportunities and Challenges. *Journal of Universal Computer Science*, *8*(5), 506–515. (From http://www.jucs.org/jucs_8_5/onlogogy_based_skills_management/Reich_J_R.html)
(pages 49, 51, and 53)

Resnik, P. (1995). Disambiguating Noun Groupings with Respect to WordNet Senses. In D. Yarovsky & K. Church (Eds.), *Proceedings of the Third Workshop on Very Large Corpora* (pp. 54–68). Association for Computational Linguistics. (From http://acl.ldc.upenn.edu/W/W95/W95-0105.pdf)
(pages 11, 142, 144, and 148)

Sjöbergh, J. (2005). Chunking: an Unsupervised Method to Find Errors in Text. In S. Werner (Ed.), *Proceedings of the 15th NODALIDA Conference* (pp. 180–185). (From http://phon.joensuu.fi/lingjoy/01/sjoberghF.pdf)
(page 147)

Sky News. (2008). Hanged: Another Bridgend Suicide. (URL: http://news.sky.com/skynews/Home/UK-News/Bridgend-Suicide-Youth-17-Found-Hanged/Article/200812415195276)
(page 19)

Social Exclusion Unit. (2003). *Economic and Social Costs of Mental Illness.* The Sainsbury Centre for Mental Health.
(page 19)

Sowa, J. (1992). Semantic Networks. In S. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence, second edition.* Wiley, New York. (2006 Update from http://www.jfsowa.com/pubs/semnet.htm)

(pages 10, 28, 41, 46, 52, and 256)

Sowa, J. (2006). The Challenge of Knowledge Soup. In J. Ramadas & S. Chunawala (Eds.), *Research Trends in Science, Technology and Mathematics Education* (pp. 55–87). (From `www.jfsowa.com/talks/challenge.pdf`)
(pages 46, 52, and 56)

Sun Microsystems Inc. (n.d.). Java Downloads for Linux.
(URL: `http://www.java.com/en/download`)
(page 292)

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., & Wenke, D. (2002). OntoEdit: Collaborative Ontology Development for the Semantic Web. In *The Semantic Web - ISWC 2002: First International Semantic Web Conference, Sardinia, Italy, June 9-12.* (From `http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/2002_iswc_ontoedit.pdf`)
(pages 10, 45, 47, 48, 49, 51, 56, 89, and 223)

Sure, Y., Staab, S., & Studer, R. (2002). Methodology For Development And Employment Of Ontology Based Knowledge Management Applications. *SIGMOD Record*, *31*(4), 18–23. (From `http://www.sice.umkc.edu/~leeyu/class/CS690L/Reference/3.Sure.pdf`)
(page 49)

The Apache Software Foundation. (2010). Apache Tomcat.
(URL: `http://tomcat.apache.org/`)
(page 280)

The Apache XML Project. (2007). Xindice.
(URL: `http://xml.apache.org/xindice/`)
(page 279)

The Children's Society. (2008). *Young people & self harm.* (URL: `http://www.selfharm.org.uk/default.aspa`)
(page 20)

The GRiST Project. (2007). GRiST Version 3.
(Home page: `http://www.galassify.org/grist/development/docs/grist-v3.pdf`)
(page 22)

The WordNet team at Princeton University. (2005). Five Papers about WordNet.
(From `http://wordnetcode.princeton.edu/5papers.pdf`)
(page 138)

Thuraisingham, B. M. (2002). *XML Databases and the Semantic Web.* Boca Raton, FL, USA: CRC Press, Inc.
(pages 273, 279, and 287)

Tono, Y. (1999). A Corpus-Based Analysis of Interlanguage Development: Analysing POS Tag Sequences of EFL Learner Corpora. In *Practical Applications in Language Corpora.* Peter Lang Publishing, Inc. (From `http://leo.meikai.ac.jp/~tono/paper/palc99.pdf`)

(pages 11, 167, 169, and 214)

Tsarkov, D., Horrocks, I., & Patel-Schneider, P. (2007). Optimising Terminological Reasoning for Expressive Description Logics. *Journal of Automated Reasoning*, *39*(3), 277–316. Available from http://www.cs.man.ac.uk/~horrocks/Publications/download/2007/TsHP07.pdf
(pages 55, 56, and 145)

Tylera, E. C., Hortona, M., & Krause, R. (1991). A Review of Algorithms for Molecular Sequence Comparison. *Computers and Biomedical Research*, *24*(1), 72–96.
(page 326)

Unmans, A. (1998). Correspondence Analysis of the Synoptic Gospels. *Literary and Linguistic Computing*, *13*(1), 1–13. (From http://llc.oxfordjournals.org/cgi/reprint/13/1/1.pdf)
(pages 11, 164, 165, 166, 169, 213, and 214)

Wilcox-O'Hearn, A., Hirst, G., & Budanitsky, A. (1998). Real-Word Spelling Correction with Trigrams: A Reconsideration of the Mays, Damerau, and Mercer Model. In *Proceedings of the 9th international Conference on Computational Linguistics and Intelligent Text Processing* (pp. 605–616). Springer-Verlag. (From http://ftp.cs.toronto.edu/pub/gh/WilcoxOHearn-etal-2008.pdf)
(pages 67, 68, 70, 87, 94, and 95)

Wilde, E. (2006). XML-Centric Application Development. *Report No. 242, Computer Engineering and Networks Laboratory (TIK), ETH Zürich*. (URL: http://dret.net/netdret/docs/wilde-tikrep242.pdf)
(pages 276, 296, and 300)

Xu, J., & Croft, B. (1998). Corpus-Based Stemming using Cooccurrence of Word Variants. *ACM Transactions on Information Systems*, *16*(1), 61–81. (From http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.41.2236&rep=rep1&type=pdf)
(pages 98, 99, 102, 103, 104, 105, 106, 113, and 128)

Yang, Y., & Pedersen, J. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning* (pp. 412–420). Morgan Kaufmann Publishers Inc. (URL: http://reference.kfupm.edu.sa/content/c/o/a_comparative_study_on_feature_selection_8242.pdf)
(pages 144, 145, and 177)