THE
ROBERT GORDON
UNIVERSITY
ABERDEEN

A Knowledge Acquisition Tool to Assist Case Authoring from Texts

Stella Maris Asiimwe

A thesis submitted in partial fulfilment
of the requirements of
The Robert Gordon University
for the degree of Doctor of Philosophy

March 2009

Abstract

Case-Based Reasoning (CBR) is a technique in Artificial Intelligence where a new problem is solved by making use of the solution to a similar past problem situation. People naturally solve problems in this way, without even thinking about it. For example, an occupational therapist (OT) that assesses the needs of a new disabled person may be reminded of a previous person in terms of their disabilities. He may or may not decide to recommend the same devices based on the outcome of an earlier (disabled) person. Case-based reasoning makes use of a collection of past problem-solving experiences thus enabling users to exploit the information of others' successes and failures to solve their own problem(s). This project has developed a CBR tool to assist in matching SmartHouse technology to the needs of the elderly and people with disabilities. The tool makes suggestions of SmartHouse devices that could assist with given impairments. SmartHouse past problem-solving textual reports have been used to obtain knowledge for the CBR system.

Creating a case-based reasoning system from textual sources is challenging because it requires that the text be interpreted in a meaningful way in order to create cases that are effective in problem-solving and to be able to reasonably interpret queries. Effective case retrieval and query interpretation is only possible if a domain-specific conceptual model is available and if the different meanings that a word can take can be recognised in the text. Approaches based on methods in information retrieval require large amounts of data and typically result in knowledge-poor representations. The costs become prohibitive if an expert is engaged to manually craft cases or hand tag documents for learning. Furthermore, hierarchically structured case representations are preferred to flat-structured ones for problem-solving because they allow for comparison at different levels of specificity thus resulting in more effective retrieval than flat structured cases.

This project has developed SmartCAT-T, a tool that creates knowledge-rich hierarchically structured cases from semi-structured textual reports. SmartCAT-T highlights important phrases in the textual SmartHouse problem-solving reports and uses the phrases to create a conceptual model of the domain. The model then becomes a standard structure onto which each semi-structured SmartHouse report is mapped in order to obtain the correspondingly structured case. SmartCAT-T also relies on an unsupervised methodology that recognises word synonyms in text. The methodology is used to create a uniform vocabulary for the textual reports and the resulting harmonised text is used to create the standard conceptual model of the domain. The technique is also employed in query interpretation during problem solving. SmartCAT-T does not require large sets of tagged data for learning, and the concepts in the conceptual model are interpretable, allowing for expert refinement of knowledge. Evaluation results show that the created cases contain knowledge that is useful for problem solving. An improvement in results is also observed when the text and queries are harmonised. A further evaluation highlights a high potential for the techniques developed in this research to be useful in domains other than SmartHouse. All this has been implemented in the SMARTER case-based reasoning system.

Acknowledgments

I would like to express my gratitude to my supervisors Professor Susan Craw, Dr. Nirmalie Wiratunga and Mr. Bruce Taylor for the invaluable advice, help, support and encouragement during the time I worked on this research. Susan has been a great help and did her best to cheer me up during this long journey that at times seemed endless. Nirmalie was good at pulling ideas out of her hair and was the reason I got to visit some very interesting places around Aberdeen... Bruce has been my supervisor, the domain expert and above all a friend. He was always kind and eager to dispense with advice and whatever support that he could.

I am deeply grateful for the invaluable feedback from my external and internal examiners and the anonymous reviewers especially those in the CBR community. I am eternally grateful to Professor Ashwin Ram who read the gist of my work, gave me valuable feedback and chatted with me about it when we met at conferences. I would also like to thank Dr. Derrick Bridge, Dr. Ayshe Goker and Dr. Stewart Watt for listening to me and giving me tips on a number of subjects. I will never forget Stuart Massie for always willing to listen to my theories, punch holes in them, and then listen some more; always. I cannot count the number of times he let me interupt his own work. Somehow, whatever he was doing could always wait! Dr. Gary Bindley who kindly accepted to be my expert in a domain I was clueless about, and Mr. Laurie Power and his lovely wife Edith for the nice meals and for proofreading my work. The KBS group at RGU who always listened to my ideas during our meetings and preparations for conferences. Amandine with whom I had immense fun at conferences.

My life in cold and grey Aberdeen would have been rather bleak if I had not had a few friends outside my research life. Ratiba my partner in crime, who almost infected me with her craziness! One of us had to be the calm and collected one and so I gladly volunteered. My friends Cecilia, Keith Herrera, Angella, and Jackie the cynic, Chris, K who supported me when I was going through one of the most difficult times of my life, my colleagues David, Colin and Niyi who made my life at work so easy I managed to squeeze in time to finish writting this thesis. My friends, Francesca who has always loved me unconditionally, Susan Atai and Susan Adong with whom I have grown up, Dorcas, Sonya

and Douglas. Friends in the school of computing David, Ganesh, Bayo, Selpi, Ralph, Zia, Gefeng, Rahman and Mohammed. My thanks to the janitors who made my life easier especially Jack. The support staff who always indulged my every whim - Ian B., Carol, Ann, Dianne, Maggie, Tommy, Virginia, Colin, Susan and Ian G.; the list is endless.

Finally, I owe a debt of gratitude to my family who put up with my being away for so long and still listened patiently to all my complaints about a place where they did not want me to be in the first place! I thank my parents, GDM as we fondly refer to our Dad and my Mother Ethel (who will always be Mommy to me) for the emotional and monetary support throughout this period. My sister Gloria who has always and will forever be my best friend, a friend who makes me feel blessed many times over. To the rest of my siblings Laetitia, Bridget, Angella and Matt-Carol who all always got concerned when something bothered me even when they did not fully understand it. My brother Ed who never fails to make me laugh, Fred for his support, encouragement and prayers and Jesse, who taught me to stand up for myself, encouraged me throughout this journey and gave me the greatest gift of all. May the Lord who brought all these Angels into my life, pour immeasurable blessings to all these and numerous others that I have not acknowledged here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

SmartHouse[1] technology comprises devices that help the elderly and people with disabilities to retain a level of independence within their homes which previously would have been impossible. Simple routine tasks such as opening windows or drawing curtains are extremely difficult to carry out and sometimes impossible for many individuals due to their impairments. Occupational therapists are responsible for recommending devices to assist in carrying out these tasks. Matching subsets of a person's disabilities with appropriate technology is normally not an easy task. This is because individuals have different levels and combinations of disability. Moreover, a SmartHouse device may be designed to assist in one or more types of disability. The situation has been compounded by the recent increase in the range of SmartHouse technologies making it even harder for therapists to keep abreast of the available SmartHouse devices. Consequently, there is a need for computing tools to assist in matching technology to the needs of the elderly and people with disabilities.

A number of problem-solving methodologies such as statistical inferencing mechanisms (Cox 2006) and rule-based systems (Hayes-Roth 1985) are in existence. Statistical inferencing mechanisms typically rely on the availability of large volumes of data to make inferences and to identify dependencies that are inherent in the data, in order to extract key features with which the cases are represented. Many of these systems typically ignore

[1] A house that includes technology designed to support the lives of the elderly or disabled people.

terms that appear with low frequencies, an aspect that would be detrimental to sparse domains such as the SmartHouse in which a term may appear once but still be important. Consequently, statistical means are of limited use to sparse domains since there is simply not enough data to support the underlying mechanisms on which the systems rely to extract features. Moreover, statistical approaches typically result in knowledge-poor representations, requiring the user to intervene in the retrieval step of the process another feature that makes statistical approaches unattractive.

Rule-based systems require that an expert write down all the possible rules to enable the system to make the proper inferences when solving a new problem. For practical systems, it is not feasible to expect the expert to think of all possible rules before hand. Moreover, as new knowledge is acquired, these rules may have to be revised and/or more rules created. This makes rule-based systems unscalable. Furthermore, in order for an expert to create rules that are effective in problem-solving, it is imperative that the expert thoroughly understands the domain. This can be quite difficult for weak theory domains such as the SmartHouse. In such domains, the reasons behind a solution's suitability are not always straightforward and it is thus difficult to replicate the thought process during problem solving, as a set of rules.

In practical systems, the large number of rules required to cover every possible situation may make problem solving unacceptably slow. This is because the system would have to test a large number of rules to identify those that apply to the new problem situation. Moreover, there is potential for inconsistency in the created rules.

Case-based reasoning is particularly suited to weak theory, or highly dynamic domains, as it enables the solving of problems without totally understanding the domain. Moreover, it is easier for the occupational therapist to remember and reuse solutions for past solved problems than to design a whole new solution for every new problem. This is the underlying idea in Case-based Reasoning (CBR) where problems are solved by remembering solutions to similar past problems and re-using them to solve new similar problems, just like humans do much of the time. New SmartHouse devices and novel combinations of disability can be incorporated into the solution during the adaptation stage of the CBR cycle. Thus for this project, CBR was chosen over the other problem-solving methodologies. In this research, SmartHouse past problem-solving textual reports were used to

obtain the knowledge for use in the CBR system.

Humans interpret text by using their notion of domain knowledge, the concepts[2] comprising that knowledge and the relationships between them. Indeed, when a reader who is familiar with President Clinton's cat *Socks* encounters the sentence "*These socks are warm and comfortable*", they would have to decide if the word *Socks* refers to a *cat* or a *piece of clothing*, in order to make the correct interpretation of the text. Thus, if a machine is to correctly interpret a piece of text, whether for purposes of case authoring, or interpretation of queries during problem-solving, it should have access to a conceptual model of the domain in question. The model will comprise important words and phrases in the domain and their relationships with each other. This model can be a template, or a basis for a template, on to which the texts or queries are mapped, in order to create cases or retrieve similar cases for solving new problems. Engaging domain experts to create a conceptual model can be costly not to mention the difficulty of maintaining the model as new knowledge is obtained. Moreover, errors and inconsistences are likely to occur as different experts may have different views of what they perceive to be important and should be contained in the model. Thus there is a need for automated tools that create, or help in creating, domain-specific conceptual models to assist in the tasks of case authoring and query interpretation. Documents describing important domain concepts are a good source of knowledge for creating such models. However, textual cases by their nature are usually heterogeneous as the same words are seldom used to describe the same situations. Queries are also likely to comprise words or phrases that have the same meaning as, but are different from, those containing case knowledge. Thus it is essential to have a language model upon which the conceptual model is built, and with which texts will be interpreted during case authoring and query processing. Creation of the language model involves developing a domain-specific synonym mapper so that although a different word is used in the conceptual model, synonyms of that word will be recognised as such enabling mapping of heterogeneous texts to a standard conceptual model.

This project uses SmartHouse past problem-solving reports to develop a language model. The language model is used to create a harmonised knowledge source from the SmartHouse reports, out of which key phrases are identified in order to create a conceptual

[2] *Concept* refers to a word or phrase with important meaning in the domain.

model. CBR makes use of a collection of solved problems called *cases*, which are stored in a *case-base*. The conceptual model is used to transform the extracted key phrases into structured cases and to interpret queries during problem-solving. Consequently, the project has developed a CBR tool to assist in matching SmartHouse technology to the needs of the elderly and people with disabilities to give them some degree of independence from carers. The techniques developed in this project are generic enough to be useful in other domains.

The lack of (suitable) data is a common problem encountered when developing problem solvers using knowledge contained in textual sources. Indeed, it is difficult to have data that cover all the possible scenarios initially, even when a reasonably big size of corpus is available. The problem is exacerbated in sparse domains. This gap in data has typically been compensated for by incorporating domain knowledge (Weber, Evanco, Waller & Verner 2004). Domain knowledge can be incorporated either by using it to guide system development, or by engaging an expert to provide the knowledge by interacting with the developed system for example in mixed-initiative approaches. In this project, domain knowledge is used to aid the key phrase extraction process. The concepts in the developed conceptual model are interpretable by a human thus allowing an expert to incorporate more knowledge as appropriate. The rest of this chapter gives an overview of the CBR problem-solving paradigm, a type of CBR called *textual CBR*, motivation behind the research and research objectives.

1.1 Case-based Reasoning

Case-Based Reasoning is a way of interpreting or solving a problem by finding a similar past case and adapting it to suit a new problem (Riesbeck & Schank 1989, Kolodner 1993b, Aamodt & Plaza 1994). When a new problem situation is encountered, the most similar cases to the current problem are retrieved from the case-base. The past cases and the current problem are compared to obtain the part(s) of the retrieved cases that can be re-used to solve the new problem. The proposed solution may further be revised using domain-specific knowledge to obtain the final solution to the new problem. In the SmartHouse domain, it is at this stage that new SmartHouse devices may be incorporated

in the solution. The parts of this new experience that are likely to be useful for future problem solving may be retained and stored in the case-base thus also making CBR a paradigm that enables *sustained learning* (Aamodt & Plaza 1994).

1.1.1 The CBR Cycle

Central to CBR is determining similar past cases that can be re-used in the new problem situation. Each case typically contains a description of the problem, plus a solution i.e., how the problem was addressed. In the SmartHouse domain, a case would capture the problems/impairments of a previous person and the SmartHouse devices that were installed in their home. The knowledge and reasoning process used by an expert to arrive at the solution is also implicit in the solution. The cycle, as shown in Figure 1.1, entails the following:

1. RETRIEVE the most similar case or cases from the case-base. Retrieval starts with a (partial) problem description and ends when a best matching previous case has been found.

2. REUSE the information and knowledge in that case to solve the problem. This involves comparing the past and current case in order to determine parts of the past case that can be used in the current case.

3. REVISE the proposed solution. This might involve evaluating the solution generated by reuse. If the solution is successful in solving the current problem, it is stored in the case-base to enable learning from the new experience. If the solution is not successful in solving the current problem then it is repaired using domain-specific knowledge. In the SmartHouse domain, this is when new devices may be incorporated into the solution.

4. RETAIN the parts of this experience likely to be useful for future problem solving in the case-base.

The retrieval stage of the CBR cycle relies on the system's ability to compare a new case with those in the case-base in order to determine the most similar case(s). Cases that are stored as free-form text cannot be effectively compared since such text is unstructured.

In this situation therefore, a structured representation of the unstructured text needs to be crafted in order to make effective case comparisons.

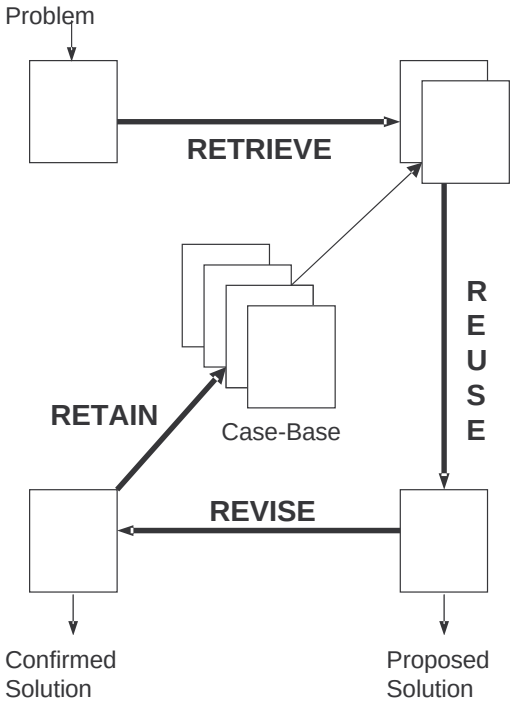


Figure 1.1: The CBR Cycle From Aamodt & Plaza (1994)

CBR has made significant headway in the fields of medical diagnosis (Schmidt & Gierl 2000), law (Rissland & Ashley 1987, Rissland & Daniels 1995, Aleven & Ashley 1997, Aleven 2003), and fraud detection (Wheeler & Aitken 2000). It has also been applied to the pharmaceutical domain (Craw & Rowe 2002) and used in knowledge modeling (Cañas, Leake & Maguitman 2001), knowledge management (Weber, Aha, Branting, Lucas & Fernandez 2000), and in the SmartHouse[3] domain (Wiratunga, Craw, Taylor & Davis 2004) among others.

1.1.2 The Knowledge Container Model

CBR relies on the availability of domain-specific knowledge in order to suggest useful solutions to problems. According to Richter (1995), not all knowledge is stored in the cases but rather there are four *knowledge containers* within a CBR system.

[3]Their case-base was built manually which is the main motivation behind this project.

Vocabulary This is essentially case representation which includes the attributes and possible values of these attributes; the latter are used to describe cases and the case structure.

Case knowledge This is the knowledge inherent in the case itself. It could be attribute-value knowledge regarding the problem and solution parts of the case. It is imperative that the problem be described in a way that enables matching with new problems. The solution part may include additional information pertaining to the path that led to the solution.

Retrieval Knowledge This knowledge defines how the most relevant case is determined during the retrieval stage of the CBR cycle and the means by which similarity between a new and old case is calculated.

Adaptation knowledge This is additional knowledge needed for the retrieved solution(s) to be fitted to the new problem situation.

Indexing and maintenance are the other knowledge containers that support the CBR system (Patterson, Anand, Dubitzky & Hughes 2000). All the containers are related and it is possible to move knowledge between them. The vocabulary, case knowledge and retrieval knowledge are the main foci of this project.

1.1.3 CBR Approaches

There are three main categories of CBR; Structural, Textual and Conversational Case-based reasoning.

Structural Case-based Reasoning This is CBR where cases have well-defined attributes and their corresponding values. The cases may be organised as flat files or structured in an object-oriented manner. Directory assistance is a good example domain as cases (e.g., people, companies) tend to have common features such as name, address, etc. Recommender systems where all products have the same features also fall under this category.

Textual Case-based Reasoning This is a kind of structural CBR where cases are created from textual sources. It results in cases that may be structured or not and an

index that will be used to retrieve relevant cases to solve or interpret a new problem. The SmartHouse, medical and law domains are good examples.

Conversational Case-based Reasoning Conversational CBR (CCBR) is a form of interactive CBR that uses a mixed-initiative dialog between the system and user during case retrieval. The cases in this type of CBR are attribute-value or question-answer lists that the user interacts with by either refining the problem, or selecting one of the presented solutions. Help desk and Recommender systems are good example domains.

1.2 Textual Case-based Reasoning

There have been several definitions for Textual Case-based Reasoning (TCBR). Lenz, Hübner & Kunze (1998b) define TCBR as CBR applied to cases described as textual documents as opposed to structured cases. This contradicts the view presented by Roth-Berghofer (2004) that TCBR involves creation of structured cases from semi-structured text. Roth-Berghofer's (2004) view does not take into account those CBR systems whose cases are problem-solving experiences that were stored as unstructured documents. Wilson & Bradshaw (2000) defined TCBR as "CBR that will make use of text to enable or to enhance its reasoning process". They analyse the knowledge-richness of a case representation and on the roles which the knowledge-rich and knowledge-poor parts of the case play in the reasoning process. As a result, TCBR in which the full-text document is used in reasoning is referred to as *strongly textual* and that where more knowledge-rich components are used in the major part of the reasoning process as *weakly textual* CBR. On the other hand, Lamontagne, Langlais & Lapalme (2003) used the term *strongly textual* to refer to those cases where the problem and the solution were both textual.

From these definitions, it can be said therefore, that TCBR involves the creation of cases from textual sources and reasoning with cases that may be unstructured (whole documents that are knowledge-poor), semi-structured (e.g. forms comprising a knowledge-poor textual part and some well-defined knowledge rich components), or structured (e.g. database records with well-defined attribute-value pairs). Therefore, the type of CBR that will be dealt with in this project can be described as *strongly textual TCBR* in the sense

that both the problem and solution parts are textual.

1.2.1 Case Types

Watson (1997) presents the case types: homogeneous versus heterogeneous, and episodic versus prototypical. *Homogeneous* cases all have the same features that are easy to elicit. An example is the real estate domain where it is not uncommon to have buildings with the same features. On the other hand, *heterogeneous* cases have different features but may share some. It is hard to elicit a full set of features and the system has to learn new case features throughout its lifetime. Patient cases in a hospital and the SmartHouse are good example domains because, people may share some features like age and sex but have different ailments or impairments.

Episodic cases are those cases which can be obtained from records of events. Equipment faults are example cases. *Prototypical* cases are designed by experts as examples of typical events and require knowledge elicitation. Typical symptoms of a disease are examples of prototypical cases.

The required case type has an impact on the techniques and knowledge engineering effort needed for case authoring. This project will create structured cases from semi-structured SmartHouse textual reports. The cases will be heterogeneous as only some disabilities are common among some people. Since there are many types of disability and the range of SmartHouse devices is wide, the system will also continuously learn new case features.

1.2.2 Case Authoring

Zaluski, Japkowicz & Matwin (2003) defined *case authoring* as the "acquisition of new experiences that are not represented in the case-base", while Weber et al. (2000) referred to case authoring as the "identifying, eliciting, representing, and indexing (of) cases". The first definition applies to this project in a general sense and the second one explicitly lists the processes that are going to be tackled.

Problem-solving in CBR is by analogy in which a new problem is solved by recognising its similarity to a specific known problem. It is therefore imperative that the chosen case representation allows for effective comparison. However, the problem-solving experiences

may not be in the same form as the desired case representation, since they may have been recorded for an entirely different purpose rather than to be used for case-based reasoning. It is for this reason that some case-based systems use original problem-solving experiences as cases, but others can only use cases that have been extracted from the original data.

The documents from which cases are authored may be unstructured, semi-structured or structured. The target case representation may also be unstructured, semi-structured or structured depending on the intended application of the CBR system. Even the type of structure, whether flat or hierarchical, also depends on the intended usage of the system and the domain in question. For example, early document retrieval systems like FAQ Finder (Hammond, Burke, Martin & Lytinen 1995) and the hotline scenario presented by Lenz, Hübner & Kunze (1998a) in their CBR-Answers project used whole unstructured documents as cases as it was up to the user to determine the meaning of text in the documents and not the system. On the other hand, in domains such as SmartHouse where cases are not very easy to acquire, it is desirable to have a structure that allows for retrieval of components of multiple cases in order to match new problem descriptors and one that enables case comparison at different levels of specificity to enhance retrieval effectiveness. In this situation, a hierarchical case structure may be beneficial. Watson & Perera (1998) demonstrated that cases that were represented as a hierarchy of smaller cases lead to more efficient retrieval than those using a simple flat case representation. Wiratunga, Craw, Taylor & Davis (2004) also realised better retrieval efficiency when they used generalised concepts to add levels of abstraction to cases constructed from textual SmartHouse reports. Whatever the situation, the knowledge engineering effort required to author cases and the means employed, depend on the structure in which past experiences were recorded, available sources of additional knowledge and the target case representation structure. In order to contextualise the case authoring effort in this project, it is important to have an insight into the nature of the SmartHouse problem-solving reports that were used to obtain the cases.

1.3 SmartHouse Reports

SmartHouse problem-solving experiences are recorded as textual reports. Each report captures the problems/impairments of the person with disabilities and the SmartHouse devices that were installed in their home to assist them in carrying out different tasks. Figure 1.2 shows an excerpt from one report. First, it briefly summarises the person's problems. It may mention a disease like *Alzheimer's Disease* that results in disabilities, or explicitly state the person's disabilities e.g., *mobility problem* or impairments e.g., *hearing impairment*. Disabilities and impairments are typically referred to as types of *problems* e.g., *hearing problem*, types of *difficulties* e.g., *hearing difficulty*, or kinds of *impairments* e.g., *hearing impairment*.

Christiansen & Baum (1997) describe *impairments* as "*the loss and/or abnormality of mental, emotional, physiological, or anatomical structure or function.*" The authors further explain that the term includes all losses and abnormalities including pain as a limiting experience. They further report *disability* as having been defined as "*the inability to perform or as a limitation in performing socially defined activities and roles expected of individuals within a social and physical environment as a result of internal or external factors and their interplay*". For simplicity, and since impairments and disabilities are both equally important in SmartHouse device recommendation, we do not make a distinction between disabilities and impairments. Thus both disabilities and impairments will be referred to as *disabilities*. Diseases that result in disabling conditions will be referred to as *diseases*. The terms describing diseases will be *disease-terms* and those describing disabilities will be *disability-terms*. Typically, the causes of a person's disabilities i.e., disability or disease terms, are mentioned in the summary and may not be repeated in the problem description text where symptoms and problem-areas are elaborated. Sometimes both the disability-term and disease are mentioned. For example in Figure 1.2, the cause of the person's difficulties (*Alzheimer's Disease*) appears in the summary and it is not repeated in the rest of the report.

The following sections of the SmartHouse reports record the different ways in which the person's disabilities manifest themselves. Each section describes a particular area of difficulty or risk. The excerpt shows a description of the *wandering* problem the person

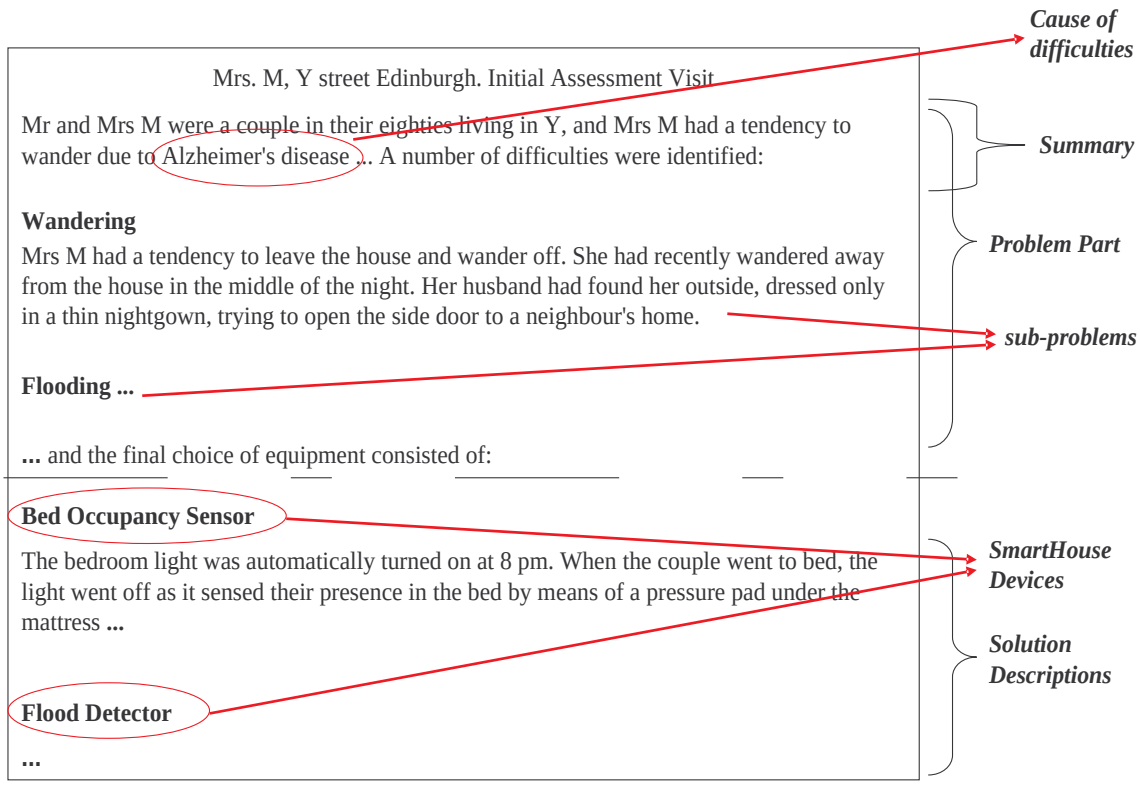


Figure 1.2: SmartHouse Report Excerpt

had. Every problem-area is given a summary heading, but they do not always accurately describe the content. *Telephone operation* may be used as a heading of a section describing a person's inability to use their telephone because they had difficulties *hearing* the caller. In another report, the same heading could be used for a description of a person's difficulty in using their telephone because their *mobility* problems prevented them from reaching the phone in time to answer the call. The summary and the problem-description sections make up the problem part of a SmartHouse case. We shall refer to each problem-description section as a *sub-problem* since it is only a part of the whole problem description.

Lastly, the report mentions the solution package which lists the SmartHouse devices or recommended activities, each with a description of how they help. Although it is not always obvious from the text, every sub-problem has a corresponding list of solutions. This is not always a 1-1 mapping because a particular sub-problem can generate the need for more than one SmartHouse device. However, it is possible to map each sub-problem to its corresponding solution components. The example in Figure 1.2 shows that a *Bed occupancy sensor* was installed to help the person with their *wandering*. A report records

one or more sub-problems with accompanying solutions. Thus the summary and all the different problem-areas make up the problem part of a SmartHouse case. The different device descriptions comprise the solution part.

1.4 Motivation behind the Research and Objectives

CBR is meant to offer a cost-effective solution to the so-called *knowledge acquisition bottleneck* since solutions do not have to be designed from 'first principles' in every new problem situation. However, the knowledge engineering effort required to extract cases from textual sources can lessen the advantages gained by developing a CBR system instead of using other problem-solving methodologies. This is especially so if the text is unstructured or semi-structured. In an earlier study by Wiratunga, Craw, Taylor & Davis (2004), a CBR prototype system for the SmartHouse domain was developed but cases were *manually* authored from textual SmartHouse reports. This was a demanding task involving first, the identification of domain-specific concepts from semi-structured text then assembling the concepts into hierarchically structured cases. Thus in this project, the process of creating cases from the textual reports is automated as much as possible.

Techniques in machine learning, natural language processing and information retrieval (IR) have in the past been combined in efforts to automatically identify features for indexing cases. IR-based methods employ shallow statistical inferences that typically result in knowledge-poor representations. This results in poor retrieval effectiveness as the representation determines the cases that will be retrieved. Current research in Textual CBR aims to create more knowledge-rich case representations to enable effective retrieval and reasoning (Weber, Ashley & Brüninghaus 2005). Techniques in natural language processing have been explored, for purposes of obtaining case representations from text but their heavy reliance on grammar makes them unattractive in domains where problem-solving experiences were not recorded following strict grammatical structure. Machine learning approaches typically rely on training data that has been tagged by an expert, which may have the consequence of making the cost of developing such systems prohibitive. This research has developed a tool that automatically creates knowledge-rich case representations from textual sources without the need to access large volumes of tagged data.

It is important that the cases capture the knowledge in the original reports if they are to be useful for problem-solving. Hence it is crucial for the case representation to capture important domain concepts and the relationships between them. In the SmartHouse domain, concepts are groupings of descriptors of people's disabilities and their manifestations. Some of this information may be in the summary, but the bulk of it is embedded in the sub-problem descriptions of each SmartHouse report.

Therefore, the first step in authoring cases from these texts is to identify and extract the knowledge embedded in the text which in turn, is used to determine case and similarity knowledge. It has been observed that humans do not interpret text at word-level, but do so at a much higher level of abstraction where concepts are manipulated (Gabrilovich & Markovitch 2007). For example, an occupational therapist who reads about a *wheelchair user* immediately thinks about the person's *mobility*. Indeed, it is not possible for a human to read and understand text about a domain without (domain) knowledge to which they can relate what they are reading. If the human relates the words to a domain that is different from one to which the text is related, they will either misunderstand the meaning of the text or not understand the text at all. Thus, it is unreasonable to expect a machine to manipulate a piece of domain-specific text in a meaningful way, without access to a knowledge model that pertains to that domain. The humans' ability to organise information into concepts, in order to extract meaning that is beyond the words they read, is what the case authoring tool in this project attempts to mimic. A domain-specific conceptual structure is built to enable the case authoring task to interpret text in the SmartHouse reports, in a meaningful way.

This project has created SmartCAT-T[4], a tool that assists case authoring from texts. A combination of techniques from various spheres of research have been used to enable the system to learn to identify key phrases that depict domain-specific concepts in a given SmartHouse report, and to group the key phrases under the appropriate concepts. For example, a phrase like `wheelchair` will be identified as belonging to the `mobility` concept or one like `wandering` as belonging to the `dementia` concept.

The SmartCAT-T case authoring tool makes use of background domain knowledge to obtain a set of anchors and to use the anchors to highlight key phrases in the reports.

[4]**Smart C**ase **A**uthoring **T**ool for **T**ree structure cases

SmartCAT-T uses the key phrases to learn a hierarchically structured conceptual model onto which reports are mapped to create the corresponding structured cases. The solution for each case is attached to the most-specific concept node(s) of that case.

SmartHouse device recommendation is based on people's disabilities, tasks which people find difficulty carrying out due to their impairments, and risks people face as a result of their impairments, or the diseases from which they suffer. Indeed, when an occupational therapist needs to recommend SmartHouse solutions for a new person, she will be more interested in the person's disabilities and areas in which the disabilities manifest themselves, than in the medical condition that caused the disabilities. This approach to problem-solving is not unique to the SmartHouse domain. In the real world, problem-solving begins with a list of problem descriptors for which a list of solutions is sought. Therefore, this project focuses on structuring the data pertaining to domain problems, for it is these that will be useful for problem-solving. Consequently, the development of the conceptual model for the SmartHouse domain focuses on structuring the problem parts of the SmartHouse reports. Each problem-part of a SmartHouse report is regarded as a document. The conceptual structure is based on people's disabilities and impairments. Furthermore, the concepts in the conceptual model are interpretable, allowing for expert refinement of knowledge.

The conceptual model is editable, a feature that allows the system to learn from an expert. The system uses the conceptual model to create rules regarding the data. Thus the inferences the system makes on new data improve as more knowledge is acquired. The idea is that the demand on the expert will gradually reduce as system learning progresses.

The project had the following research questions:

- Is it possible to automate the knowledge elicitation task and still obtain cases that are useful for problem-solving?
- If it is not possible to create knowledge-rich cases using a fully automated system, is it possible to semi-automate the knowledge elicitation task whilst ensuring that the amount of required manual intervention does not inhibit system scalability?
- If the knowledge elicitation task is automated or semi-automated, will the resulting identified domain concepts allow for human knowledge refinement and system

learning?

The following objectives were then obtained from the research questions:

- to design a case structure onto which unstructured textual reports will be mapped;
- to design and implement a semi-automated tool to map unstructured text to the case structure obtained above;
- to develop a keyword hierarchy to enable structuring of cases with different levels of specificity; and
- to design and implement a case-based tool to assist in the design of SmartHouse solutions.

However, in the SmartHouse domain, individual words have often been found insufficient to capture the meaning of important expressions. For example the phrase *intercom operation* does not mean *an intercom* or *an operation* but rather *the way to operate an intercom*. Phrases containing two or more words have been found to be more informative. Consequently, the *keyword hierarchy* mentioned in the objectives has been revised to *concept hierarchy* where a concept is a group of words and phrases. Furthermore, in order to deal better with the problems of polysemy and synonymy which are inherent in the English language, the vocabulary is restricted by harmonising the text.

1.5 Thesis Overview

This thesis will develop a case authoring tool to enable the extraction of knowledge from semi-structured text. Hierarchically structured cases allow comparison at different levels of abstraction. Thus the extracted knowledge will be assembled into structured cases. The task of creating structured cases from the documents is divided into the following steps:

1. Harmonising the text in order to obtain a uniform vocabulary;
2. Representing the documents with only those terms that actually describe the problem (key phrases);

3. Using the representative terms to create a hierarchically structured conceptual model that reflects important features in the domain and their relationships; and

4. Mapping each document onto concepts in the conceptual model in order to create structured cases.

Each of these steps shall be explained in detail in the next chapters. The SmartCAT-T system architecture is illustrated in Figure 1.3. Documents in the collection are used to create synonym mappings for the domain by making use of WordNet. A text harmonisation module transforms the documents into harmonised texts by utilising the synonym mappings and Google. The harmonisation module is also used to meaningfully interpret queries during problem-solving. Background knowledge is used to discover disability and disease terms pertaining to each harmonised document. A documents disability and disease terms, Latent Semantic Indexing (LSI), and background knowledge are all utilised in identifying a document's key problem descriptors that are not disability or disease terms. All the discovered key problem descriptors together with the document's disability and disease terms are used to represent the document. Formal Concept Analysis is applied to the documents' representatives to obtain a conceptual model with which structured cases are created.

A survey of the literature pertinent to this research is carried out in Chapter 2. The survey examines techniques that are related to the different modules that comprise this project's case authoring goals and identifies gaps in current research in order to justify the work carried out in this project.

In Chapter 3, a description of the harmonisation procedure is given. The SmartHouse reports are harmonised in order to create a uniform vocabulary and hence overcome the problems of polysemy and synonymy during the later stages of knowledge extraction. Central to the harmonisation is an unsupervised approach to Word Sense Disambiguation. WordNet is used to assign each polysemous word with a sense that is applicable in the domain.

Chapter 4 presents the extraction of key phrases which comprise disease terms, disability-terms and other key problem-descriptors. Disease terms are identified by making use of a list of known diseases and disability terms are obtained by making use of the commonly

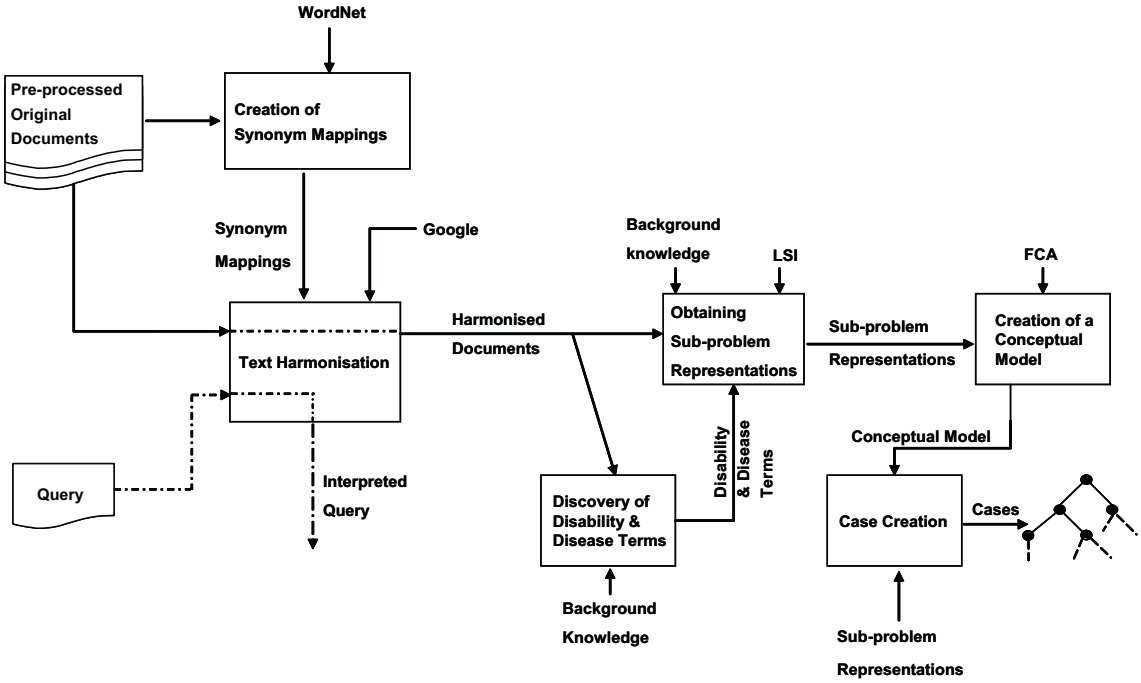


Figure 1.3: System Architecture

occurring complaint-description terms *difficulty*, *difficulties*, *impaired*, *problem*, and *impairment*. The system makes use of background knowledge to identify other terms that are not disability or disease terms, but are key problem-descriptors. The key phrases are used as representations for the corresponding sub-problems.

Chapter 5 examines how Formal Concept Analysis can be employed to create a conceptual model. The sub-problems in the text and their representative (key) phrases are used to create a formal context onto which Formal Concept Analysis is applied in order to obtain formal concepts and a conceptual model. Mappings of sub-problems to their respective solution descriptions is also carried out. Phrase overlaps between sub-problem and solution texts are used to identify solution description texts for each sub-problem. These mappings are later used to match the appropriate solution to each authored problem-part in order to complete the case creations.

In Chapter 6, a case authoring tool SmartCAT[5] is presented. SmartCAT uses the sub-problem representations to obtain a conceptual model. It then uses the model to create cases and to organise the cases into a structure based on relationships between concepts

[5]**Smart C**ase **A**uthoring **T**ool

in the conceptual model. However, the cases so obtained are flat structured which do not allow for comparison at different levels of problem abstraction.

Chapter 7 presents a description of the SmartCAT-T case authoring tool. SmartCAT-T changes the nature of the key phrase representations and consequently reduces dependencies between the various objects in the formal context. It then creates more redundancy in the resulting context in order to overcome multiple inheritance in the subsequent conceptual model. SmartCAT-T uses the tree structured conceptual model to create hierarchically structured cases. The created cases can be compared at various levels of abstraction and are not prone to the effects of multiple inheritance as do the cases created using SmartCAT.

Chapter 8 presents an evaluation of the SmartCAT-T system. The quality of the case content is measured against original documents. An evaluation is also carried out to estimate the contribution of the various modules that comprise SmartCAT-T to the effectiveness of the tool. The effect of employing the case structure in retrieval is also evaluated against employing a flat structure. SmartCAT-T is also benchmarked against a high-standard Information Retrieval tool.

Chapter 9 investigates the extent to which the techniques developed in this research can be applied to other domains. This is achieved by subjecting the techniques to the domain of air and marine safety investigation and carrying out a short evaluation to determine the usefulness of the cases created by the SmartCAT-T approach, but using documents in this domain.

A summary of the the contributions of this research and possible directions for future work are presented in Chapter 10.

1.6 Summary

This thesis will develop a tool that assists case authoring from semi-structured textual reports in the SmartHouse domain. Consequently a CBR tool will be developed to assist in matching the needs of the elderly and people with disabilities with appropriate SmartHouse technology. SmartHouse problem-solving experiences are used to create cases in order to populate the case-base. However, the reports were recorded as free-form textual semi-

structured reports. This kind of text does not allow for effective case comparison in order to retrieve the most appropriate cases in a new problem situation. Therefore, a structured representation of the reports is required. This representation can be acquired by extracting knowledge from the textual reports and presenting it in a structured form.

Developing a CBR system when the knowledge is embedded in textual sources typically presents the so-called *case acquisition bottleneck*. Access to domain-experts and time are important aspects that need to be addressed (Zaluski et al. 2003). Case-based reasoners that work with textual sources have addressed this problem in different ways. The means employed depend on the nature of the text and (intended) application of the system. The term *case authoring* is in this project used to refer to the *identifying*, *eliciting*, *representing*, and *indexing of cases* (Weber et al. 2000).

SmartHouse device recommendation is a form of decision support like that used in recommender systems and industrial trouble shooting. Case authoring involves steps such as the identification of a feature vocabulary and the possible values the features can have (feature-value pairs), feature organisation, case indexing and determination of relevant solutions for each problem structure. Organising feature vocabularies make the TCBR systems feasible and effective because the organisations can be used for traversal in search of new solutions and comparison at different levels of problem specificity.

The next chapter presents a survey of the pertinent literature that has been carried out. It provides an evaluation of related work in the field and justifies the need for this research by identifying the gaps it will fill. It illustrates and gives examples of case representations where knowledge was originally available as text. The survey demonstrates that the methods used to elicit knowledge from text depend on the nature of text(whether it is structured, semi-structured, or unstructured), availability of additional knowledge, and the desired case structure. Finally, it makes the case for the chosen methodology in order to obtain the targeted case representation structure.

Chapter 2

Literature Survey

This chapter presents a survey of work that is related to this project and identifies the gaps it will fill in order to justify the research. Textual Case-based Reasoning (TCBR) research can be grouped into 4 main categories: (1) assessing similarity between textual cases, (2) mapping of texts to structured representations, (3) adaptation of textual cases and (4), automatic generation of case representations from textual sources (Weber et al. 2005). This project automatically creates cases from texts and thus falls in the last category. In particular, case authoring involves mining text for domain-specific concepts and assembling the concepts into structured representations of the original (textual) problem-solution pairs. Thus, this survey will focus on knowledge elicitation techniques and automatically creating case representations from textual sources.

Generally, techniques employed for knowledge elicitation depend on the structure of the original text and the targeted case representation. Approaches ranging from simple techniques based on standard Information Retrieval (IR), to deep natural language processing (NLP) have been employed by many researchers to create unstructured, semi-structured and structured cases from textual sources. IR-based techniques usually require large amounts of data, are easy to deploy but afford knowledge-poor representations. On the other hand, more sophisticated methods are required to obtain knowledge-rich representations.

This chapter will present, case representations provided by different TCBR systems and the techniques used to elicit knowledge to construct these representations. The survey

will concentrate on those CBR systems that have made use of textual sources to obtain case representations or indexing knowledge. It will also include knowledge elicitation approaches from textual sources, that are potentially useful for case authoring.

One of the factors affecting the effectiveness of approaches that aim at automatically creating case representations from textual sources, is the systems' ability to deal with synonymy and polysemy i.e., Word Sense Disambiguation (WSD). Feature extraction techniques such as Latent Semantic Indexing (LSI) and Propositional Semantic Indexing (PSI) that address this issue, result in features without an underlying structure to enable case comparison at various levels of abstraction. Information extraction systems address the synonymy/polysemy problem by using a standard template to which all information should conform while manual/semi-automatic approaches typically require that the authors use a standard vocabulary or that, a collection of tagged documents be used to learn key features for case representation. The use of a standard template requires all the target information to be known a priori. Consequently, some important information may be ignored by the extraction process because it is not represented in the template, or some slots could still be filled even when their corresponding information becomes obsolete. Tagging the documents for learning is not always practical and is not feasible for big document collections and using a standard vocabulary can be restrictive in both case authoring and retrieval. Thus, it is important to incorporate word sense disambiguation into the techniques for creating case representations. Consequently, this chapter shall also present some techniques on word sense disambiguation.

Figure 2.1 organises the CBR systems that will be discussed. Authors of the systems are indicated in instances where the system names are not known. References are shown in addition to the system name in instances where a system appears more than once in the Figure.

2.1 Case Representations in Textual CBR

In case-based reasoning, new problems are solved by re-using or adapting solutions for previously solved problems. However, sometimes the original problem-solving experiences are not available in a way that supports comparison of new problems with the solved ones.

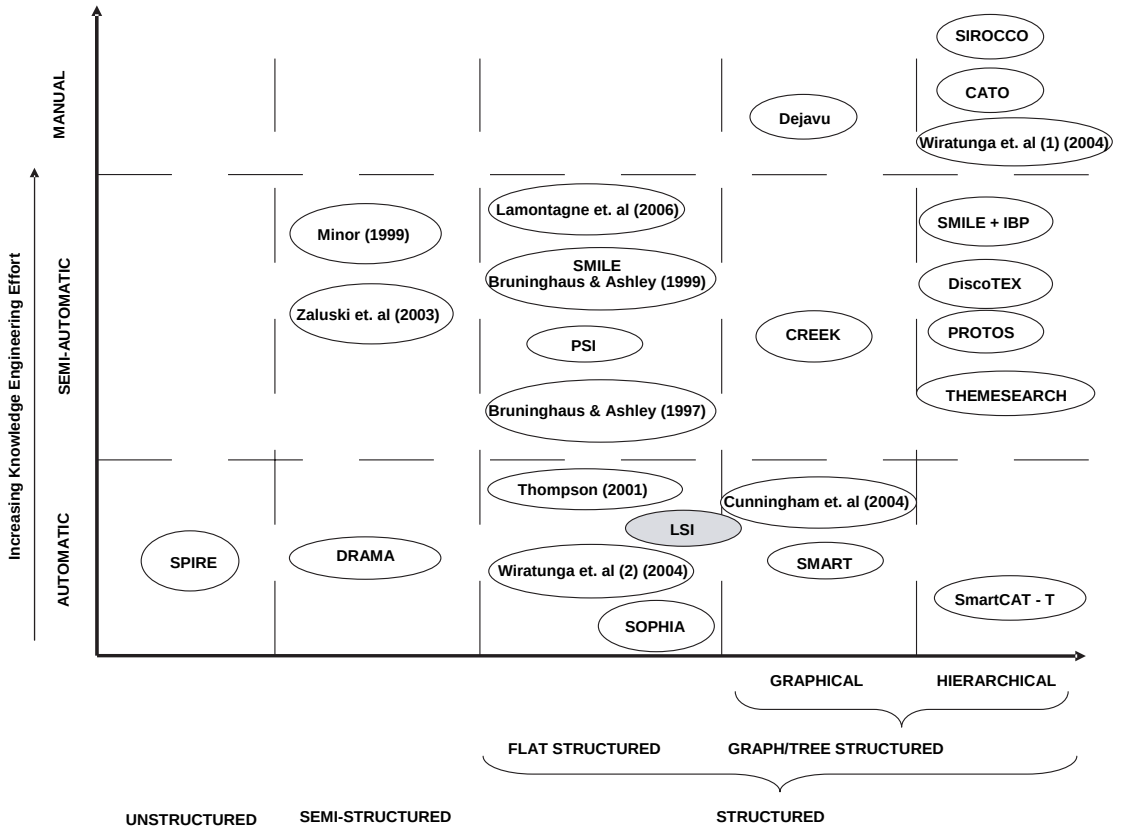


Figure 2.1: Textual CBR Systems: Case Authoring Technique vs. Case Representation

Thus it is often necessary to present the original artefacts in a way that provides a means for comparison and consequently, reasoning in new problem scenarios. Three major types of case representations have been used in CBR:

- Feature vector representations - a case is represented as a vector of feature-value pairs. Case comparison is carried out by matching features in the new and target case.
- Structured representations - a case is represented as a graph or tree-structure. Case comparison is by finding the most common sub-graph between a new case and target case or by pair-wise comparison of the nodes in the two cases. In the case of hierarchical structures, comparison at different levels of case abstraction is also possible.
- Textual representations - the cases are represented as the original documents or parts of documents. However, a weak structure is imposed on the cases to facilitate retrieval.

In complex domains such as planning, software engineering and our SmartHouse, the vector representation is not expressive enough for real applications and the structured representations are preferred. The techniques discussed in this survey have been broadly classified as those that would result in either *knowledge-poor* or *knowledge-rich* case representations.

2.2 Approaches Resulting in Knowledge-poor Representations

Approaches that employ techniques from standard Information Retrieval (IR) typically result in knowledge-poor representations. These approaches are adequate for retrieval oriented systems where the search for documents, information in documents, text or data is carried out, but the interpretation and understanding of the retrieved documents is the responsibility of the user. A document is treated as a bag-of-words (BOW) whereby text is broken up into individual tokens and word order is not taken into account. Terms which discriminate documents are then determined by ranking the terms using a frequency-based technique. The most popular ranking mechanisms are based on the Vector Space

Model (Salton & McGill 1986) while other techniques like the Chi-square statistic may also be employed.

2.2.1 The Vector Space Model

Documents are represented by the words they contain. Terms will have different levels of importance in a document and over the entire document collection. Consequently, each term is associated with a weight that reflects these importances (Salton & Buckley 1988) hence the emergence of a weighting scheme. A weighting scheme will typically have three components; a local weight, a global weight and a normalisation factor. Thus, the weight w_{ij} of a term i in document j will typically be given by:

$$w_{ij} = l_{ij} \times g_i \times n_j \tag{2.1}$$

where l_{ij} is the term's local weight in the document j, g_i is its global weight and n_j is the normalisation factor for the jth document in the collection (Kokiopoulou & Saad 2004).

A vocabulary of the terms in the document collection is created. Each document j, is then represented as a vector based on the vocabulary where the non-zero elements are the document's term-frequencies or *TF(i,j)*. However, in order to reduce on computational costs, stopwords such as *the*, *of* and *to* are removed from the text as they are so common and therefore, not considered to convey useful information. Words are also reduced to their stems or lemma. Stemming refers to stripping a word of its suffix and lemmatisation is assigning a word to its grammatical base word form. This enables the different forms of a word to be regarded and thus treated the same. A term is likely to be important in a document if it is frequent in the document but infrequent in the other documents in the corpus. Putting this into consideration gives rare terms a higher chance of getting identified as relevant to the documents in which they appear. This is done by using the Inverse Document Frequency (IDF) of a term i and is given by:

$$IDF(i) = -log(\frac{n}{N}) \tag{2.2}$$

Where N is the number of documents in the corpus and n is the number of documents that

contain term *i*. When the *IDF* value is multiplied with the term frequency *TF(i,j)*, the Term Frequency Inverse Document Frequency (*TFIDF*) weighting of a term i in document j is obtained:

$$TFIDF(i,j) = TF(i,j) \times IDF(i) \tag{2.3}$$

Thus, according to Equation 2.1, the TFIDF weighting comprises a local weighting (TF) and a global weighting (IDF) but does not incorporate a normalisation factor. A term will be identified as important if it occurs frequently in some document but infrequently in other documents. Although TFIDF is used in the Vector Space Model as a term selection metric, it or its variants can be used in isolation to rank terms. However, none of these metrics incorporate semantics and thus it is possible to rank as low, infrequently occurring but important terms.

2.2.2 Apache Lucene

Apache Lucene[1] is a tool that is widely used in the Information Retrieval (IR) community. Lucene makes use of a combination of the Vector Space Model (Salton & McGill 1986) and the Boolean model (Norreault, Koll & McGill 1977, Salton, Fox & Wu 1983) to determine a document's relevance to a given query. In response to a query q Lucene assigns a score to each document d according to equation 2.4.

$$Score(d,q) = \sum_{t \in q} (tf(t,d).idf(t^2).boost(t,q).boost(t,d).norm(d)) \tag{2.4}$$

where

- $tf(t,d)$ is the term frequency factor for the term t in the document d.
- $idf(t)$ is the inverse document frequency of the term.
- $boost(t.field,q)$ is the boost of the field of t in query q.
- $boost(t.field,d)$ is the boost of the field of t in d, as set during indexing.

[1] http://lucene.apache.org

- $norm(d)$ is the normalisation value which depends on the number of terms in the document.

Thus, Lucene is quite sophisticated in its own right as it makes use of a combination of parameters to obtain the final ranking for a document. However, it treats each document as a bag-of-words and does not take word order or semantics into account.

2.2.3 The Chi-square Statistic

Another common approach to identifying key terms is to make use of the chi-square statistic. The one presented by Matsuo & Ishizuka (2004) measures the degree of association between all terms and the most frequent terms in a document. The frequent terms are used as anchor terms to extract keywords from the rest of the text. The probability of occurrence of each anchor term, normalised so that the sum is 1, is obtained. This is the relative frequency or unconditional probability of the anchor term. A term w that appears independently from a set of anchor terms, has its distribution of co-occurrence with the anchor terms similar to their unconditional probabilities. Conversely, if this term has a semantic relation with a particular set of anchor terms $g \in G$, co-occurrence of the term w and g will be greater than expected; biasing the distribution. A term that shows co-occurrence with a particular set of frequent terms may have an important meaning in a document. Therefore, the degree of bias of co-occurrence can be used as an indicator of term importance. This bias is a measure of the chi-square value of the term w and is given by:

$$\chi^2(w) = \sum_{g \in G} \frac{(freq(w,g) - n_w p_g)^2}{n_w p_g} \tag{2.5}$$

Where p_g is the unconditional probability of an anchor term $g \in G$, n_w is the total number of co-occurrences of term w with anchor terms in G, and $freq(w,g)$ is the number of sentences in which w occurs with g. Thus, $freq(w,g)$ represents the observed frequency, and $n_w p_g$ gives the expected frequency of term w.

However, to take account of the fact that a term that appears in a long sentence is more likely to co-occur with an anchor term than those that appear in short sentences, the terms n_w and p_g are revised: p_g becomes the sum of all terms in sentences in which g appears divided by the total number of terms in the document; and n_w becomes the total

number of terms in sentences in which w appears. This approach does not make use of a corpus which is a desirable feature for domains like the SmartHouse where the document collection is small. However, it favours large documents where high term frequency leads to meaningful values of the chi-square.

The above techniques do not consider semantics and word order when ranking terms or documents. Hence systems based on these techniques are mainly retrieval-oriented where the task is to identify those documents or texts that are relevant to a user's need. Typically, the derived case representations are knowledge-poor and systems make no attempt to understand case content. In many situations, original documents or their summaries are considered as cases. The developer thus incurs no case authoring cost and only an indexing strategy needs to be determined. However, the user needs to have a source of domain-specific knowledge for case adaptation. Such systems are not very effective in domains like SmartHouse and law where it is important to capture context.

2.2.4 IR-based Approaches in Textual CBR

SPIRE (Daniels & Rissland 1997) is an information retrieval system that was applied to the domain of personal bankruptcy. During problem-solving, SPIRE makes use of two case-bases to automatically generate queries which are used by a retrieval engine to retrieve whole text documents and to highlight those passages that are likely to contain useful knowledge for problem-solving. One case-base contains solved past problems which are represented as feature case-frames. Each case feature then has a case-base of actual text excerpts containing information pertaining to the feature. A simple indexing strategy is employed where the names of the features are used for indexing.

The Chi-square technique introduced by Matsuo & Ishizuka (2004) was applied to documents in the SmartHouse domain in order to identify representative features for the cases. The technique identified some key words that the expert had also deemed to be key but, it failed to pick out important low-frequency words and was worse when two or more word phrases were used as these were even less common than words. However, it performed better than the TFIDF technique as the Chi-square technique does not rely on the availability of a corpus whereas TFIDF tends to degrade with small document collections.

2.3 Approaches that Result in Knowledge-rich Representations

The approaches used to obtain knowledge-rich representations go beyond simple key word identification, by identifying features or concepts that discriminate documents or parts of documents. Identifying domain-specific concepts from text is a challenging task in which problems associated with natural language like word sense disambiguation, word boundary detection, syntactic and semantic ambiguity all need to be addressed. Techniques exist that have been used to elicit knowledge from free-form text for CBR and for purposes other than CBR. They range from manual means to sophisticated approaches exploiting Natural Language Processing (NLP) techniques. These techniques have, in this survey, been classified as either *manual* or *automatic/semi-automatic*. These approaches will be discussed in the next 2 sub-sections.

2.3.1 Manual Approaches

CBR systems that require a lot of manual intervention in the various steps of creating the case base do not scale up due to the effort required to elicit and organise knowledge from the textual sources in order to create the case representations. However, for purposes of completion, a few examples will be presented in this survey.

CATO (Ashley & Aleven 1996) is a case argument tutorial program for law students. CATO's cases are represented as textual summaries of the facts and decisions and a set of indexing factors. In CATO's factor hierarchy which was manually developed (Ashley & Aleven 1997), factors are factual strengths and weaknesses described in text. The hierarchy embodies information relating factors to those legal issues for which they are relevant. Figure 2.2 shows a part of this hierarchy. The base-level specific factors are linked through more abstract factors to the high-level legal issues at the top. The links are weighted according to the level of support they provide in regard to the conclusion at the next higher level and the weighting is shown by the thickness of the lines.

SIROCCO (**S**ystem for **I**ntelligent **R**etrieval of **O**perationalized **C**ases and **CO**des) (McLaren & Ashley 1999) is a CBR retrieval and analysis system applied to the domain of engineering ethics. It is intended to help engineers and students analyse new cases by retrieving

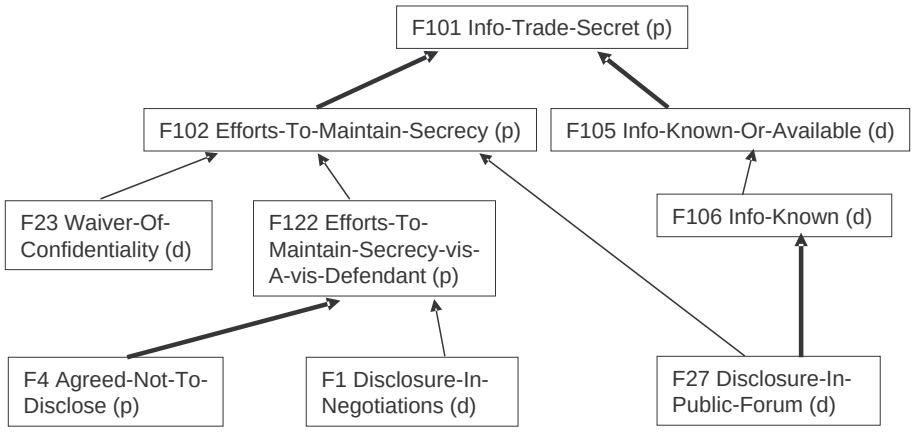


Figure 2.2: CATO's Factor Hierarchy

relevant past engineering ethics cases. Case authoring is manual where a case acquisition tool is used to transcribed cases into the Ethics Transcription Language (ETL). The language has a standard vocabulary which ensures that different people describe the same cases in a consistent manner. Cases are described as ordered lists of facts which in turn depict actions or events. SIROCCO stores all primitives in a *Fact Hierarchy* of the most important actions and events that typically occur in the engineering ethics domain. Like CATO, cases in SIROCCO can be retrieved and compared at different levels of the hierarchy.

Wiratunga, Craw, Taylor & Davis (2004) developed a CBR prototype system manually from a set of textual SmartHouse reports. The identification and mapping of concepts from semi-structured text to a structured representation was manually carried out by the domain expert. The prototype system is referred to as *Wiratunga et. al (1) 2004* in Figure 2.1.

2.3.2 Automatic/semi-automatic Approaches

Current research which focuses on automatically or semi-automatically creating knowledge-rich case representations can be classified into 2 broad categories; the first category extracts key phrases or predictive features for representing the textual cases, while the second category typically employs information extraction systems to obtain structured representations of the textual sources. These 2 types of approaches shall now be discussed.

Key Phrase and Feature Extraction

Keywords alone do not provide enough information as a word may be important in some contexts and not in others. Key phrase and feature identification address this problem to some extent as important word-groups are considered and sometimes semantics are also captured. An example of a key phrase extractor is KEA, based on an algorithm by Eibe, Paynter, Witten, Gutwin & Nevill-Manning (1999). KEA learns key phrases from a set of training documents by means of a naïve Bayes learning scheme. It employs the TFIDF technique to rank phrases in order to choose the ones that best represent the document at hand. However, this algorithm excludes from the computation phrases that appear only once in the document, a feature that may be undesirable when using small document collections. Key phrase extractors have been employed to aid CBR tasks. For example, Lamontagne, Bentebibel, Miry & Despres (2006) made use of KEA to learn key phrases from a set of reports in the domain of marine and air safety. They then used ANNIE, an information extraction tool of the GATE NLP platform (H. & Tablan 2002) to identify factual information that could be converted into higher categories e.g., *Quebec* (factual information) ==> *LOCATION* (higher category). However, the sentences used in the training set had to be manually tagged which is a limiting factor for their approach. Thus this approach is classified as semi-automatic in Figure 2.1 because of the manual effort required to tag documents for training the key phrase extractor.

In a bid to automate the indexing of cases in the law domain, Brüninghaus & Ashley (1997) treated the task of assigning CATO's factors (introduced in sub-section 2.3.1) to previously unseen legal opinions as a learning problem. In order to represent the texts as feature/value vectors, a bag-of-words approach was used where term weighting was done using the TFIDF metric. A set of labeled full-text legal opinions was used to train a naïve Bayes classifier to assign factors to cases. The bag-of-words approach was insufficient for enabling the classifier to generalise from examples to abstract fact patterns that depict CATO's factors. This was mainly due to the fact that the vectors used had many terms that were not related to the classes (the factors); a consequence of not incorporating domain knowledge.

In further work, Brüninghaus & Ashley (1999) used text summaries (referred to as

squibs) instead of full text opinions in their SMILE[2] system (Brüninghaus & Ashley 1999). Furthermore, the factors were now assigned to sentences in the squibs instead of whole documents and a domain-specific thesaurus was used to incorporate domain knowledge in the learning task. The ID3 decision tree learning algorithm (Quinlan 1993) was used to learn to classify squib sentences with their corresponding factors. Each sentence was represented by its non-stopwords. The problem of synonymy was addressed by using a single attribute to represent words that belonged to the same synonym set, as provided by the thesaurus. The results were better although still not satisfactory due to the use of single words for the representations.

Zaluski et al. (2003) developed a semi-automated tool that assists in authoring cases from textual historical experiences in the aerospace domain. Case creation involved the identification of symptoms and recommended solutions. Regular expressions were created by manually analysing the texts regarding fault isolation procedures. The most frequently used action verbs (e.g., *replace*) were used to create regular expressions such as */(replace) the (.*)/i.*. A set of rules were also generated from the trouble shooting manual. In the first stage of case creation, the regular expressions were employed to identify important features from manufacturer's documentation. In the second stage, the problem-solving experiences were then utilised in updating the cases that were created in the first stage. The manual intervention required to obtain regular expressions from the texts and to complete the case authoring makes the use of this approach prohibitive especially in domains such as SmartHouse where important pieces of texts appear in many ways and thus manually scanning the text for them is impractical.

Wiratunga, Koychev & Massie (2004) extracted features from free-form text using boosted decision stumps. They addressed the problem of semantics by employing association rule induction to capture patterns of co-occurrence between features, in order to acquire the semantic relationships inherent in the texts. This approach relies on frequency of first-order co-occurrences of words, and higher-order co-occurrences (Chakraborti, Wiratunga, Lothian & Watt 2007) are not taken into account. This makes the approach insufficient for domains with small collections such as SmartHouse, where some important features may not co-occur frequently enough to be identified. The approach is referred to as *Wiratunga*

[2]SMart Index LEarner.

et. al (2) 2004 in Figure 2.1.

SOPHIA (Patterson, Rooney, Dobrynin & Galushka 2005) is a domain independent system that automatically extracts features for representing textual documents. Sophia employs distributional clustering (Pereira, Tishby & Lee 1993) to create word groups that co-occur in similar documents. The word clusters can represent the textual documents and are also used to cluster semantically similar documents. However, Sophia requires the corpus to be heterogeneous and does not put word order into account. The former feature can be restrictive during the development of the case base. Taking word order into account can help to incorporate context and thus ignoring it can be detrimental to the retrieval effectiveness during problem solving.

Latent Semantic Indexing (Deerwester, Dumais, Furnas, Landauer & Harshman 1990, Berry, Dumais & O'Brien 1995) is a feature extraction technique that relies on word co-occurrence. The features so extracted are linear combinations of terms in the document collection and have an underlying structure. However, the interpretability of the representations and underlying concepts remains uncertain (which is why it has been shaded grey in Figure 2.1). Thus expert initiated refinement of knowledge is difficult for these features.

Thompson (2001) applied Ripper (Cohen & Singer 1996) a rule induction system, to obtain features for text categorization tasks in the law domain. Ripper features can be a set of word stems contained in the documents. Thompson's (2001) work is close in spirit to that presented by Wiratunga, Lothian, Chakraborti & Koychev (2005) and Wiratunga, Lothian & Massie (2006). Wiratunga et al. (2005) proposes Propositional Semantic Indexing (PSI), an approach for automatically acquiring indexing vocabulary. PSI uses rule induction to extract features that capture more semantics than the traditional IR-based approaches. The extracted features are logical combinations of existing features and are more understandable to humans than those produced using Latent Semantic Indexing. However, PSI is a supervised technique. In further work, Wiratunga et al. (2006) introduce 2 unsupervised methods for selecting features for representing cases. All these feature extraction approaches resulted in better retrieval effectiveness than IR-based ones but the representative features still lack an underlying structure relating the different concepts. Thus, these representations do not allow for case comparison at different levels

of knowledge detail.

Mining Text for Structure

Approaches that mine text for structure can further be categorised into 2 groups; those that create graphical structures and those that create hierarchical structures.

1. Approaches that Create Graphical Structures

CREEK (Aamodt 1994) is a knowledge intensive case-based reasoning method in which cases are submerged within a semantic network that acts as the domain model. The model comprises general and factual information. In the model, there are various relations through which the concepts are interlinked such that each concept has many relations to other concepts. CREEK also contains a top-level ontology and a case base. During problem-solving, CREEK makes use of syntactic information to get initial matches between a query and cases in the case base. It then uses general knowledge and a reasoning mechanism of plausible inheritance to generate an explanation that supports a hypothesis which allows it to make further inferences and thus refine the retrieved results it had obtained earlier. CREEK has a very good reasoning mechanism. However, it is not clear from the literature, how much of this knowledge was obtained automatically. However, since cases are embedded in a semantic network, CREEK will be classified as a graphical system. Another graphical system is that presented by Cunningham, Weber, Proctor, Fowler & Murphy (2004) which automatically constructs graphs from texts. Each case is a graph whereby the nodes are words. Word order is preserved by adding arcs between adjacent words. Representing the original texts with these graphs improved on the BOW approach because of its ability to capture word order and negation.

Déjà Vu (Smyth, Keane & Cunningham 2001) is a hierarchically organised system that designs plant-control software. Leaf nodes are tagged with sub-solutions. However, identifying abstract cases and discovering the relationships between them is done manually. Asiimwe, Craw, Taylor & Wiratunga (2007) address this shortcoming by automatically creating a concept hierarchy with which cases are automatically created and organised in their SMART system.

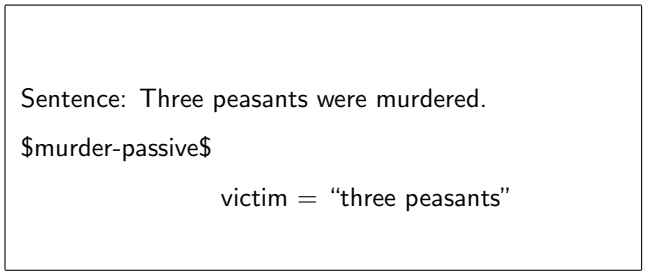
However, when cases are modelled as graphs, similarity between cases is typically obtained by finding graphs in the case base that contain the maximal common subgraph to the query. The complexity of this matching task is NP-hard. Furthermore, in systems such as possessiveciteCunningham-ECCBR-2004, the synonymy/polysemy problem is not addressed and so words that are the same are bound to be treated as different, thus reducing the similarity weighting between similar cases. The subgraph isomorphism problem is reduced to polynomial time when the cases being compared are both tree-structured (van Leeuwen 1990). This advantage of tree structures is what was exploited in this research, albeit at the expense of some of the expressiveness afforded by graphical representations.

2. Approaches that Create Hierarchical Structures

This category typically employs information extraction systems to obtain structured representations of the textual sources. Information extraction (IE) involves the selection of desired fields from the given information, by extracting common patterns that appear along with the information. A set of rules is required in order to identify relevant information for extraction. These rules are dependent on the domain of interest and are peculiar to the domain's writing style. They should be able to extract explicitly stated information and that which can be inferred.

IE systems exist that extract information from structured, semi-structured and unstructured textual sources. CRYSTAL (Soderland, Fisher, Aseltine & Lehnert 1995) and AutoSlog (Riloff 1996b) are example systems which make use of so-called concept nodes to fill a previously known template. A *concept node* (CN) is a case frame that is used to identify important information in the sorrounding text. Information extraction is triggered by activation of the frame using pre-determined linguistic expressions. Below is an example[3] CN used to extract information regarding the 'victim' from a sentence "Three peasants were murdered" in the terrorist domain:

[3]The example is borrowed from the same source.



The concept node is triggered by the verb `murdered` but is activated by the passive form of the verb (were murdered). That way, the object of the passive form of murder is extracted as the `victim`. Both systems use concept nodes and rules that distinguish between major syntactic constituents like subject, verb and direct object. However, in AutoSlog, a human edits the proposed concept definitions.

Protos (Porter, Bareiss & Holte 1990) is a system that uses a mixed-initiative approach to knowledge acquisition. The system may request the expert for knowledge or the expert may volunteer knowledge. Furthermore, there is a provision for the expert to incorporate knowledge about case features and their categories, at any time. In this system, in each concept category, there are concepts which are described extensionally with a collection of exemplars. Exemplars are described with features. For example, in the category `CHAIR`, there could be two exemplars `Chair1` and `Chair2`. The two chairs have a number of features common to both of them like `Seat` and `Backrest` but where `Chair1` has a feature `Wood`, `Chair2` may have a feature `Metal`. The system was seeded with some cases that were classified and explained by the expert. When presented with a new case, Protos attempts to classify it by comparing the case features with exemplars in its knowledge base. It also tries to provide an explanation for the classification. If the classification is wrong or the explanation insufficient, the expert is asked to provide more information. Protos uses the acquired knowledge to extend and revise existing knowledge so that the new case is correctly classified and explained. Cases that are considered useful for future classification are retained. Consequently, Protos learns not to make the same misclassification for the same type of case. The mixed-initiative approach is a good way to incorporate domain knowledge in the case authoring step and for retrieval (Gu & Aamodt 1994).

In an effort to manage a steadily growing set of textual documents in the domain of test-specification generation, Minor (1999) addressed the case-acquisition bottleneck by using a simple case authoring process. In this project, a test case mainly consisted of a

textual part and some additional attribute-value pairs. Some seed cases or parts of cases were used to aid in the authoring of more cases. The case author filled out some of the sections of a case and this information was then used to retrieve similar cases, an idea with the same subject, or clues on how similar cases were phrased. Seeding is an effective way of introducing knowledge on to which the system can gradually build.

In the THEMESEARCH system, Lenz & Glintschert (1999) considered documents as cases. They used information extraction techniques such as pattern matching and concept nodes to obtain a structured representation of the semantics of a given text.

DRAMA (Leake & Wilson 1999) is a system that uses CBR to facilitate retrieval and adaptation of expert knowledge that has been captured through concept mapping. Leake & Wilson (1999) categorised concept maps as semi-structured since they include structural information but do not use standard syntax or standard sets of attributes. These cases could also be described as weakly-textual since the textual parts of the cases were not the predominant factor used in reasoning (Wilson & Bradshaw 2000). Consequently, it was not necessary to employ sophisticated means to acquire indexing knowledge. Simple IR techniques based on the vector space model were employed for this task.

In the SMILE + IBP system, Brüninghaus & Ashley (2001) obtained better presentations by capturing actions in multi-word phrases, recognising negation and using abstraction for actors and events. An information extraction tool, Autoslog (Riloff 1996a) was employed to identify important features from marked-up case summaries. Autoslog employs NLP and IE methods to extract information from text. SMILE (Brüninghaus & Ashley 1999), a system that automatically indexes cases for CATO, then used the extracted information to learn indexing of cases and to classify case texts under CATO's factors. This resulted in a case representation that captured information relevant to finding factors.

In the DiscoTEX framework (Yong 2004), a structured template with pre-defined slots is constructed from text. The slots are obtained by first tuning an information extraction system using a corpus of documents annotated with filled templates.

Common among the approaches that rely on information extractors is that they are largely domain-specific and the user is required to have an idea of the target information. The need to label the texts in order to tune the information extractors makes these

approaches unscalable and thus impractical in real world scenarios. In the SmartHouse domain, IE methods have been exploited to obtain seed words for the key phrase extraction step. This was carried out by making use of commonly occurring word-patterns like *problem*, *impairment* and *difficulty*.

2.4 Extraction of Concept Hierarchies

Concepts are groups of terms with an underlying meaning. The thesis is that these concepts are better representatives of the text than a bag-of-words. The task of hierarchically structuring cases can benefit from methods that extract word/term hierarchies from text. Hearst's (1994) work aimed to discover taxonomic relations from electronic dictionaries and provides a basis for most approaches in this area. Hearst extracted patterns that can be used to detect hypernym-hyponym relationships between nouns where hypernym-hyponym is an *is-a* semantic relation between word meanings. The definition for a hyponym-hypernym relation is that: **x** is said to be a hyponymy of **y** if native speakers of English accept the sentence constructed as "An x is a (kind of) y" (Hearst 1994, Miller 1995, Hearst 1998). Conversely, **y** is a hypernym of **x**. An example of the phrases Hearst used to identify these relations is the phrase *such as*, in a sentence *...fruits such as banana* or the phrase *and other*, e.g., in *...banana and other fruits* to surmise that *fruit* is a hypernym of *banana*.

WordNet (Miller 1995) is a lexical database with a hierarchy of hypernyms-hyponyms for words. This makes it useful for semantic tagging of text. However, WordNet was manually developed and was found to be insufficient for domain-specific text. This shortcoming was addressed by Caraballo (1999) who sought to automatically build a hierarchy of nouns and their hypernyms from text, using no other lexical sources. First, for each noun, Caraballo created a vector containing counts of the number of times another noun occurred with it in conjunction or appositive.

Similarity of the vectors for noun-pairs was measured by computing the cosine of the angle between them. The most similar pairs were grouped under the same parent. She then calculated similarity for the parents and continued with this grouping until all the nouns were grouped. Nouns that had a zero similarity with every other noun were not

included in the tree. This may not be a desired feature for sparse domains as the similarity threshold may result in the exclusion of important nouns. It is from the nodes of this tree that hypernyms were obtained. It is worth noting that unlike most statistically-based work, in both Hearst and Caraballo's systems, a single occurrence was enough to detect a hypernym-hyponym pair. This is a desirable feature in domains such as SmartHouse, where data is sparse.

Sanderson & Croft (1999) developed a technique where term A subsumes term B if and only if the set of documents containing term A was (approximately) a superset of the set of documents containing term B. The approach did not rely on any knowledge source. Nanas, Uren & Roeck (2003) extended Sanderson & Croft's (1999) work by identifying symmetric relations between term pairs while Glover, Pennock, Lawrence & Krovetz (2002) have explored the use of anchor texts in a hyperlinked environment to determine the level of hierarchy of a term. Chuang & Chien (2004) showed the usefulness of the subsumption hierarchies from Sanderson & Croft (1999) for interactive query expansion. These techniques typically do not put the meaning of the words or word order into consideration and as a result, relations that show a term as being an aspect of another term, e.g., *disease - outbreak* are extracted alongside hyponym/hypernym relations like *banana - fruit*. Thus, although the approaches can be fully automated, many of the extracted relations are not useful in most applications.

Valencia-García, Ruiz-Sánchez, Vicente, Fernández-Breis & Martínez-Béjar (2004) present an interesting approach to knowledge acquisition from text. Their methodology is based on the fact that in natural language, knowledge entities are usually related using verbs. In this approach, verbs that represent relationships between concepts are stored and used to automatically identify the corresponding knowledge in text. Knowledge is incrementally acquired as each sentence is analysed in turn. Each word's grammar category in a sentence is used together with the incremental acquisition method to infer concepts, attributes and values. The knowledge extraction process is supervised by an expert to ensure semantic relationships are also extracted. The use of an expert in the knowledge acquisition process is a useful way to incorporate domain knowledge into the system especially when such knowledge is hard to acquire like it is in the case of the SmartHouse domain.

Maedche & Staab (2000) and Cimiano, Hotho & Staab (2005) developed an approach

for automatically acquiring concept hierarchies from a text corpus. They made use of NLP and Formal Concept Analysis to carry out shallow text processing and association rule mining. A parser was used to extract syntactic dependencies between words which were then used to produce a lattice. The lattice was converted into a kind of partial order constituting a concept hierarchy. A corpus of 1 million words was used to learn the relationships between words. Semantic disambiguation of ontology members was also carried out by an expert.

Other approaches have used the Web as a big corpus from which to test for validity of linguistic patterns from which a taxonomy is then built. Here the idea is to create a hypothetical relation e.g., *banana is an animal* and *banana is a fruit*. The number of hits returned is used to determine the validity of the relation. Pankow (Cimiano, Ladwig & Staab 2005) and KnowItAll (Etzioni, Cafarella, Downey, Popescu, Shaked, Soderland, Weld & Yates 2005) are some of the systems using this approach.

Finally, there are systems whose primary goal is not the extraction of a term taxonomy but to assist the user in identifying meaningful relations by offering few promising options. Examples are OntoMat (Volz, Handschuh, Staab, Stojanovic & Stojanovic 2004) and SemTag (Dill, Eiron, Gibson, Gruhl, Guha, Jhingran, Kanungo, Mccurley, Rajagopalan & Tomkins 2003).

2.5 Word Sense Disambiguation

Research in Word Sense Disambiguation (WSD) has its roots in Weaver's (1949/1955) work where he suggested a window of words that would help put the word to be disambiguated into context and thus successfully disambiguate it. Early approaches to WSD (Hirst 1987, Small & Rieger 1982) were based on rule-based systems which heavily relied on large amounts of hand-crafted knowledge. The cost of manually creating the rules rendered the systems untenable for large, usable systems. Thus these approaches were only useful in restricted domains. To overcome this weakness, more recent efforts such as those by Lehman (1994) and Charniak (1994) are based on statistical techniques which scale up more easily and can thus be applied to large corpora. One major drawback of these approaches is that the corpus may not have sufficient information to identify the

appropriate word/word relationships to support disambiguation. Later approaches such as that presented by Li, Szpakowicz & Matwin (1995) overcome this shortcoming by making use of WordNet to constrain the set of possible word classes and, like the approach employed in this project, depends on syntactic rather than statistical information. Our work is also related to that presented by Klapaftis & Manandhar (2005) who make use of Google to find contextually related terms. The terms help in assigning the correct WordNet sense to each term under disambiguation. However, their approach depends on frequency information regarding the word to be disambiguated which makes it unattractive for sparse domains.

In this project, the problem of synonymy and polysemy is addressed by harmonising the text before it is mined for structured cases. Synonym mappings are created for each polysemous word after which each word is disambiguated in order to map synonymous words to a common token to obtain harmonised text. The use of the synonym mapper allows standard text to be used for case authoring and removes the need to restrict queries to a standard vocabulary. This method neither relies on hand-tagged data nor on statistical information of words in the document collection.

2.6 Chapter Summary

Designing a case-based reasoning system can be challenging when the problem-solving experiences are captured as text. This is because the text has to be interpreted in a meaningful way in order to create cases that are useful for problem-solving. IR-based techniques are easy to deploy but result in knowledge-poor representations. On the other hand, sophisticated techniques that rely on natural language processing and information extraction techniques tend to be brittle and still require a lot of knowledge engineering effort to identify suitable patterns for feature extraction. Some researchers have, in the past, relied on information extraction tools but these too require a set of tagged documents for learning. Generally, approaches that rely on human effort for tagging, feature or concept identification or template generation, do not scale up and are thus impractical in real world scenarios. That is why this project creates a tool, *SmartCAT-T*, for automatically creating case representations without the use of tagged data.

Flat structured representations do not allow for case comparison at different levels of abstraction as do graph and hierarchically structured cases. However, comparison of graph structured representations can be a very complex task when the cases have a big number of nodes. On the other hand, tree-structured case representations are easier to compare during problem solving. Thus, the case representations created in this project are tree-structured. Hence, *SmartCAT-T* will lie in the bottom-right corner of Figure 2.1 as indicated.

The textual documents from which the cases are to be created are usually heterogeneous as the same words are seldom used to describe the same situations. Queries are also likely to comprise words or phrases that have the same meaning as, but are different from, those containing case knowledge. Thus, it is essential to address the problem associated with synonymy and polysemy so that texts are correctly interpreted during case authoring and query processing. Current research has addressed this issue by using a range of methods such as restricting the terms used in the text, tagging documents to tune information extractors, using word co-occurance distributions to obtain rules and hence capture context or ignoring the synonymy/polysemy problem altogether.

The next chapter presents the creation of a language model that was used in this project. Creation of the language model involves developing a domain-specific synonym mapper so that word synonyms are recognised as such enabling mapping of heterogeneous texts to standard case representations and queries.

Chapter 3

Text Standardisation by Synonym Mapping

This chapter focuses on creation of the language model and interpreting the texts according to that model, in order to create a harmonised knowledge source with which the conceptual model will be created. The task of modeling the language is applied to texts in the SmartHouse domain. Figure 3.1 illustrates the process of creating a standardised knowledge source using the documents in the collection. WordNet and all the documents are used to create synonym mappings for the domain. These are essentially mappings of word-groups to representative words in order to allow words of the same group to be replaced by their representative word in the text. The text harmonisation process changes documents into harmonised texts by making use of Google and the synonym mappings. A query will also be meaningfully interpreted by making use of the harmonisation process.

3.1 Creating Domain-specific Synonym Mappings

In order for a machine to make any reasonable attempt at interpreting the text it reads, it is important that it is able to recognise the different meanings of a given word or phrase. For this, it needs to have access to knowledge of the different synonyms a given word can take in the domain. A common approach to determining semantically related words is to determine their distributional similarities. The assumption here is that similar words will appear in similar contexts. Thus comparing contexts of two words can help to determine

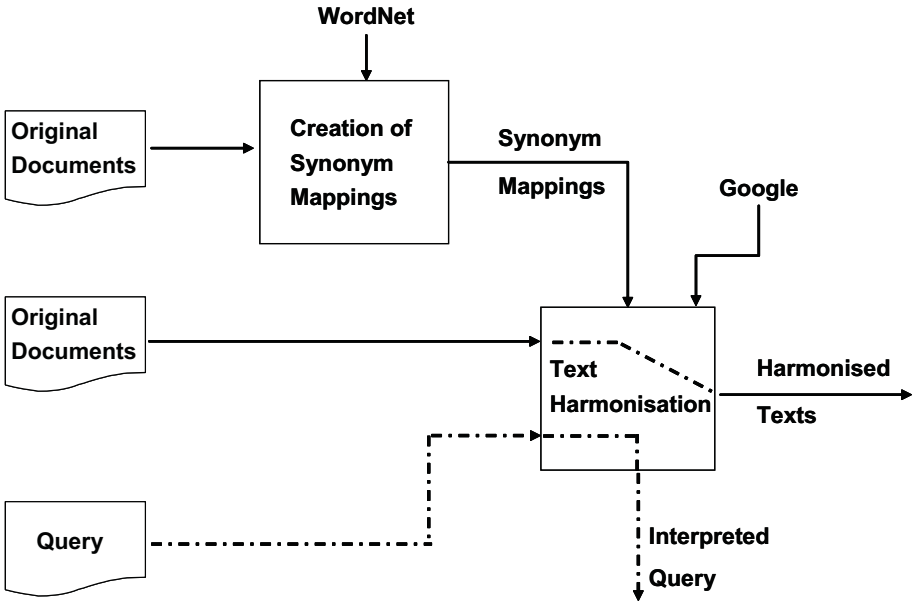


Figure 3.1: Text Standardisation and Query Interpretation

if the words are semantically related or not. The techniques presented by van der Plas & Bouma (2005) and Curran & Moens (2002) are good examples of such approaches. They make use of grammatical relations to define the context of a given target word.

Latent Semantic Indexing (Deerwester et al. 1990) is another technique that has been explored by many researchers wishing to determine similarity between terms. In Latent Semantic Indexing, the documents provide the term's context for it is assumed that semantically related words tend to appear in the same documents. Massie, Wiratunga, Donati & Vicari (2007) also rely on the thesis that similar words will distribute similarly. The commonality among these approaches is that they provide ranked lists of semantically related words where one would expect synonymous words to appear at the top and hyponyms[1] and hypernyms[2] at the lower ranks since the similarity between a word and its synonym should be higher than that between the word and say, its hypernym. Unfortunately, it is not always the case that the ranked lists reflect these different similarities. van der Plas & Tiedemann (2006) obtain better scores for precision and recall when they use parallel corpora where documents are translated into multiple languages, as compared to using monolingual corpora. However, these authors also report "unsatis-

[1] A hyponym is the subordinate of a given word.
[2] A hypernym is the superordinate of a given word.

factory" scores for precision. In order to ensure the reliability of synonym mappings, we use WordNet (Fellbaum 1998) a lexical knowledge base that was created by hand to look up the synonyms of the various polysemous words in our document collection.

3.1.1 WordNet

In WordNet, synonymous nouns, verbs, adjectives and adverbs are grouped into *synsets* where a synset represents a lexical concept. Synsets are interelated by means of concept and lexical based relations. Synonyms are organised according to concept-superconcept relationships called hyponymy, thus forming a concept hierarchy. For example, the noun *mango* is a hyponym (subordinate) of the noun *fruit*, and *fruit* is a hypernym (superordinate) of *mango*. WordNet also has links relating words representing relations such as antonyms, meronymy and so on. Using WordNet, we obtain clusters of words with similar meaning whereby, a word representing the cluster can be used to replace the others in text, and still retain the meaning of the original sentence.

3.1.2 Creating the Clusters

We cluster words according to their synonymy relationships; words that are synonyms in a given context are clustered together. In order to determine the context, we need the word's part-of-speech and the different meanings or *senses* of each word. A CLAWS part-of-speech tagger (Marshall 1987) was used for this purpose.

If two words with the same part-of-speech appear in the same WordNet synset, they are deemed to be synonymous in a particular context, and thus clustered together. We consider only strict synonyms i.e., words that belong to the same WordNet synset. So we build clusters of synonymous words and out of each cluster, we choose the word that appears most frequently in our texts and use it as a representative for the other words in the cluster. The representative word will substitute all the cluster words, in text during harmonisation. If more than one word in a cluster have the highest frequency, a representative is randomly chosen. To illustrate, suppose that we have the nouns, *house* and *home* appearing in the document collection with frequencies 2 and 3 respectively. Figure 3.2 illustrates 1 sense for each of the two nouns as given by WordNet. *House* has 12 senses in all and *home* has 9.

Sense 4 of **house**

family, household, **house**, **home**, menage -- (a social unit living together; "he moved his family to Virginia"; "It was a good Christian household"; "I waited until the whole house was asleep"; "the teacher asked how many people made up his home")
=> unit, social unit -- (an organization regarded as part of a larger social group; "the coach said the offensive unit did a good job"; "after the battle the soldier had trouble rejoining his unit")

Sense 7 of **home**

family, household, **house**, **home**, menage -- (a social unit living together; "he moved his family to Virginia"; "It was a good Christian household"; "I waited until the whole house was asleep"; "the teacher asked how many people made up his home")
=> unit, social unit -- (an organization regarded as part of a larger social group; "the coach said the offensive unit did a good job"; "after the battle the soldier had trouble rejoining his unit")

Figure 3.2: Sample Wordnet Senses for *House* and *Home*

House is synonymous with *home* in *house*'s sense number 4 and *home* is synonymous with *house* in *home*'s 7th sense. Since *home* is more frequent in the document collection, it is made a representative of the synset in which *house* appears. The assumption is that the most frequently used version of a word is also the most preferred in the domain. We only consider the frequency of the word in the particular part-of-speech. This is because a word could be frequently used in say, its noun form and not as a verb. If we used the overall frequency, a rarely used word might be used to represent a cluster. Linguistically, this should not pose a problem since all the text will be mapped onto a common conceptual model eventually. However, we want the conceptual model to comprise words that are fairly common in the domain to ensure it is easy to comprehend by the user.

Furthermore, to ensure that words from unseen text will be correctly interpreted, all other synonyms of the representative word for this sense are included in the clusters. Thus this process selects WordNet synsets that are relevant to the domain and use the most frequent words in the synsets as their representatives. These representative words will be used in place of each word in a synset, whenever the word appears in one of our documents or a query.

3.2 Harmonising the Text

The harmonisation task aims to remove syntactical differences between texts that are semantically equal so that the process of creating a conceptual model interprets similar texts as such. Words like "*dwelling*", "*house*", and "*home*" all mean essentially the same thing and as such, one of them could be used as a substitute for the others and still retain a significant portion of the original meaning of the sentence. This results in less sparse and more cohesive data. The harmonisation task is analogous to *data cleansing* in the data mining community. The difference is that the data cleaning community aims to remove errors and duplication in data; we harmonise the text by using the same words to express a particular meaning, which in turn, also helps to avoid duplication.

Text harmonisation is carried out in two stages. In the first, possessive pronouns are replaced with one common article ***"the"***. For example, the sentence "*He found it difficult to open* ***his*** *door*" becomes "*He found it difficult to open* ***the*** *door*".

In the second stage, each polysemous word[3] is replaced with its representative synonym. However, the word needs to be assigned its appropriate sense in order to determine the right representative with which to replace it, if the meaning of the original sentence is to be retained. Therefore, this stage involves the resolution of syntactic and semantic ambiguity in text; problems that are common in natural language processing.

3.2.1 Resolving Ambiguity

Humans incrementally process the meaning of what they are reading, as they progress through a given sentence. Thus, understanding the text requires that one makes a decision as to the most likely of all the possible meanings of the text (Simpson 1984). This is because the English language is highly ambiguous having words with multiple related meanings (polysemy), different words with the same spelling (homography), and words with different spellings that sound the same (homophony). Syntactic ambiguity is easy to resolve as it only requires that the word's part-of-speech be known to ensure that a word is replaced by one with the same part-of-speech. Resolving semantic ambiguity requires that the right sense of the word be obtained. The obvious difficulty with this task is that polysemous

[3] A polysemous word is one with more than one different meaning in text.

words will have more than one sense. The task therefore, is to find the meaning of the word, in the context in which it has been used, which is also the means by which humans carry out disambiguation.

A polysemous word w will appear in as many clusters as the number of the different meanings it has, as obtained from the document collection. However, our synonym mapper is based on the documents we have at hand and we can therefore not claim to have all the knowledge that is relevant to the domain, in these documents. This essentially means that there might be synsets that were not included because they were not shared by more than one word in our collection, a factor that triggers the clustering process. Therefore potentially, each word has at least one sense that was not stored. Thus our task is to determine if a given target word should be replaced by one of its possible representative synonyms and if so, to choose the correct synonym representative that will not change the meaning of the sentence. We shall refer to each of the potential representatives as a *potential representative*.

One major factor to consider when standardising text is that, in the SmartHouse domain, single words are often not meaningful, and thus concepts will comprise more short phrases than single-word terms. For example, phrases like *abnormal gait* and *dexterity problem* will be important but the individual words on their own may not carry much meaning. Thus not only do we want to retain the original meaning of the sentences, we also want the phrases that will be extracted in the process of creating the conceptual model, to be meaningful. Thus, when we replace a word with its representative, we want the short three or more word phrase in which the word appears, to still be grammatically sound.

Although Word Sense Disambiguation (WSD) has been reported to improve precision in information retrieval (Schutze & Pedersen 1995), some authors like Sanderson (1994) reported poorer performance unless disambiguation was at least 90% accurate. Thus disambiguation must be highly accurate if the language model is to be effective in case creation and query processing. We hypothesise that one way to achieve high accuracy in unsupervised Word Sense Disambiguation, is to disambiguate only those words for which evidence to support the disambiguated version can be obtained. To illustrate, suppose that we have a sentence "*He found it difficult to open the front door*". Now, *front* has 10 senses

in its noun version. In its 6th sense, *front* is synonymous with *movement* [4]. Suppose that we wish to disambiguate the word *front* thereby determine whether it should be replaced by its synonym *movement* in order to harmonise the text. If *movement* is a synonym of *front* in the context of this sentence, then it should be possible to replace *front* with *movement* and still retain the meaning of the sentence. So if we can find evidence that *movement* can replace *front* in this context, then we conclude that in this sentence, *front* can be disambiguated, otherwise, we do not carry out the disambiguation.

3.2.2 Obtaining the Context

We have set out to determine the correct sense S_w of a given polysemous target word w in the context C_w, out of all the possible senses of the word as obtained from our synonym mapper. To obtain the context of the target word, we use syntactic information contained in the sentence in which it resides. We shall refer to the sentence containing the target word as the *target sentence*. The context of the target word is then obtained as follows:

1. Obtain a parse of the target sentence by using the Link grammar parser (Sleator & Temperley 1993).

2. If the target word is a noun, use the parse to extract the noun phrase in which the target word appears. If the target word is a verb, extract the verb phrase. This shall be referred to as the *target phrase*.

3. Extract the verb or noun associated with the target phrase depending on whether the target word is a noun or a verb. We shall refer to this as the *extension*.

The target phrase and the extension make up the context C_w that is used to derive the correct sense S_w of the target word w in the target sentence.

3.2.3 The Word Sense Disambiguation Algorithm

The task is to determine if a given word w should be replaced by any of its representatives, and if so, to choose the representative that will not change the meaning of the original sentence. Suppose that a polysemous word w has several potential representatives. We

[4] Meaning *a group of people with a common ideology who try together to achieve certain general goals.*

use the context obtained above (steps 1 to 3) to carry out Word Sense Disambiguation as follows:

1. Get the target phrase and replace the target word with the first potential representative. Use this *potentially standardised* phrase and the extension to create a Google query.

2. Retrieve the documents in which the query terms appear in the **same** sentences ensuring that the potential representative and the extension have been used with the same respective parts-of-speech as the target word and extension have, in the target sentence.

3. Repeat the procedure for all potential representatives.

4. Choose the potential representative with the largest number N of returned documents. If $N \geq 1$, then the potential representative can be used to replace the target word, in the target sentence.

The Word Sense Disambiguation process is illustrated in Figure 3.3. The target sentence contains the target phrase and the extension. The target word in the target phrase is replaced by one of the potential representatives to create a potentially standardised phrase. The potentially standardised phrase and the extension are used to generate a Google query. The target sentence and all google returned sentences in which the potentially standardised phrase and the extension appear together, are parsed using the Link Grammer Parser. Each Google sentence parse is compared to the parse of the target sentence. If there are Google sentences in which the potential representative and the extension have the same Parts of Speech as the target word and the extension have in the target sentence, then disambiguation takes place.

The requirement that the potential representative and the extension appear in retrieved documents with the same respective parts-of-speech as the target word and extension do in the target sentence, is a good contributing factor towards ascertaining context. Thus finding a document in which this condition is fulfilled is taken to be evidence that the potentially standardised phrase is a sensible one. The assumption is that if there is a document in which someone has used the potential representative in the same way as the

target word, then the potential representative can be used in place of the target word in this sentence, without altering the meaning.

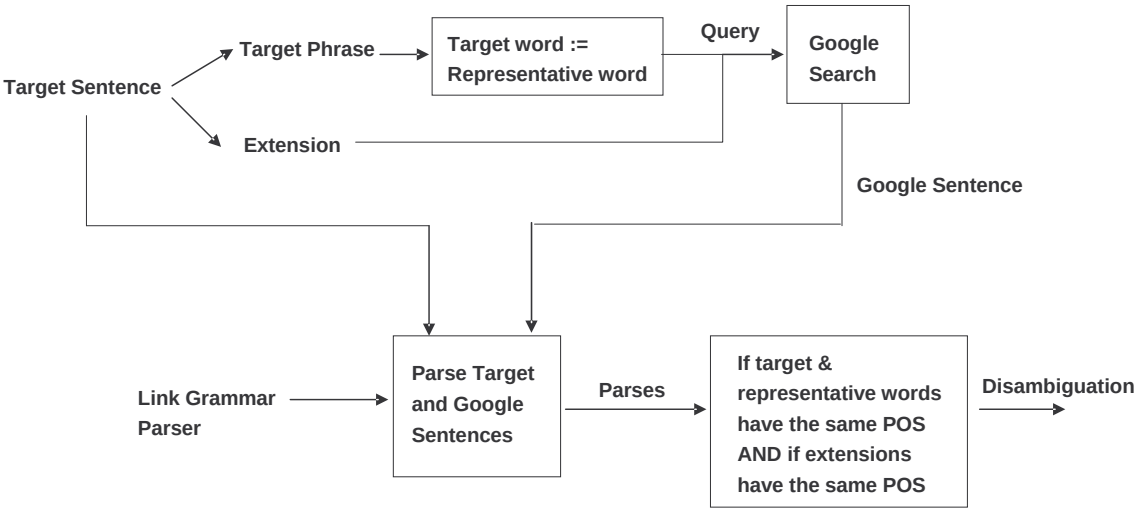


Figure 3.3: The Word Sense Disambiguation Tool

The internet is used as a very large corpus where we assume that if the potentially standardised phrase is sensible, the probability of getting a document in which it appears in the same sentence as the extension and used in the same context as the target sentence, tends to 1. If no document is returned after the above operation, we have no evidence that replacing the target word with that particular potential representative will not alter the meaning of the target sentence and we thus do not replace the target word with the representative word. The level of confidence of the disambiguation is dependent on the number of returned documents in which the potentially standardised phrase is used in the same context as the target phrase. It is also dependent upon the length of the target phrase – the bigger the phrase, the more confident we are of the disambiguation. It is worth noting that although the algorithm is aimed at resolving ambiguity of single words, the use of target phrases that are not single words results in the disambiguation of some longer phrases.

Consider again the sentence *He found it difficult to open the front door.* Suppose that we wish to determine if *front* should be replaced with *movement* in this sentence. Figure 3.4 shows a parse of the target sentence. In this case, the noun phrase *the front door* will be the target phrase and the extension will be *open.* In order to determine if

movement is a synonym for *front* in this sentence, we replace *front* with *movement* in the target phrase changing it from *the* ***front*** *door* to *the* ***movement*** *door*. This potentially standardised phrase and the extension are used to create a Google query. Google returns a document on Rugby news in which the query words appear in the same sentence: "*And during the season,* ***the movement door*** *remains* ***open***." The parse of this sentence is shown in Figure 3.5.

```
he found.p it difficult.a to open.v the front.n door.n .

Constituent tree:

(S (NP He)
   (VP found
       (NP it)
       (ADJP difficult
             (S (VP to
                    (VP open
                        (NP the front door))))))
   .)
```

Figure 3.4: A Parse of the Sentence "*He found it difficult to open the front door.*"

Examination of the parses for target and Google-returned sentences shows that, *open* in the potentially standardised phrase returned in the Google document is not used in the same context as that in our target phrase. This is evidenced by the different parts-of-speech the extension, *open*, exhibits in the different sentences. In the target phrase, the extension is used as a verb and in the Google sentence, it is used as an adjective. Since no other documents are returned where the context is the same as that of our target word, we conclude that in this instance, *front* cannot be replaced by *movement* and still retain the meaning of the original sentence.

3.3 Evaluation

We are only interested in harmonising sentences where we can obtain accurate disambiguation. Thus our focus is on the correctness of the harmonised sentences and not necessarily on the overall effectiveness of the disambiguation tool. The Word Sense Disambiguation algorithm has been tested on documents in the SmartHouse domain. Ten sentences containing 14 polysemous words were harmonized and presented to a domain expert and to

```
and during the season.n , the movement.n door.n remains.v open.a .

Constituent tree:

(S And
   (PP during
       (NP the season))
   ,
   (S (NP the movement door)
      (VP remains
          (ADJP open)))
   .)
```

Figure 3.5: A Parse of the Sentence "*And during the season, the movement door remains open.*"

a native speaker of English. The sentences chosen were those that were deemed to have sufficiently changed from their original versions. Since it is sometimes impossible even for humans to pick the right sense of a word from WordNet, we thought asking our two test experts to do the same would not be reasonable. Instead, we placed the representative word right next to the target word in the text and asked them whether replacing the target word with the representative word would alter the meaning of the sentence. The target and representative words were highlighted, each with different colors to make the task easy. Figure 3.6 shows 5 of the harmonised sentences put to our experts. The target words in the original sentences and the representative words in the harmonised sentences are shown in bold for clarity.

All disambiguations were deemed correct by both experts i.e., replacing words with their representatives did not alter the meaning of the sentences. In 4 instances, the native English speaker thought that although the meaning of the sentence had not been altered, the representative word was as good as the word it was replacing while in the other cases, she preferred the representative word. This shows that it is possible to harmonise text and not loose important information, which is what we set out to do. The sentences presented to the experts were those that our tool was able to harmonise so this was not a test of the overall performance of the Word Sense Disambiguation tool but rather, of the correctness of those words that were used to replace others during text harmonisation.

Important domain terms such as *wheelchair* or *mobility problem*, will usually appear as nouns or noun phrases. Furthermore, in natural language, knowledge entities are usually

Original Sentence	Harmonised Sentence
She had a tendency to leave the **house**, leaving the carer behind.	She had a tendency to leave the **home**, leaving the carer behind.
She relied on **care staff** to close **her** windows.	She relied on **care worker** to close **the** windows.
He had **trouble** sleeping, constantly waking throughout the night.	He had **difficulty** sleeping, constantly waking throughout the night.
Mr. RC was at risk from fire and scalding because he was unaware of the special **dangers**.	Mr. RC was at risk from fire and scalding because he was unaware of the special **risks**.
…because of her **poor vision** relied on listening to the commentary rather than watching the action.	…because of her **poor sight** relied on listening to the commentary rather than watching the action.

Figure 3.6: Sample of Original and Harmonised Sentences

related using verbs and so verbs or verb phrases are also likely to be important. Consequently, the key phrase extraction task is aimed at identifying important nouns (and noun phrases) and important actions. Thus in order to reduce on computational effort, only nouns and verbs are considered for harmonisation. Figure 3.7 shows the transformation in the original text of the *wandering* sub-problem if Figure 1.2. (a) is the original text, (b) the lemmatised version, and (c) is the harmonised text for the sub-problem. The harmonised terms are in bold for purposes of clarity.

3.4 Chapter Summary

This chapter has presented an unsupervised approach to Word Sense Disambiguation. WordNet is used to assign each polysemous word with a sense that is applicable in the domain. The Word Sense Disambiguation algorithm makes use of syntactic information of the sentence in which a target word appears and information from Google, to assign the correct sense to the target word, in the context in which it appears. This approach has potential applications for ontology learning and general knowledge modeling. The texts returned by Google provide evidence that the disambiguated phrase is plausible in the English language. Although automated, the technique neither relies on statistical information of the word in the document collection nor does it require the use of hand-

(a) Original Text

> ***Wandering:***
> *Mrs M had a tendency to leave the house and wander off. She had recently wandered away from the house in the middle of the night. Her husband had found her outside, dressed only in a thin nightgown, trying to open the side door to a neighbor's home...*

(b) Lemmatised Text

> *wandering mrs m had a tendency to leave the house and wander off. she had recently wander away from the house in the middle of the night. her husband had found her outside, dress only in a thin nightgown, try to open the side door to a neighbor home...*

(c) Harmonised Text

> *wandering mrs m had a tendency to leave the* **home** *and wander off. she had recently wander away from the* **home** *in the middle of the night. her husband had found her outside, dress only in a thin nightgown, try to open the side door to a neighbor home...*

Figure 3.7: Harmonisation of the *wandering* sub-problem

tagged data.

Chapter 4

Key Phrase Extraction

In the previous chapter, the textual documents were harmonised in order to make the data more cohesive. It is important that the cases and conceptual model capture relevant domain knowledge if they are to be useful for problem-solving. In the SmartHouse domain, concepts are groupings of descriptors of people's diseases, disabilities, and their manifestations. In this chapter, key phrases that make up the concepts which enable the creation of a conceptual structure and cases, are identified and extracted. Some of these key phrases may be in the report summaries but are predominantly embedded in the problem-description texts. Each difficulty described in a sub-problem will be represented by its key phrases before all the representations are exploited to create a conceptual model and consequently, the cases.

The process of key phrase extraction begins in Section 4.1 where terms that are likely to contain useful information are extracted. It is out of the extracted terms that key phrases will be obtained. Key phrases will comprise disease-terms, disability-terms and other terms like "*unable to open the door*", that are key descriptors of people's difficulties. Disabilities and diseases are identified in Section 4.2. Other terms that are key problem-descriptors are identified in Section 4.3.

Figure 4.1 shows the architecture of the key phrase extraction module. Terms are extracted from each document's pre-processed sentences. All extracted terms are utilised in the construction of a disability look-up table. All terms and seed words are used to identify disability and disease terms that are explicitly stated in each document. The

identified explicit disability and disease terms together with implicit disability-terms that can be obtained by making use of the disability look-up table, all comprise a documents disability and disease terms. All extracted terms and Latent Semantic Indexing (LSI) are used to identify important topics in the document collection. The key topics and a document's disability and disease terms are all used to discover that document's key problem descriptors that are not disability or disease terms. All the discovered key problem descriptors together with the document's disability and disease terms make up the key phrases for that document.

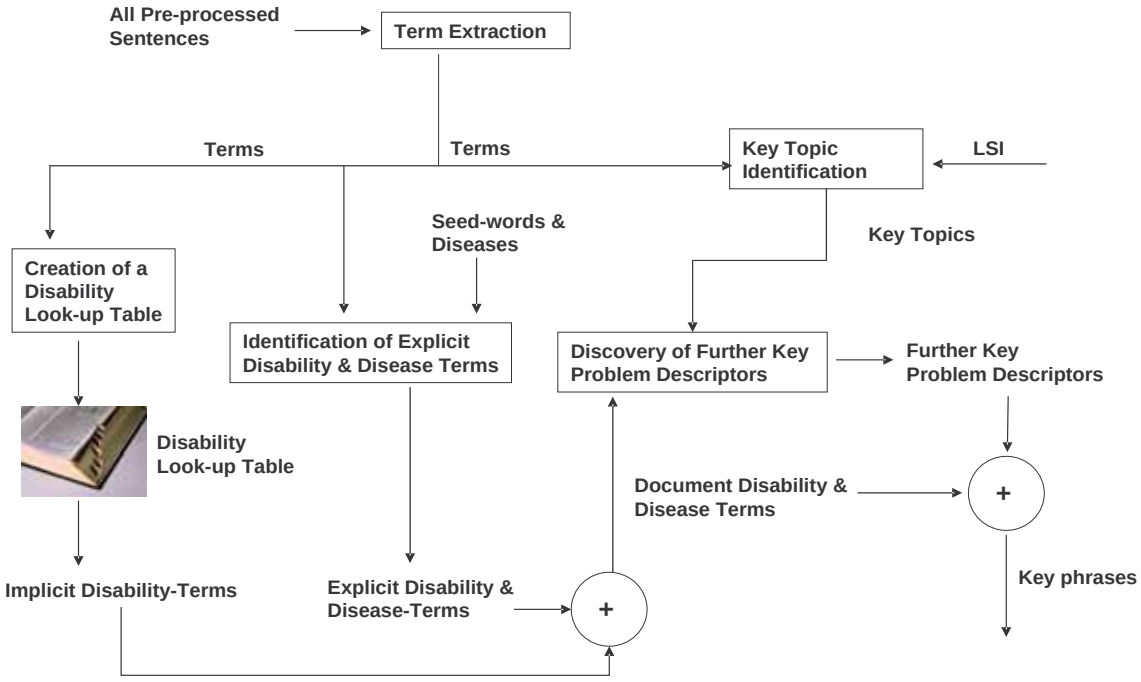


Figure 4.1: Architecture for the Key Phrase Extraction Module

4.1 Term Extraction

Term extraction begins with a pre-processing step where non-alphanumeric and numeric characters are removed. Words are then transformed into their respective lemmas[1] with the help of a lemmatiser in the RASP system (Briscoe, Carroll & Watson 2006). Using lemmas reduces computational effort but also ensures that words are reduced to forms that are complete words. We choose to use word lemmas instead of word stems because

[1]Lemmatisation is the process of stripping a word in order to leave its root form.

aggressive stemmers like Porter's (Porter 1980) sometimes produce word stems where the original word is not clear. Lemmas are a better alternative to aggressively stemmed words because we want to make the case representations and the nodes in the concept hierarchy easy to comprehend in order to allow manual refinement.

Research involving feature extraction and creation of concept hierarchies has often focused on words and finding patterns and relationships between them. In the SmartHouse domain, important concepts are often characterised by short phrases comprising 2 or 3 words like *hearing impairment* and *unable to communicate*, and only a few single words are highly meaningful. Thus we are interested in finding patterns and relationships between terms that comprise mostly phrases that are longer than one word, and it is such terms that are extracted. Terms are extracted in the form of trigrams, bigrams and unigrams. Each document is split up into sentences. A sentence is taken to be a group of words that is separated by a period, comma, bracket or other delimiting punctuation. Terms are then obtained from each sentence as follows:

1. All word subsequences of length 3 (trigrams) are extracted, discarding all that begin or end with a stopword.

2. All word subsequences of length 2 (bigrams) are extracted, discarding all that begin or end with a stopword.

3. All non-stopword unigrams are extracted.

The effect is that each sub-problem is transformed into a set of terms consisting of lemmatised phrases. Extracted terms are also meaningful because they have not been distorted by the removal of internal stopwords. Figure 4.3(a) is a lemmatised version of the text describing the *wandering* sub-problem in Figure 1.2, which is shown here again in Figure 4.2 for the reader's convenience. Figure 4.3(b) illustrates the extracted terms. These terms will be filtered to obtain those that actually contain useful knowledge.

4.2 Identification of Disease and Disability-terms

Disease-terms are identified by making use of a list of known diseases {*cerebral palsy, multiple sclerosis, dementia, ataxia, aphasia, cerebral vascular accident*} that result in

Mrs M, Y Street, Edinburgh. Initial Assessment Visit:
Mr and Mrs M were a couple in their eighties living in Y, and Mrs M had a tendency to wander due to ***Alzheimer's disease*** *... A number of difficulties were identified:*
Wandering:
Mrs M had a tendency to leave the house and **wander** *off. She had recently wandered away from the house in the middle of the night. Her husband had found her outside, dressed only in a thin nightgown, trying to open the side door to a neighbor's home... The final choice of equipment consisted of:*
Bed occupancy sensor
The bedroom light was automatically turned on at 8 pm. When the couple went to bed, the light went off as it sensed their presence in the bed by means of a pressure pad under the mattress...

Figure 4.2: Report Excerpt

(a) Lemmatised and Harmonised Text

wandering mrs m had a tendency to leave the home and wander off. she had recently wander away from the home in the middle of the night. her husband had found her outside, dress only in a thin nightgown, try to open the side door to a neighbor home...

(b) Extracted terms

wandering, tendency to leave, leave the home, home and wander, tendency, leave, home, wander, wander away, away, night, husband, thin nightgown, dress, thin, nightgown, open the side, side door, open, side, door, neighbor

Figure 4.3: Sample Text and Extracted Terms

disabilities. The list is compiled using brochures from the website of Tunstall,[2] one of the leading providers of telecare solutions in the United Kingdom.

Discovery of disability-terms is carried out in a more automated manner by exploiting the fact that people's disabilities are referred to as types of *difficulty*, *problem* or *impairment* in the SmartHouse reports. It is these words that are used as seeds that aid identification of disability-terms in the document collection.

4.2.1 Identification of Disability-terms

A logical process has been developed to identify disability-terms in the text. The process is as follows. First, seed-words are obtained by making use of lemmatised versions of the words *difficulty*, *difficulties*, *impaired*, *problem*, and *impairment*. Figure 4.4(a) shows

[2]Further information can be found at: http://www.tunstall.co.uk

the seed-words. All the extracted 2-word terms (Section 4.1) that match with the seed-words are then selected. This yields disability-terms such as *hear impairment* (hearing impairment), *impair mobility* (impaired mobility), *learn difficulty* (learning difficulties) and *mobility problem* as shown in Figure 4.4(b).

(a) Seed-words

problem, impair, impairment, difficulty

(b) 2-word Disability-terms Containing Seeds

mobility problem, impair mobility, hear impairment, hear difficulty, impair hear, learn difficulty, impair cognition, cognitive problem, dexterity problem...

(c) New Seed-words Comprising Important Parts of Disability-terms in (b)

mobility, mobility, hear, hear, hear, learn, cognition, cognitive, dexterity ...

(d) Disability-terms Extracted Using the New Seeds in (c)

mobility problem, impair mobility, mobility was slow, severe mobility problem, hear difficulty, impair hear, learn difficulty, poor dexterity, dexterity problem, slow cognition, impair cognition, cognitive problem...

(e) Resulting Disability and Disease Terms

mobility problem, impair mobility, mobility was slow, severe mobility problem, hear difficulty, impair hear, learn difficulty, poor dexterity, dexterity problem, slow cognition, impair cognition, cognitive problem, dementia, ataxia, aphasia, cerebral palsy, multiple sclerosis...

Figure 4.4: The Various Stages of Disability Term Discovery

The seed-words do not carry much meaning on their own since in the SmartHouse domain, everything is *a problem*, *a difficulty*, *difficult*, and everyone has some type of *impairment*. On the other hand, words that appear with the seed-words are significant as evidenced from the two-word terms that are extracted in Figure 4.4(b). Thus in the next step of disability term discovery, words that appear with the seed-words as shown in Figure 4.4(b) become new seed-words. For instance, from the term *hear impairment* we use *hear*, from *impair mobility* and *mobility problem* we use *mobility*, and from *learn difficulty* we utilise *learn*, and so on. Figure 4.4(c) shows a sample of the new seed-words. Pattern-matching with this new set of seed-words and all extracted terms yields terms describing people's disabilities in various ways. Aside from the disability-terms,

this step also captures the *degree* of impairment of a person if it is explicitly stated in the document. While in the previous step *mobility problem* was captured, this step captures ***severe*** *mobility problem* or even ***low*** *cognitive* ***functioning*** which matches with *cognitive* in *cognitive problem*. Figure 4.4(d) shows some of the extracted disability-terms with the new ones underlined. The list of diseases is added to this set of disability-terms to obtain the full set of disability and disease terms. Figure 4.4(e) shows some of the disability and disease terms obtained; the diseases are underlined for clarity.

4.2.2 Construction of a Disability Look-up Table

A look-up table is constructed to enable the search for disability-terms in instances where the disabilities are not *explicitly* stated. The look-up facility is important because SmartHouse device recommendation is based on the individuals' disabilities and thus, the disability has to be determined in order to recommend the appropriate assistive technology. Consequently, it is paramount to reflect in each sub-problem's representation, the disabilities responsible for that difficulty, if the representation is to be useful in recommending SmartHouse devices for new clients. The look-up table provides a means for determining disability-terms that are only mentioned *implicitly*.

The terms extracted in sub-section 4.1 enable discovery of interactions between terms in the different documents. These interactions are used to generate association rules. What we need are rules whose conclusions are disability-terms. That way, documents in which the body of the rule appears can be mapped on to the disability term. We argue that since extracted terms are typically 2 or 3 word phrases, co-occurrences of different terms is significant. Thus co-occurrences of terms in the different sub-problems are employed to mine association rules relating the sub-problems.

The higher the number of co-occurrences between any two terms, the more the evidence that they are related. Thus the number of co-occurrences determines the support for the consequent rule. We do not expect to have a high number of co-occurrences since the terms do not comprise only single words. Therefore a low support threshold of 3 is used. Only those rules whose conclusions are disability-terms are taken into account, and of these only those whose conclusions apply for every term in the body (i.e., rules with a confidence of 100%), are selected. Consequently, we are able to generate rules like

cerebral palsy→mobility problem $\{4\}$, where 4 is the support for the rule. The rules allow the association of a disease like *cerebral palsy* with the disability *mobility problem* or a term like *wander* with *dementia*. Hence it is possible to use some terms in a document to associate it with disability-terms that were not explicitly stated in the document. This information is stored in a way that allows the system to look up disability-terms by making use of some other terms. Figure 4.5 shows the structure of the disability look-up table. The terms in the column marked *Term* can be used to search the table; the terms return the disability-terms in the *Disability* column.

Term	Disability
forget	dementia, learning difficulties,
confusion	dementia
risk from fire	dementia, learning difficulties, mobility
risk of fall	dementia, mobility
wander	dementia
flood	dementia, learning difficulties
spinal tumour	mobility
alzheimers disease	dementia
multiple sclerosis	mobility
cerebral palsy	mobility
wheelchair	mobility
close the door	mobility
intercom operation	mobility, hearing impairment
walk stick	mobility
behaviour problem	learning difficulties
window open	mobility, dexterity
close the window	mobility, dexterity
open the door	mobility, dexterity
rely on care worker	mobility, dementia, dexterity
telephone operation	mobility, hearing impairment
bathroom	mobility, dementia, learning difficulties
flood	learning difficulties, dementia
stroke	cerebrovascular accident

Figure 4.5: Structure of the Disability Look-up Table

4.2.3 Assigning Disabilities to Each Document

A document's *explicitly stated* disability and disease-terms are obtained by identifying in the whole set of disease and disability-terms, those that appear in that document. Each of the document's extracted terms is then used to search in the disability look-up table, any disability-terms that might pertain to the document, but were not explicitly mentioned in the document's text. Consider the document in Figure 4.6. In the document, the disease *Alzheimers disease* is explicitly mentioned. However, the text also contains the term *wander* which returns the disability *dementia*, when used to search for disabilities in the disability look-up table. *Alzheimers disease* also returns *dementia*, when used to search for disabilities in the disability look-up table. Thus this document will have a set of disease terms {*Alzheimers disease*} and a set of disability-terms {*dementia*}.

However, a disability-term is only considered as pertaining to a document if it is identified by at least two-thirds of the terms in the document that can look up disability terms. The requirement that at least two-thirds of the terms support a choice for a disability-term helps to filter out disability-terms that are irrelevant with respect to that sub-problem. Support from half of the terms would be equivalent to random selection and thus a bigger fraction was required.

Mrs M, Y Street, Edinburgh. Initial Assessment Visit:

Mr and Mrs M were a couple in their eighties living in Y, and Mrs M had a tendency to wander due to ***Alzheimer's disease*** *... A number of difficulties were identified:*

Wandering:

Mrs M had a tendency to leave the house and **wander** *off. She had recently wandered away from the house in the middle of the night. Her husband had found her outside, dressed only in a thin nightgown, trying to open the side door to a neighbor's home... The final choice of equipment consisted of:*

Bed occupancy sensor

The bedroom light was automatically turned on at 8 pm. When the couple went to bed, the light went off as it sensed their presence in the bed by means of a pressure pad under the mattress...

Figure 4.6: Report Excerpt

4.3 Identification of Other Key Problem-Descriptors

Key problem descriptor terms that are not disease-terms or disability-terms are identified by using disease-terms or disability-terms as *anchors*. Anchors are key phrases that are also pointers to other terms that are key. The idea of using anchors to find other key phrases is analogous to hints or clues that a human would need to perform the same task. A person who is familiar with the domain will not need hints as to what will comprise key phrases. However, one who is not familiar with the domain, will need a feel of what would typically comprise a key phrase. In the SmartHouse domain, such a person might need to be told that *"words that are related to difficulties, problems, impairments or diseases will be important"*. The person will then use these clues to identify terms that are important. This use of background knowledge renders the system independent of large volumes of data whist allowing it to learn from small data sources.

McCartney & Sanders (1990) argue that case representations should be complete as it is necessary for the case to have the knowledge necessary for problem-solving. Consequently, the system aims to strike a balance between case completeness and the risk of including terms that will reduce the precision and overall accuracy of the system. This is achieved by identifying as many important terms as possible, while avoiding inclusion of terms that are irrelevant or harmful to problem-solving.

Anchors are obtained by noting that the text describing people's problems must in some way be related to the disability or the disease causing their difficulties. Thus disability-terms and disease terms are used as anchors with which other key problem-descriptors in the document collection are identified. Each problem-descriptor is identified by its anchors whereby, a term's *anchor* is one which occurs in the *same* document as the term. The anchor set comprises disability-terms pertaining to the document(which may be explicitly or implicitly stated) and any disease terms that are explicitly stated in the document.

A documents' terms are used to search the disability look-up table, for any disabilities pertaining to the document but were only implicitly mentioned. In addition to the implicitly stated disability-terms, a document may have one or more that are explicitly stated. Having more than one disability-term in a document could be because the person had more than one disability or another person's disability, say, that of their spouse, might be

mentioned in order to put the subject's problems into context. Furthermore, it is possible for a sub-problem to be due to more than one disability. For example, a wheelchair user may find it difficult to open their door because they do not have sufficient strength in their hands (*dexterity problem*) to carry out this task from a sitting position (*mobility problem*). Thus a document will have one or more members in its anchor set.

Latent Semantic Indexing is a popular technique for identifying features in a document collection. Anchors and latent semantic indexing are in this work utilised in the discovery of key problem-descriptors that are not disability or disease-terms.

4.3.1 Latent Semantic Indexing

Latent semantic indexing (LSI) is a technique that identifies patterns of term[3] distributions across a set of documents (Deerwester et al. 1990, Berry et al. 1995). The underlying principle in LSI is that similar terms will occur in the same documents. A set of terms whose patterns are sought and a set of documents containing these terms are used to create a term-document incidence matrix. An example of a term-document matrix from the SmartHouse domain is shown in Figure 4.7. Each term results in a space with many dimensions and has an entry in each vector that represents a document. Conversely, each document is a vector with as many entries as there are terms. Consequently, documents that have many terms in common will have their vectors near each other in the space. For example in Figure 4.7, document 3 has the terms *wander* and *dementia* in common with document 6. Indeed, the angle between document 3 and document 6's vectors is smaller (25.3°) than that between say, document 5 and document 6's vectors (90°). Thus documents 3 and 6 appear to be more similar to each other than to documents with which there are few or no common terms. Similarly, each row of the term-document matrix can be viewed as a term vector where each entry represents the weight of the term in each respective document.

Decomposition of the term-document matrix is a key step in LSI. This decomposition is carried out by using a technique called Singular Value Decomposition or SVD (Golub & Kahan 1965, Golub & Van Loan 1989). A term-document $(m \times n)$ matrix A is decomposed

[3]Terms are words, phrases, or any other indexing units that can be used to identify the contents of a text.

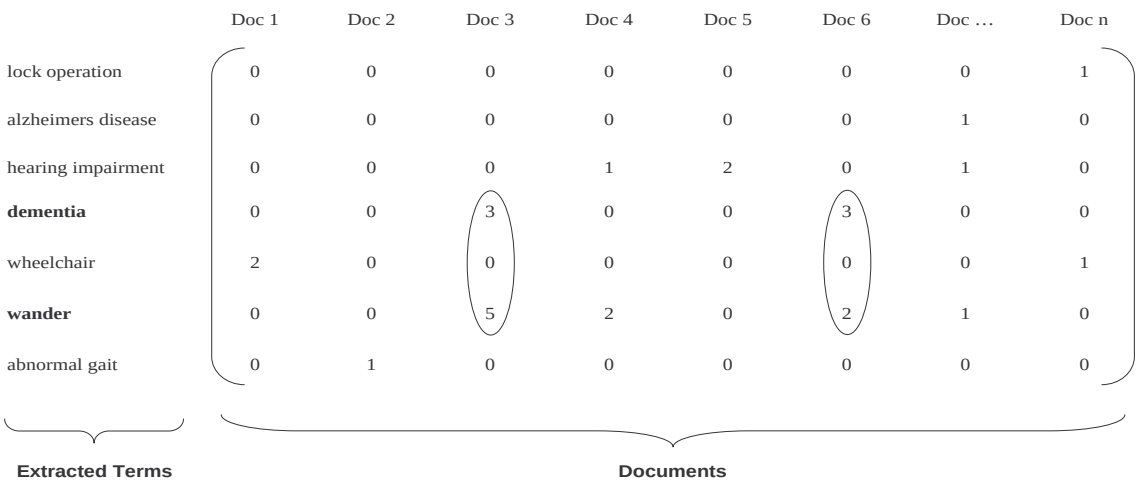


Figure 4.7: Term-Document Matrix

into the product of 3 matrices such that:

$$A_{(m\times n)} = U_{O_{(m\times d)}} \times S_{O_{(d\times d)}} \times V^T_{O_{(d\times n)}}$$

where, U_O and V_O are orthogonal matrices and S_O is a diagonal matrix whose elements are, by convention, all positive and arranged in descending magnitude (Deerwester et al. 1990). The columns of U_O and V_O are called the *left* and *right singular vectors* respectively, and the diagonal elements of S_O are called the *singular values*.

For each column U_i and V_i of U_O and V_O respectively, the relation $AV_i = S_iU_i$ holds. This shows that, each right singular vector is mapped by A onto the corresponding left singular vector with the corresponding singular value as the *multiplication* factor. The relation further shows that, each column in U_O corresponds to a singular value. Consequently, the columns in U_O are ranked according to their importance as reflected by their corresponding ranked singular values in S_O.

Each column in U_O represents a *topic* as a linear combination of all terms in the collection. Thus LSI finds common dimensions in which both the documents and terms can be expressed in the same space i.e., the *topic space*. We use the term *topics* to refer to *concepts* created by LSI so that they are not confused with other concepts in later parts of the thesis.

The top r largest singular values can be retained, the smaller ones are set to zero,

and the rows in V_O^T and columns in U_O which correspond to the small singular values are deleted. This process is essentially equivalent to retaining the important features (topics) that were latent in the document collection, and discarding the less influential ones. The process leaves an approximation $\tilde{A}$ of the original matrix A as:

$$\tilde{A}_{(m\times n)} = U_{(m\times r)} \times S_{(r\times r)} \times V^T_{(r\times n)} \tag{4.1}$$

$\tilde{A}$ is the matrix of rank r that best approximates the original matrix A (Golub & Loan 1996). Thus the r most important topics in U_O are identified by retaining the r highest singular values. This implies that these most important topics are represented by the $U_{(m\times r)}$ matrix shown shaded in Figure 4.8. $U_{(m\times r)}$ and $V_{(n\times r)}$ are also referred to as *truncated* term and document matrices respectively.

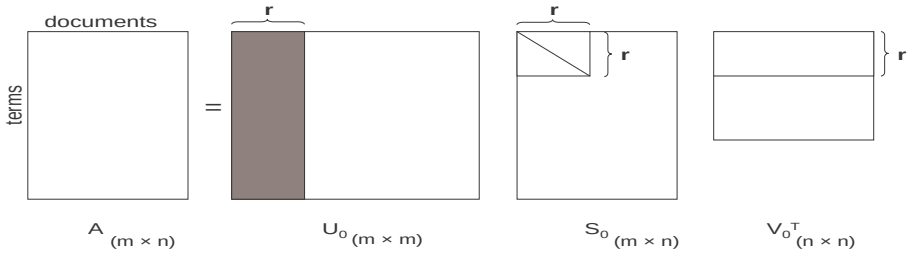


Figure 4.8: Singular Value Decomposition and Latent Semantic Indexing

The value of r should not be so small as to result in discarding of important features but it should be large enough to cover the important domain topics (Berry et al. 1995). The best choice for r is dependent on the document collection. The appropriate way to determine r is still an open research problem and thus r is typically determined empirically. For the SmartHouse domain, there are 38 documents and therefore 38 topics. Out of the 38 topics, the anchors are found to have more significant weights in the top 9 topics than in the rest. This is why the top ($r = 9$) singular values and hence the nine most important topics are utilised. Two important term characteristics, *similarity* and *importance*, can be derived from the matrix approximations obtained above. These are used to identify key phrases and are briefly described below.

Term Similarity

Let T be a term-term square matrix whose entries reflect similarities between the corresponding terms. These entries also reflect co-occurrences of the terms in the different documents. T can be obtained by making use of the term-document matrix A:

$$T = AA^T \tag{4.2}$$

Using the SVD-decomposed version of A, then

$$\begin{aligned} T &= USV^T(USV^T)^T \\ &= US\underbrace{V^TV}_{I}SU^T \\ &= US^2U^T \\ &= (US)(US)^T \end{aligned}$$

When the truncated versions of the matrices are used, T_r is the truncated term-term similarity matrix:

$$T_r = (U_{(m\times r)}S_{(r\times r)})(U_{(m\times r)}S_{(r\times r)})^T \tag{4.3}$$

This matrix reflects co-occurrence patterns of the first and higher orders. Higher order co-occurrences are a result of transitive relationships between terms. Two terms which do not co-occur in a single document could be highly related by separately occurring with the same terms. Kontostathis & Pottenger (2002) show that a connectivity path (and therefore a relationship) exists for every non-zero element in the truncated matrix. The longer the path, the higher the order of co-occurance, and the weaker the similarity relationship between the terms in question.

Term Importance

The entries in the $U_{(m\times r)}$ matrix are importance scores of each term in every key topic. The singular values in $S_{(r\times r)}$ are an indication of the importance of each corresponding topic in $U_{(m\times r)}$. This knowledge of the importance of each key topic in the *whole document collection* can help to compare the importance scores of each term over the entire collection.

This is done by multiplying $U_{(m\times r)}$ by $S_{(r\times r)}$ which leads to the accentuation of the entries in $U_{(m\times r)}$. So the weights of terms in the accentuated $U_{(m\times r)}$ matrix are now a measure of the *importance* of the individual terms to the key topics with respect to the *whole document collection*.

4.3.2 Obtaining the Term-Document Matrix

This section details the process of determining term weights in order to create the term-document matrix. The documents are represented as an incidence term-document matrix A for the extracted terms. An entry a_{ij} is the product of three weights:

- a local weighting of a term i in document j. The local weighting scheme shows the importance of the term in a particular document and reflects the extent to which the term is representative of the document's content.
- a global weighting of the term i. The global weighting scheme shows a term's importance in the entire collection.
- normalisation factor for document j. The normalisation factor is intended to ensure that terms in longer documents are not deemed to be necessarily more important than those in shorter documents and vice versa. This is because terms in longer documents are more likely to occur with a higher frequency than those in shorter documents. Consequently, these terms might be seen to be more important because of their frequency of occurrence. On the other hand, terms that occur in shorter documents might be given a higher weighting over those that occur in longer documents.

Studies to examine the efficiency of various term weighting schemes found the logarithmic frequency local weighting and the logarithmic entropy global weighting to give relatively high efficiencies for retrieval tasks (Salton & Buckley 1988). The SmartHouse reports cover a wide range of problem-areas such that important terms may or may not occur with high frequency. The log function dampens the effects of the high frequency terms, and thus ensures that low frequency terms are not ignored. Consequently, the log

frequency is used as the local weighting scheme:

$$t_{ij} = \log_2(f_{ij} + 1) \tag{4.4}$$

where f_{ij} is the frequency of term i in document j. Similarly, a log entropy weighting is used as the global weighting scheme:

$$g_i = 1 - \sum_{k=1}^{N} \frac{P_{ik} \log_2(P_{ik})}{\log_2(N)} \tag{4.5}$$

where P_{ik} is the relative frequency of term i in document k, compared to the collection of N documents. The document vectors are also normalised by employing the cosine normalisation factor (Salton & Buckley 1988) to each document j:

$$\text{Cosine Normalisation} = \sqrt{\sum_{i=1}^{n} (g_i t_{ij})^2} \tag{4.6}$$

where g_i is the global weighting of the ith term, t_{ij} is its local weight in the document j, and n is the number of terms in the document. Thus each term a_{ij} in the incidence matrix has a weight given by:

$$a_{ij} = \frac{\log_2(f_{ij} + 1)(1 - \sum_{k=1}^{N} \frac{P_{ik} \log_2(P_{ik})}{\log_2(N)})}{\sqrt{\sum_{t=1}^{n} (\log_2(f_{ij} + 1)(1 - \sum_{k=1}^{N} \frac{P_{ik} \log_2(P_{ik})}{\log_2(N)}))^2}} \tag{4.7}$$

4.3.3 Identifying non-anchor Key Problem-descriptors Using Term Similarity

Anchor terms are key phrases that are exploited in order to identify further key problem-descriptors. Terms that are "*similar*" to anchor terms are also regarded as key since anchor terms are key. These terms' similarity with anchor terms will be above some similarity threshold. Terms are expected to have their highest similarity scores with themselves i.e., a term is expected to be more similar to itself than to any other term. Thus, an anchor term's similarity score with itself is used as a basis for setting the threshold for determining if other terms are similar to it. A term is deemed as key if it is "sufficiently" similar to one of its document's anchor terms.

A document will have one or more anchor terms. When the anchor set contains

one element, the threshold is based on the score for that anchor term. In instances where the anchor set has more than one element, the anchor terms are likely to have different similarity scores with themselves. In these circumstances, the threshold is based on the average similarity score. The document in Figure 4.6 (Page 63) has the anchor set {*alzheimers disease*, *dementia*} as illustrated in sub-section 4.2.3. Figure 4.9 shows the terms identified for the *wandering* sub-problem of Figure 4.6, for different threshold values. Additional terms that are identified whenever the threshold is reduced are shown in italics. The thresholds are a percentage of the average of the anchor similarity scores.

10%	**20%**	**40%**	**50%**	**60%**
wandering leave the home home and wander tendency leave home wander night *husband* *open* *door*	wandering leave the home *home and wander* tendency leave home *wander* night	wandering leave the home tendency *leave* home night	wandering leave the home tendency home night	wandering leave the home tendency home night

Figure 4.9: Comparison of Similarity Thresholds

As Figure 4.9 shows, a low threshold results in inclusion of terms that are not relevant and a high one has the consequence of ignoring important terms. Indeed, no important terms are identified for the thresholds 40%, 30% and 20%, and at 10%, some important terms such as *open* and *door* are identified but at that threshold, unimportant terms such as *tendency* and *home and wander* are also included. This trend was observed for most of the documents and it can be attributed to the low similarity scores obtained by the anchor terms. A threshold of 50% of the average similarity score was found to result in identification of reasonable key phrases. In instances where the document has one anchor term, the threshold was lowered to 40%.

Suppose that we want to determine if the term *wander* in Figure 4.6 is key. This is achieved by looking up *wander*'s similarity to the anchor terms using the similarity

matrix, a portion of which is shown in Figure 4.10. This figure illustrates that similarity between *dementia* and itself is *5.78*. Similarity between *alzheimers disease* and itself is *3.34*. *Wander* will be deemed as key if it is found to be "sufficiently" similar to either *dementia* or *alzheimers disease*. This means that *wander* will only be regarded as key if its similarity with either anchor term is at least equal to *2.28* (50% of the average of 5.78 and 3.34). Figure 4.10 shows that similarity between *wander* and *dementia* is *6.60* and that between *wander* and *alzheimers disease* is *5.02*. These similarity scores are higher than the set threshold (*2.28*) and so *wander* is deemed as key. It is also noteworthy that although *dementia* and *alzheimers disease* may not occur in the same documents as shown in the term-document matrix in Figure 4.7, the LSI-generated similarity matrix shows them as related through their sharing of common terms (resulting in a similarity score of 4.06).

	lock operation	**alzheimers disease**	hear problem	**dementia**	wheelchair	***wander***	abnormal gait
lock operation	9.40	-0.09	-0.16	-0.07	3.93	-0.08	0.05
alzheimers disease	-0.09	3.34	0.67	4.06	0.20	5.02	-0.50
hearing impairment	-0.16	0.67	13.0	-0.10	0.56	0.38	1.22
dementia	-0.07	4.06	-0.10	5.78	0.70	6.60	0.25
wheelchair	3.93	0.20	0.56	0.70	8.50	0.75	0.20
wander	-0.08	5.02	0.04	6.60	0.08	0.79	0.12
abnormal gait	0.05	-0.50	1.22	0.25	0.02	0.12	0.94

Extracted Terms **Extracted Terms**

Figure 4.10: Term-Term Similarity Matrix

4.3.4 Identifying non-anchor Key Problem-descriptors Using Key Topics

Terms that are important with respect to given topics will occur in documents in which the topics are described. These key terms will occur with a relatively high level of importance in these topics. This means that if we can identify some terms that we know to be important in *key* domain topics, other terms that occur with similar degrees of importance, in the *same* topics, can also be assumed to be key to the domain.

We have identified key topics and the importance scores of each extracted term in every key topic. We have also obtained the accentuated $U_{(m\times r)}$ matrix which allows us to compare importance scores across the document collection. We have identified anchor terms and we know them to be key and so can now use them to identify further terms that are also key. The assumption is that

> *If a term* **t** *is important in the domain, terms that are important in key topics where* **t** *is important, will invariably be key to the domain.*

So what this technique does is deem as key, terms that are important in key topics where their anchor terms are found to be important. The term *wander*(Figure 4.6) will be used again to illustrate the process.

For each of the members of Figure 4.6's anchor set {*Alzheimers disease*, *dementia*}, we find topics in which they have their highest importance scores. In this example, we find a topic in the accentuated $U_{(m\times r)}$ matrix, in which *alzheimers disease*, has the highest importance score. The portion of the accentuated $U_{(m\times r)}$ matrix in Figure 4.11 shows *alzheimers disease* as having its highest importance score (*7.29*) in topic 3. Similarly, dementia's highest score (*8.02*) is also found in topic 3. This anchor set has more than one member and so the threshold becomes the average importance score for the anchor set which in this instance is *7.66*.

Figure 4.12 shows the terms identified for the *wandering* sub-problem of Figure 4.6, for different threshold values. Additional terms that are identified whenever the threshold is reduced are shown in italics. 90% of the average importance score was generally found to result in the identification of reasonably important phrases without including too many unimportant terms. Therefore, Figure 4.6 terms with importance scores that are equal or above *6.9* (90% of 7.66) will be regarded as key. *Wander*'s importance score of *7.48* (in topic 3) is above this threshold and consequently, it will be identified as key. The effect is that terms that are nearly as important as their anchors, are identified as key.

4.3.5 Term Filtering

Terms that are substrings of other terms in the **same** sentence are discarded. This is to reduce redundancy in the data, but at the same time ensure that short terms are not

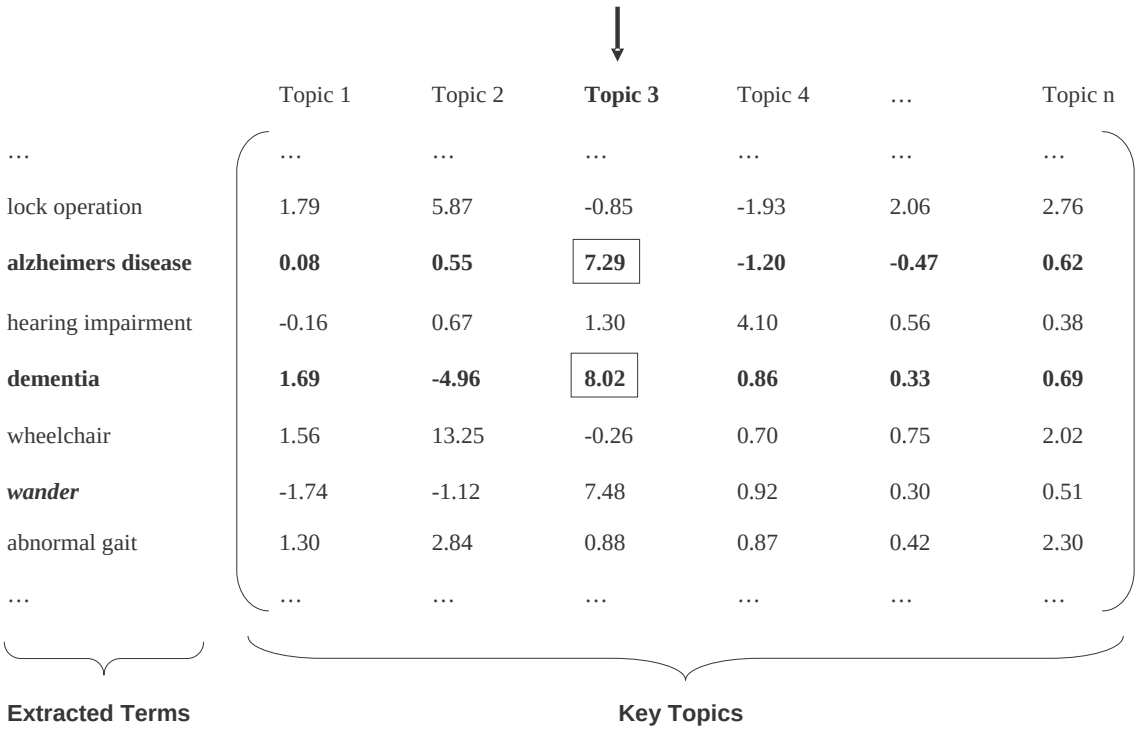


Figure 4.11: Accentuated $U_{(m\times r)}$ Matrix Showing Term Importance for Key Topics

discarded just because they happen to be sub-strings of terms that appear in other parts of the document. While limiting the incorporation of single-word terms, the process ensures that each key phrase appears by itself or as part of a longer phrase. The assumption here is that in any given sentence, a single word is unlikely to occur independently of short phrases in which it occurs. For example in the sentence "*Case knowledge is a key knowledge source in case based reasoning*", phrases like "*case knowledge*" and "*case based reasoning*" will be retained but the single word "*case*" will be discarded since it is a substring of the two phrases and can therefore be assumed not to occur independently of the two phrases. However, "*case*" will be retained if it appears in a sentence like "*A case is made up of a problem and solution part*". This process also concatenates those key phrases that are adjacent to each other in a sentence. This allows the information to be more complete. Shared substrings not at sentence level are not discarded for we believe that, if a two or more word phrase appears in more than one sentence or document, then it is potentially important.

20%	30%	50%	80%	90%
wandering				
tendency to leave				
leave the home				
home and wander				
leave				
home				
wander				
wander away				
away				
night	wandering	wandering	wandering	
husband	leave the home	leave the home	leave the home	
thin nightgown	home and wander	home and wander	*home and wander*	wandering
dress	leave	leave	*leave*	leave the home
thin	home	home	home	home
nightgown	wander	wander	wander	wander
open the side	away	away	away	away
side door	night	night	night	night
open	husband	husband	husband	husband
door	open	open	open	open
neighbor	door	door	door	door

Figure 4.12: Comparison of Importance Thresholds

4.4 Discussion

Figure 4.13 shows the evolution of the key phrase extraction process for the *wandering* sub-problem in Figure 4.6. Figure 4.13(c) shows terms which were found to be key by using the similarity scores of the anchor terms. The terms in Figure 4.13(d) are key phrases obtained using the anchors' importance scores.

A sample of the discovered key phrases is compared to text where an expert was asked to highlight key phrases. Figure 4.14 shows (lemmatised) text highlighted in bold by the expert and by LSI's importance and similarity term characteristics respectively, for one sub-problem. The different n-grams are underlined in the latter instances. Generally, the text highlighted using LSI compares very well with that highlighted by the expert. Although the expert does not highlight the heading *door open*, it clearly is an important term since both the expert and LSI agree that *difficult to open the door* is important in that context. It is not possible for LSI to highlight the whole term *poor flexibility in the*

(a) Lemmatised Text

wandering mrs m had a tendency to leave the house and wander off. she had recently wander away from the house in the middle of the night. her husband had found her outside, dress only in a thin nightgown, try to open the side door to a neighbor home...

(b) Extracted terms

wandering, tendency to leave, leave the home, home and wander, tendency, leave, home, wander, wander away, away, night, husband, thin nightgown, dress, thin, nightgown, open the side, side door, open, side, door, neighbor

(c) Key Phrases Using Term Importance

wandering, leave the home, home, wander, away, night, husband, open, door

(d) Key Phrases Using Term Similarity

wandering, leave the home, tendency, home, night

Figure 4.13: The Process of Identifying Key Phrases

joint as important since the terms it is presented with are not more than 3 words long (trigrams). However, it highlights the important parts of the term. In this example text, term importance identifies key phrases that are closer to Expert-highlighted phrases, than term similarity.

4.5 Summary

This chapter has described the process of extracting data that is embedded in the sections of semi-structured textual documents. A set of anchor terms were identified and used to highlight other key problem-descriptor terms in the documents. This was achieved with the aid of Latent Semantic Indexing which helped to identify useful features in the data, out of which further key problem-descriptors were identified. Using term importance yields problem-descriptors that are closer to those highlighted by the expert than when term similarity is employed. Indeed, when importance thresholds were utilised in key phrase identification, there was a distinct demarcation between thresholds that result in the identification of a reasonable number of important terms without including too many unimportant terms, and those that include many of the unimportant terms. On the other hand, when term similarity was employed, many important terms were not identified and

Expert's Highlighting

door open
when mr. M wish to open the front door, he had to project himself forward to ***reach the door handle and lock****. because of* ***poor flexibility in the joint****, he found this* ***task to be extremely difficult and physically tiring****. also, the* ***position of the lock*** *made it both* ***difficult to open the door*** *from the inside upon leave, or from the outside when return home.*

LSI Highlighting Using Term Similarity

door open
when mr. M wish to open the front door, he had to ***project*** *himself forward to* ***reach the door*** ***handle and lock****. because of* ***poor flexibility*** *in the* ***joint****, he found this task to be extremely* ***difficult and physically tiring****. also, the* ***position*** *of the* ***lock*** *made it both* ***difficult to open the door*** *from the* ***inside upon leave****, or from the outside when* ***return home****.*

LSI Highlighting Using Term Importance

door open
when mr. M wish to open the front door, he had to project himself forward to ***reach the door*** ***handle and lock****. because of* ***poor flexibility*** *in the* ***joint****, he found this task to be extremely* ***difficult and physically tiring****. also, the* ***position*** *of the* ***lock*** *made it both* ***difficult to open the door*** *from the inside upon leave, or from the outside when return home.*

Figure 4.14: Text Highlighted by Expert and LSI

at low thresholds when they were (identified), many other unimportant terms were also identified as important. It is for this reason that terms generated using the importance technique are used instead of those obtained by using the similarity scores. In the next chapter, all knowledge pertaining to each sub-problem is included in the sub-problem's representation as illustrated in the architecture (Figure 4.15) of the case authoring task. A sub-problem's representation comprises key phrases identified in the sub-problem's text and relevant knowledge from other parts of the report such as the summary. The sub-problem's representations are then used to create conceptual models of the data.

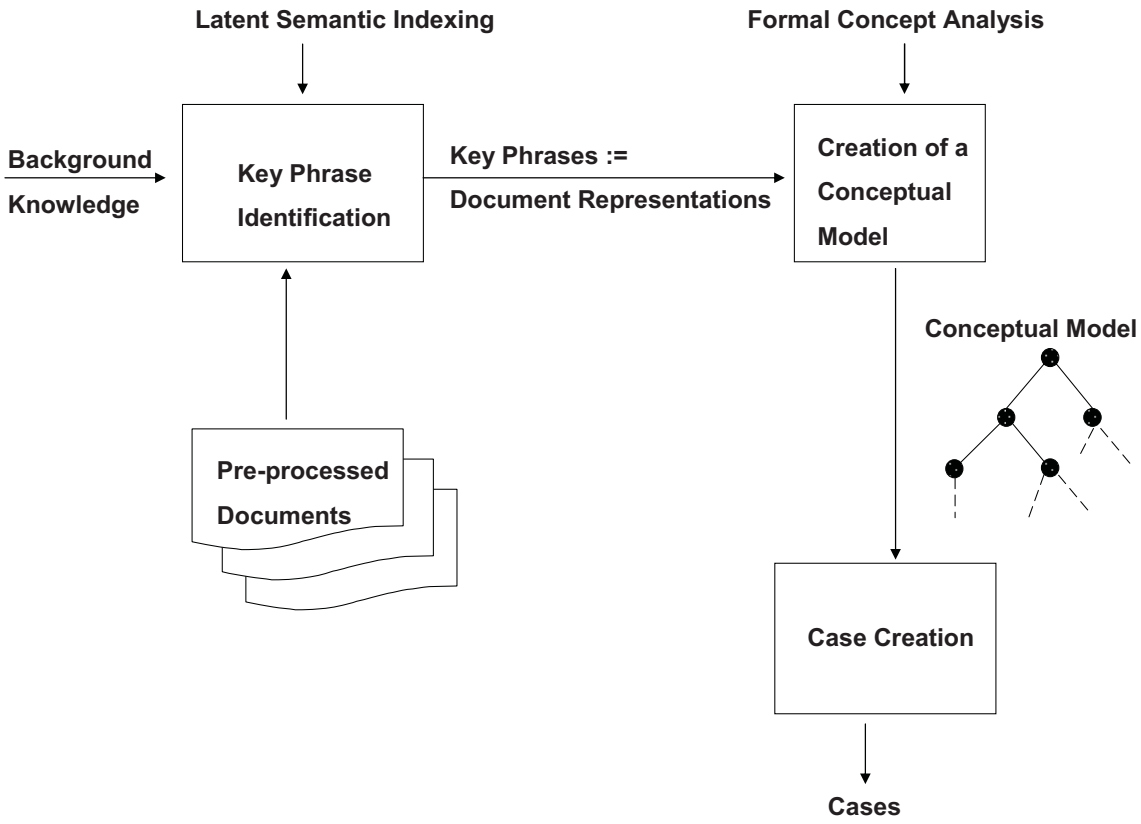


Figure 4.15: Architecture of the Case Authoring Process

Chapter 5

Using Formal Concept Analysis to Create Conceptual Models

In the previous chapter, terms were extracted from which key phrases were identified. The key phrases comprise disability-terms, disease-terms, and other key problem-descriptor-terms that were discovered by using background knowledge and Latent Semantic Indexing. A number of researchers such as Maddi, Velvadapu, Srivastava & de Lamadrid (2001) have used LSI to create ontologies and other conceptual structures. However, the nature of relationships between LSI-produced concepts is still uncertain. Humans interpret text by manipulating concepts at various levels of abstraction (Gabrilovich & Markovitch 2007). For example, an occupational therapist that reads about a *wheelchair user* will associate the person with a *mobility problem*. Consequently, this project is interested in mimicking the way humans interpret text in order to reasonably interpret text for case authoring and query interpretation during problem-solving. This is carried out by organising the concepts in the data according to their concept-superconcept relationships. In this work, Formal Concept Analysis (FCA) is employed to create conceptual models of the data extracted in Chapter 4. Furthermore, typically during problem-solving, knowledge of the new problem is presented and it is this knowledge that is used to find a solution. Thus problem-solving depends on knowledge of the current problem in relation to the domain problem space. Therefore, this project builds conceptual models of the problem space because it is these that will be useful for retrieving similar past cases and for query interpretation.

Each document contains a summary in addition to one or more sub-problems and their corresponding solutions as depicted in Figure 5.1. In earlier work in the SmartHouse domain by Wiratunga, Craw, Taylor & Davis (2004), case decomposition into smaller cases proved to be effective in obtaining realistic solutions as judged by the domain expert. SmartHouse device recommendation for a particular difficulty is dependent on the person's disabilities but generally independent of any other difficulties the person may be facing. For example, recommendation of a *door opening* device (in response to a door opening difficulty) will depend on the reason the person is having the difficulty but is unlikely to be influenced by a *window opening* difficulty the same person may be facing. Thus decomposing the problem space of each document into its different sub-problem elements and authoring these as fully-fledged cases does not lead to information loss regarding interactions between the different sub-problems because these are generally not important.

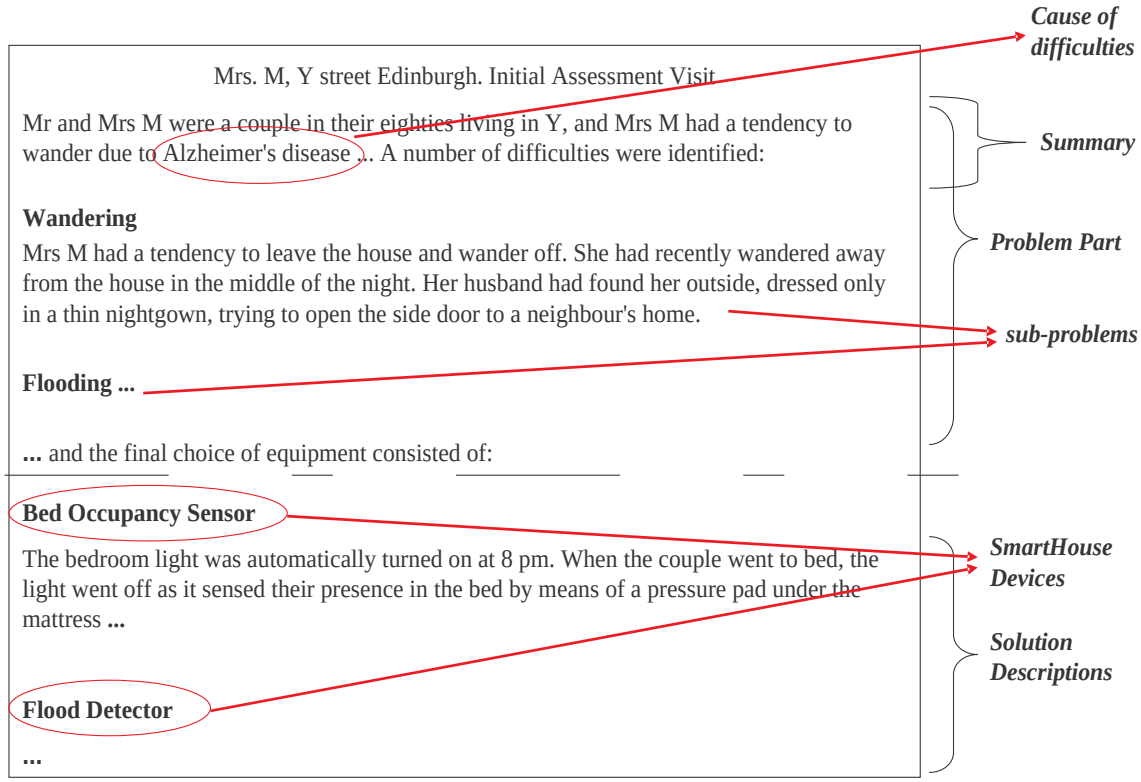


Figure 5.1: SmartHouse Report

Instead, when cases are authored from sub-problems, retrieval will also focus on solutions to a particular difficulty as opposed to a set of solutions that solve many, but different, difficulties. This in turn, reduces the amount of effort required to adopt a so-

lution to the new problem. Different (previous) cases can then be used to solve different parts of a new problem. When recommending solutions for a client with multiple difficulties, the different solutions can be merged if necessary once the correct individual solutions to each sub-problem have been established. Consequently, in this work, each document is decomposed into its sub-problems and the sub-problem's representations are then used to create conceptual models, and authored as problem parts of cases.

This chapter gives a description of FCA and how it is applied to sub-problems' representations in order to create the conceptual models. Sub-problem representations are obtained in Section 5.1. A description of FCA and how it can be used to create conceptual models is given in Section 5.2. A case comprises a problem description part, a solution part, outcomes, etc (Kolodner 1993a). Thus case authoring also involves determining the solution description corresponding to each problem part. Section 5.3 presents the mapping of each sub-problem onto a description of a SmartHouse device that helps with that sub-problem. These mappings will be used to attach authored problems to their corresponding SmartHouse solutions in order to complete the case descriptions.

5.1 Obtaining Sub-problem Representations

A sub-problem is essentially a description of a difficulty a disabled person is having in carrying out a particular task, or a risk they are facing as a result of their disability. Consequently, each sub-problem's representation should reflect the underlying cause of the difficulty if it is to be useful for problem-solving when authored as the problem part of a fully-fledged case. This is achieved by including the sub-problem's disability-term in the sub-problem's representation. It will be important to include only the underlying disabilities in instances where the document can be mapped onto more than one disability-term. Take an example of a person with *learning difficulties* who also happens to have a *hearing problem*. In the sub-problem describing this person's inability to use their telephone because they are *unable to hear the caller*, it would be inappropriate to add the *learning difficulties* disability to this sub-problem's representation since the problem being described is due to their *hearing problem* unless *learning difficulties* is also specifically mentioned as a reason behind the difficulty described in the sub-problem.

Therefore, each sub-problem needs to be assigned its appropriate set of disability-terms, from all those discovered in Chapter 4 (sub-section 4.2.1 Page 59). A document will typically have disability-terms that have been explicitly stated and in addition, may have implicitly stated disability-terms that are obtained by making use of the disability look-up table (sub-section 4.2.2 Page 61). Consider the report excerpt shown in Figure 5.1. The document does not have a disability-term that is explicitly stated because *Alzheimer's disease* is regarded as a disease-term. However, the terms *wander* and *Alzheimer's disease* will each return the disability-term *dementia* when used to search for disability-terms in the disability look-up table. Thus *dementia* is the disability-term that (implicitly) pertains to this document. The disability-terms can come from the summary or any part of the sub-problem's description text. Thus the task of assigning a list of disability-terms to each sub-problem will involve choosing one or more disability-terms from the list that pertains to the whole document.

The system aims to identify one or more implicit and explicit disability-terms as pertaining to each sub-problem. Such disability-terms are referred to as *selected* by that sub-problem. Assigning sub-problems to appropriate disability-terms is carried out in 2 steps which are illustrated in Figures 5.2 and 5.3.

Step 1: A disability-term that explicitly occurs in the sub-problem's text is assumed to be responsible for that difficulty. Consequently, the disability-term is identified as *selected* by that sub-problem and the algorithm terminates.

In instances where no disability-term explicitly occurs in the sub-problem's text, each term in the sub-problem is used to search for implicit disability-terms in the look-up table. All returned disability-terms that are also in the list already identified for the whole document are tested for support. The rest are discarded. A disability-term is only *selected* if it is identified by at least two-thirds of the terms that can look up disability terms in that sub-problem. A record is made of the selected disability-term and the number of sub-problems in the document that have selected it.

Step 2: The second step targets sub-problems whose terms are inadequate in determining the disability-term that pertains to them. Sub-problems that are able to select a disability-term (i.e., in the first step), vote for their disability-term. A disability-

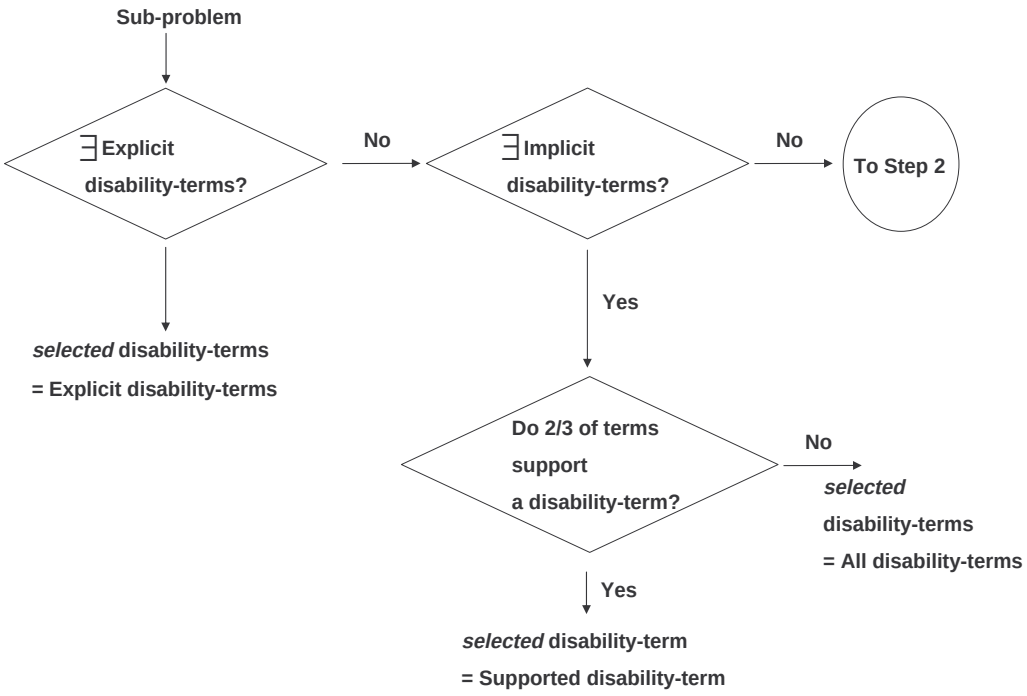


Figure 5.2: Subproblem-Disability Mapping; Step 1

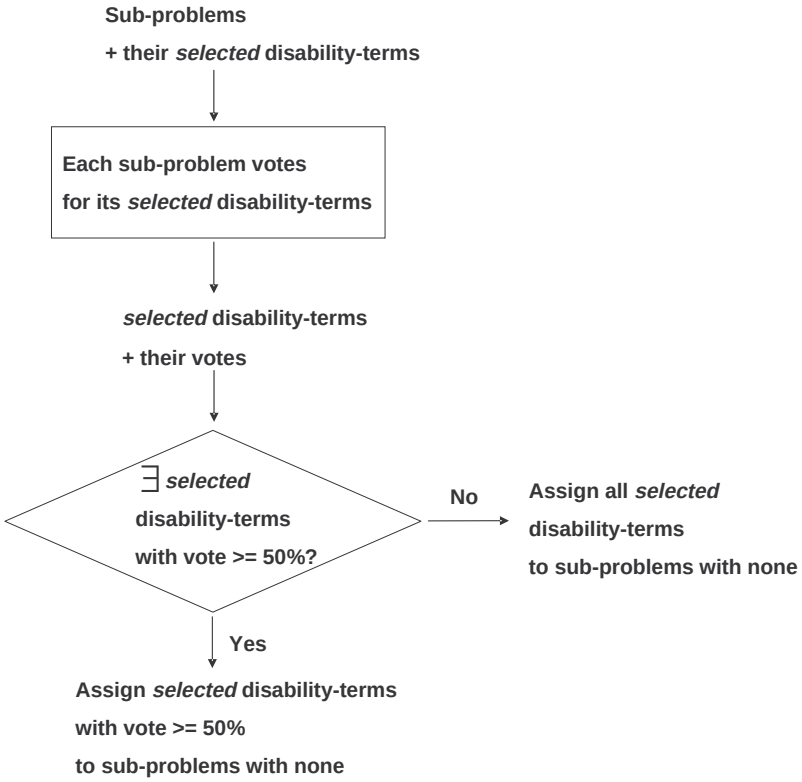


Figure 5.3: Subproblem-Disability Mapping; Step 2

term that obtains at least 50% of the vote from *all* the document's sub-problems is assigned to sub-problems that are unable to select a disability-term on their own. The vote obtained by a selected disability-term is given by:

$$\text{vote} = \frac{\text{Number of Selecting Sub-problems}}{\text{Total Number of Sub-problems in the Document}} \times 100\%$$

Sub-problems that are unable to select a disability-term do not vote. This implies that the highest vote any disability-term can attain will be less than 100% of the total vote. Thus the requirement that a disability-term obtains 50% of the total vote before it is assigned to a sub-problem ensures that the sub-problem is assigned the most common disability-terms. If the majority vote comes from less than half of the sub-problems, the sub-problem is assigned *all* the disability-terms pertaining to all the other sub-problems because there is not enough evidence to assign a particular disability-term to the sub-problem.

Each sub-problem is now represented by its key phrases which will comprise key problem-descriptors that are **not** disability or disease-terms, disability-terms that were explicitly and implicitly mentioned in the sub-problem, and all disease-terms that are mentioned in the sub-problem's parent document. Figure 5.4(a) shows key phrases obtained using the *importance* (Chapter 4 Subsection 4.3.4) scores of anchor terms for the document shown in Figure 5.1. Figure 5.4(b) illustrates the representation for the *wandering* sub-problem; *alzheimers disease* is the disease-term that was explicitly stated in the document and *dementia* is the implicit disability-term discovered by using the disability look-up table.

(a) Key Phrases Using Term Importance

wandering, leave the home, home, wander, away, night, husband, open, door

(b) Sub-problem Representation

dementia, alzheimers disease, wandering, leave the home, home, wander, away, night, husband, open, door

Figure 5.4: Enriching Representations with Implicit Disability-terms

The sub-problem representations are utilised in obtaining domain concepts and organising the concepts into a hierarchy using Formal Concept Analysis. The next Section gives

a description of Formal Concept Analysis and how it can be used to obtain a concept hierarchy of the domain data.

5.2 Formal Concept Analysis

Formal Concept Analysis is a method that is predominantly used to investigate and process information that is explicitly given (Wille 1997, Petersen 2004, Priss 2006). FCA produces a concept lattice in which the original data is structured into formal abstractions of concepts that are comprehensible and as such, can be meaningfully interpreted. In FCA, a formal context is a triple (O, A, I) where O is a set of objects, A a set of attributes and $I \subseteq O \times A$ is a binary incidence relation between O and A. I indicates which objects have which attributes.

5.2.1 Formal Context

FCA uses a formal context to identify formal concepts. A formal context is often represented as a table where a cross indicates an attribute exhibited by the corresponding object. Figure 5.5 is an example of a formal context for the SmartHouse domain. The different SmartHouse sub-problems form the set of objects, and the terms used to represent the sub-problems form the set of attributes. The sub-problems in the context table have been tagged with an identifier for that difficulty for ease of understanding. For example *telephone operation* is the identifier for a sub-problem object that has an attribute *hearing impairment* (arising from a person with a *hearing impairment* who has difficulties operating their telephone).

	ATTRIBUTES								
OBJECTS	hearing impairment	wheelchair	mobility problem	poor flexibility in the joint	unable to stretch	cerebral palsy	lack of strength in hands	multiple sclerosis	
intercom operation		X	X		X	X			
telephone operation	X								
window operation			X				X	X	
door opening			X	X					

Figure 5.5: Context for some SmartHouse Sub-problems

5.2.2 Identifying Objects and Attributes

Each FCA object and associated attributes result in a concept and will also result in superconcepts if the object has common attributes with other objects. Each sub-problem of a SmartHouse report represents a specific need for a set of SmartHouse devices. We want to create a hierarchy of concepts pertaining to people's areas of difficulty which are described in the sub-problems. It is for this reason that each sub-problem is used as an FCA object.

Each sub-problem is used as an FCA object and it follows that the terms describing the objects are used as FCA attributes. Thus the sub-problem representative terms identified in Section 5.1, become the attributes for the sub-problem FCA objects.

5.2.3 Formal Concepts

A concept is a pair $(o \subseteq O, a \subseteq A)$ such that every object in o is described by every attribute in a and conversely, every attribute in a covers every object in o. For instance in Figure 5.5, the set of objects {*intercom operation*, *window operation*, *door operation*} have the set of attributes {*mobility problem*} in common. Conversely, the set of attributes {*mobility problem*} shares a common set {*intercom operation*, *window operation*, *door operation*} of objects to which they belong. No other object has this set of attributes.

In this work, the Concept Explorer tool, ConExp[1] was used to create the concept hierarchies. ConExp's lattice viewer has a concept node's upper semi-circle filled with colour if an attribute is attached to the node in the reduced labeling. Correspondingly, if an object is attached to a concept node, then the lower semi-circle is filled. Attached attributes are shown as shaded rectangles and attached objects are shown as unshaded.

The concept hierarchy resulting from the context in Figure 5.5 is shown in Figure 5.6. Every node represents a concept and the nodes are ordered by a concept-subconcept relationship. The highest node represents the most general concept while the lowest one represents the most specific concept. Therefore, as you descend the hierarchy, and become more specific (less general), the number of objects at a node reduces while the number of attributes increases. Conversely, the number of objects increases and the number of

[1]ConExp-1.2 is available from http://sourceforge.net/projects/conexp

attributes reduces as one ascends the hierarchy.

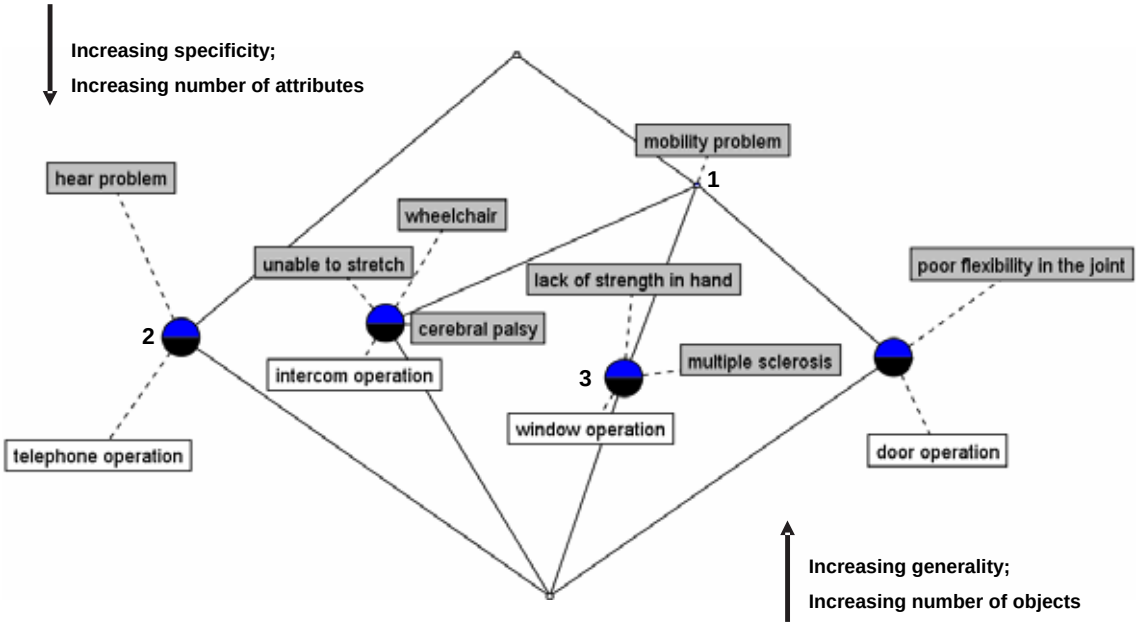


Figure 5.6: Example Hierarchy

To prevent cluttering, reduced labeling is used. An attribute is attached to the top-most concept that has the attribute in its list of attributes. Thus the attribute occurs in all attribute lists of concepts that are reachable by descending the subtree from which it is attached; for example node 3 represents a concept whose attribute set is {*mobility problem*, *multiple sclerosis*, *lack of strength in hand*}. Conversely, an object is attached to the bottom-most concept where it is part of the object list. Every concept that is reachable by ascending from this point to the top-most concept has the object in its object list; for instance node 1 represents a concept whose object set is {*intercom operation*, *window operation*, *door operation*}.

The objects associated with a concept are called its *extent*, and the attributes describing the concept are called its *intent*. The node labeled 1 is the concept created as a result of the object set {*intercom operation*, *window operation*, *door operation*} having a common set of attributes {*mobility problem*} and the set of attributes {*mobility problem*} covering a common set of objects {*intercom operation*, *window operation*, *door operation*}. This concept has intent {*mobility problem*} and extent {*intercom operation*, *window operation*, *door operation*}. Similarly, the concept shown as node 2 has intent {*hearing impairment*}

and extent {*telephone operation*}. The topmost node or the root represents a concept whose attributes are features of *all* the objects in the context; thus its intent is an empty set and extent is the set of all the objects. Conversely, the bottom most node represents a concept whose objects have *all* the attributes in the context; thus this node's intent is the set of all attributes and the extent is an empty set. In the SmartHouse domain, the extent of the bottom most concept would comprise sub-problems that are represented by ALL the key phrases in the domain, a feature that is essentially impossible to find. Consequently, for purposes of clarity, this concept node is excluded from subsequent Figures depicting concept hierarchies.

A case requires a solution description part, in addition to a problem description part (Kolodner 1993a). Thus case authoring also involves determining the solution description corresponding to each problem part. Section 5.3 presents the mapping of each sub-problem onto a description of a SmartHouse device that helps with that sub-problem. These mappings will be used to attach authored problems to their corresponding SmartHouse solutions in order to complete the case descriptions.

5.3 Matching Sub-problem Text to SmartHouse Solutions

Each sub-problem has a matching set of SmartHouse devices that help with the difficulty. A report typically contains a number of sub-problem descriptions and corresponding descriptions of SmartHouse solutions. Consider the report excerpt shown again in Figure 5.7 showing a *wandering* sub-problem and a *bed occupancy sensor* as the device that helps with that sub-problem. Thus although it is not obvious, each sub-problem can be mapped onto a list of corresponding SmartHouse device descriptions. A human performing the task of mapping SmartHouse solutions to sub-problems would rely on terms in the solution description, that refer to the difficulty described in the sub-problem. Automatically mapping sub-problems to their solutions mimics this behaviour by relying on overlaps between sub-problem and solution terms.

The aim of the mapping task is to get a solution description that best matches a problem description in order to attach the solution to an authored problem thus completing the case description. The mapping task relies on the assumption that some of the words

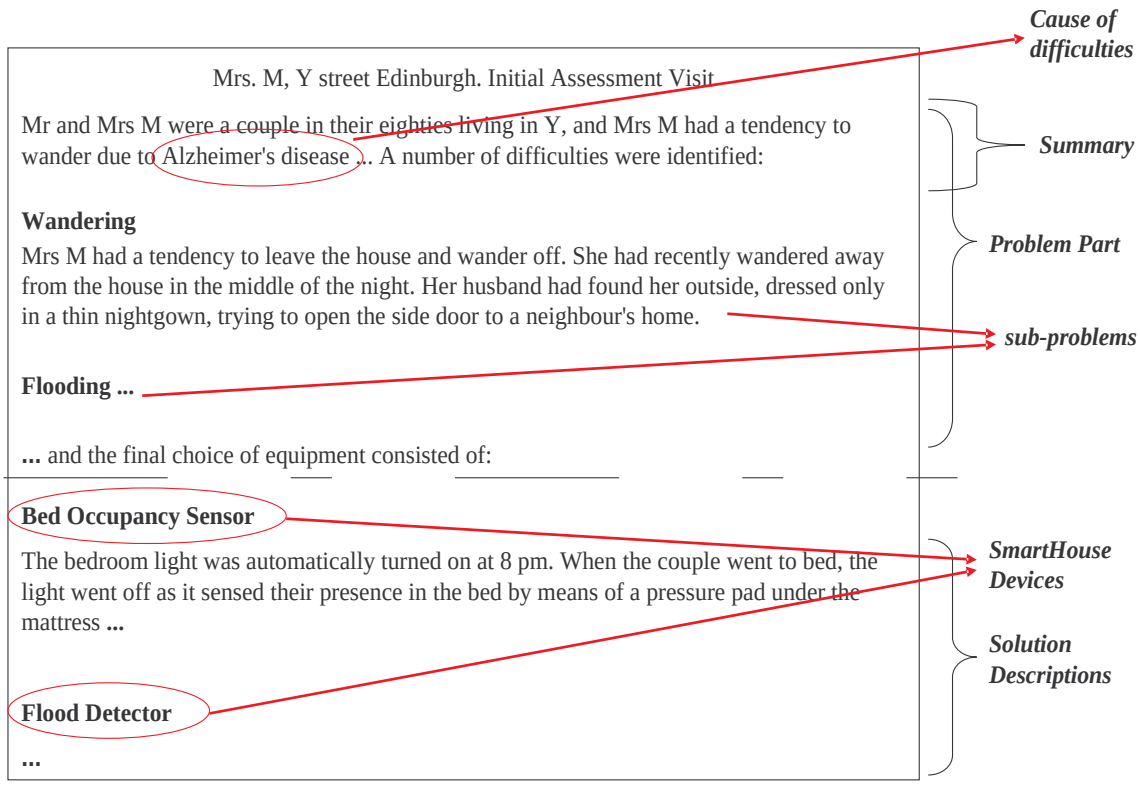


Figure 5.7: SmartHouse Report

or phrases pertaining to a sub-problem will be repeated in the description of its solution. When looking for a solution description that has something in common with the sub-problem, one is essentially asking, "*how much of the sub-problem is reflected in the solution description?*" Therefore, the proportion of terms in the sub-problem description that can be found in the solution description is used as a measure of relatedness. These terms can comprise one or more words. It is more likely for the sub-problem and solution text to have single words in common, than longer phrases. Thus two measures are utilised; one based on common single words and the other, on common phrases. It has also been observed that the name of the solution can sometimes give insight into the type of problem it is meant to solve. For example one can 'guess' with a good degree of certainty that *door opener* is a solution to a *door opening* problem. Thus matching phrases is also likely to capture these device names. Mapping is carried out for each sub-problem description in turn. The overall value of the weighting is obtained by aggregating evidence from the two measures using Mycin's approach to combining evidence (Davis, Bruchanan & Shortliffe 1985).

5.3.1 The Sub-Problem to Solution Mapping Algorithm

Mapping of sub-problems to solutions is carried out as follows:

1. Let $T_n(D)$ be the set of n-grams in document D, excluding ones that start or finish with a stopword.

2. For each sub-problem description P and solution description S, the bigram overlap B is the ratio of shared bi-grams to unique bigrams in P. The bigram overlap $B(P, S)$ between the sub-problem and solution is given by

$$B(P, S) = \frac{|T_2(P) \cap T_2(S)|}{|T_2(P)|}$$

3. Similarly, the unigram overlap $U(P, S)$ between the sub-problem and solution is given by

$$U(P, S) = \frac{|T_1(P) \cap T_1(S)|}{|T_1(P)|}$$

4. The overall degree of mapping D between the sub-problem P and solution S is obtained by using Mycin's approach to combining evidence:

$$D(P, S) = B(P, S) + U(P, S) - B(P, S)U(P, S)$$

5.3.2 Results and Discussion

The domain expert was asked to judge the correctness of obtained sub-problem to solution mappings. Figure 5.8 shows the degree of mapping between one document's sub-problems and its solution descriptions. The mappings from the expert are shown in bold. The algorithm was able to correctly identify solution descriptions that match with the sub-problems in all instances where the information it relies on was available. For example, *fire alarm* and *stove shutoff isolator* constitute the solution package for the *interfere with appliance* problem. As Figure 5.8 shows, *fire alarm* and *stove shutoff isolator* obtain considerably much higher mapping values in comparison with those obtained by the other devices in the same document. The same pattern is seen for the other mappings except for the situation involving the *flood(ing)* sub-problem. In the document, there is no section

that specifically talks about a solution for the *flooding* problem as was done for the other sub-problems because no flood detection device was provided in response to the *flooding* sub-problem. The mapping algorithm relies on information that appears in the problem and solution parts of the document and particularly, on the description of how the device is meant to help with the sub-problem. Consequently, it was not possible for the algorithm to identify the solution description for this particular sub-problem because a dedicated description does not exist.

case1 **interfere with appliance : fire alarm - 0.65** interfere with appliance : out of bedroom alert - 0.1 interfere with appliance : out of house alert - 0.0 interfere with appliance : page unit - 0.1 **interfere with appliance : stove shutoff isolator - 0.53**
leave the home unsupervised : fire alarm - 0.1 leave the home unsupervised : out of bedroom alert - 0.2 **leave the home unsupervised : out of house alert - 0.65** leave the home unsupervised : page unit - 0.2 leave the home unsupervised : stove shutoff isolator - 0.1
restlessness at night : fire alarm - 0.3333333333333333 restlessness at night : out of bedroom alert - 0.17 restlessness at night : out of house alert - 0.17 **restlessness at night : page unit - 0.67** restlessness at night : stove shutoff isolator - 0.50
flood : fire alarm - 0.13 flood : out of bedroom alert - 0.13 flood : out of house alert - 0.00 flood : page unit - 0.00 flood : stove shutoff isolator - 0.13

Figure 5.8: Mapping Problems to Solutions

The solution for the *flooding* sub-problem was mentioned in a different and general part of the report that is not part of the solutions' descriptions. Figure 5.9 shows the paragraph in which the solution for the *flooding* sub-problem appears. As Figure 5.9 shows, the paragraph also contains SmartHouse device names (underlined) for other sub-problems' solutions. It mentions *door contacts*, *movement detector* and *fire alarms*, which are solutions to the sub-problems regarding *doors*, *movement* and *fire* respectively. Since the device names have overlapping words with their corresponding sub-problem descriptions, the algorithm "sees" this paragraph as related to *doors*, *movement*, *fire* and *flooding* sub-

problems. Indeed, as an experiment to further explore the capabilities of the algorithm, this paragraph was temporarily added to the solution descriptions. The sub-problems *doors*, *movement*, *fire* obtained (very low) similar scores and the *flooding* sub-problem obtained a slightly higher score which can be attributed to the fact that it had two overlapping words (flood and water) with the paragraph as opposed to the other sub-problems that had one overlapping word.

All of the equipment, except for the door contacts, movement detector, fire alarms and paging device were located in a locked cupboard in the kitchen. ***The option for the provision of a flood detection and water shut-off was considered, but this was deemed to be unnecessary. Instead, the care staff opted for the fitting of taps that require the user to push the tops to operate, so they cannot accidentally (or intentionally) be left on.***

Figure 5.9: Solution for the Flooding sub-problem

The approach was tested on 10 documents with 31 sub-problems; 9 documents had all their sub-problems successfully mapped onto corresponding solutions. Altogether, 30 out of 31 sub-problem descriptions were successfully matched. The algorithm was found to perform very well when the phrase and word overlaps it relies on for mapping were available. The mapping scores were also found to be much higher for correct solutions than for wrong solutions. Thus there was a clear distinction between correct and wrong solutions. In the instance where the algorithm was unable to map a sub-problem to a solution description i.e., for the *flooding* problem in Figure 5.8, the mapping scores were not distinctly different for all mappings illustrating that no solution was identified as correct for the sub-problem at hand.

The algorithm was implemented in a tool that enabled the expert to carry out the mapping in a user-friendly manner. The expert found the mapping task easy because not only were the solutions ranked, the correct ones predominantly appeared at the top of the ranking. For example, *Handsfree telephone unit*, *video intercom* and *community alarm telephone base unit* are the solutions to the *intercom operation* sub-problem shown in the screen snapshot in Figure 5.10. The task was also made easier by the fact that no instances of "missing" solution descriptions were found aside from that regarding the

flooding sub-problem that was discussed earlier.

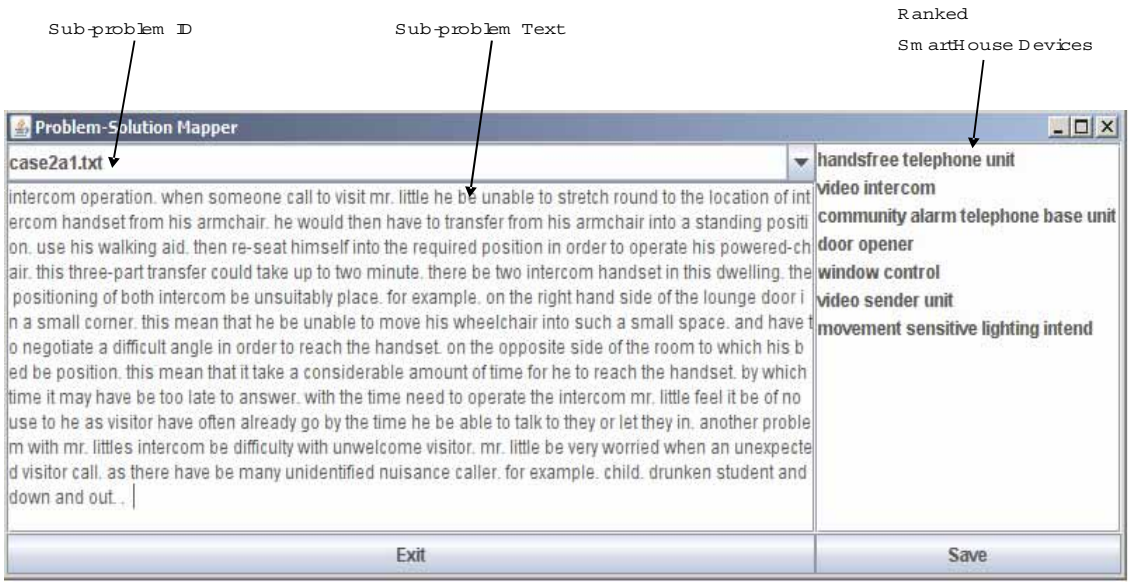


Figure 5.10: Sanctioning of Sub-problem to Solution Mappings

5.4 Summary

This chapter has shown how Formal Concept Analysis can be applied to contexts of sub-problems (objects) and their representative terms (attributes) to obtain conceptual models. Mappings of sub-problems to their respective solution descriptions have also been carried out. The mapping algorithm relies on phrase overlaps between sub-problem and solution texts in order to identify solution descriptions that match with a given sub-problem. The mappings will enable case completion by attaching the solutions to appropriate authored problem-parts of cases. The following two chapters describe two case authoring tools that use FCA to learn conceptual models and use the models to create knowledge-rich cases.

Chapter 6

SmartCAT - A Case Authoring Tool

The previous chapter presented how Formal Concept Analysis can be employed to create conceptual models of the data extracted in Chapter 4. This chapter presents SmartCAT, a tool that creates knowledge-rich cases from textual reports. Sub-problem representations obtained in Chapter 5 are used to learn a graphical conceptual model after which each sub-problem is mapped onto a particular domain-specific concept. The authored problem parts are then tagged with their corresponding solutions using the mappings also obtained in Chapter 5 in order to complete the case descriptions. The resulting cases are used to create a hierarchically organised case-based system.

Section 6.1 presents the creation of the conceptual model which is used to create and organise the cases in Section 6.2. This is followed by Section 6.3 which gives a description of how case similarity is measured during problem-solving. The usefulness of the cases and the adopted retrieval strategy are evaluated in Section 6.4.

6.1 Graphical Conceptual Models

A concept hierarchy is obtained by applying FCA to a context of sub-problem representative terms (attributes) and sub-problem objects. Naturally, the most specific concept-nodes appear at the bottom of the hierarchy and the most general SmartHouse concepts (e.g., the concepts corresponding to disability and disease-terms) appear at the top of the

hierarchy. Figure 6.1 shows part of the top portion of the resultant concept hierarchy for the context obtained from the SmartHouse sub-problems. Only the attributes attached to concepts are shown in the figure. Concept nodes pertaining to disease-terms are also not shown. Nodes labeled 1 to 8 represent nine of the twelve disabilities in the domain i.e, *mobility problem* (node 1), *cognitive problem* (node 2), *hear(ing) problem* (node 3), *learn(ing) problem* (node 4), *poor sight* (node 5), *multi system atrophy* (node 6), *dementia* (node 7), and *ataxia* (node 8). The conceptual model comprises 186 concept nodes in total.

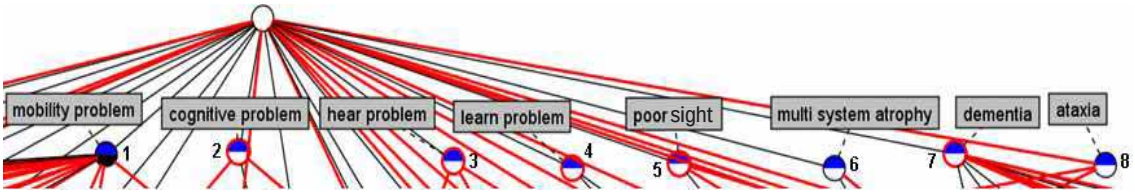


Figure 6.1: Upper Portion of the Graphical Conceptual Model for the SmartHouse Domain

As Bunke (2000) points out, graphical representations offer a number of advantages over traditional feature vector approaches, most notable of which is their ability to create rich case representations (Gebhardt, Voß, Gräther & Schmidt-Belz 1997). The graphical conceptual structure makes it possible to see the relations between the different components of domain knowledge and the consequent cases. Consider the portion of the hierarchy illustrating the *mobility* sub-tree shown in Figure 6.2. The figure shows that it is possible to surmise that *Cerebral palsy* (node 2) can result in *mobility problems* (node 1), or that some people with *mobility problems* are likely to have difficulties in *opening* (their) *doors* (node 8 and one of the attributes for node 7). Thus this type of structure makes it possible to suggest the needs of a person, or difficulties a person might be encountering, if their disability or ailment is known.

6.1.1 Comparison with Manually Obtained Concepts

The obtained FCA conceptual model has been compared to a hierarchy that was manually created in an earlier SmartHouse CBR project (Wiratunga, Craw, Taylor & Davis 2004). For the manual exercise, ten general types of problem were identified and then the more specific problems were manually extracted and clustered under the ten general classes.

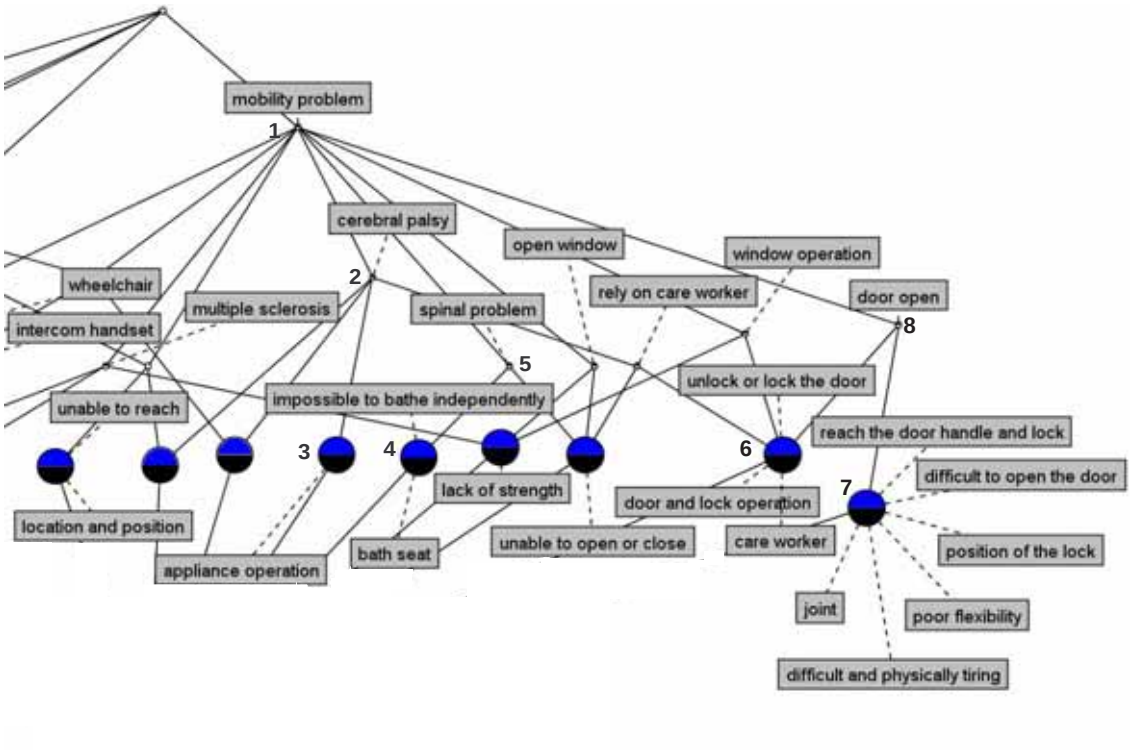


Figure 6.2: Graphical Conceptual Model for the Mobility Problem Sub-tree

Figure 6.3 shows some of the general concepts and the subconcepts for the *Challenging-Behaviour* and *General* categories.

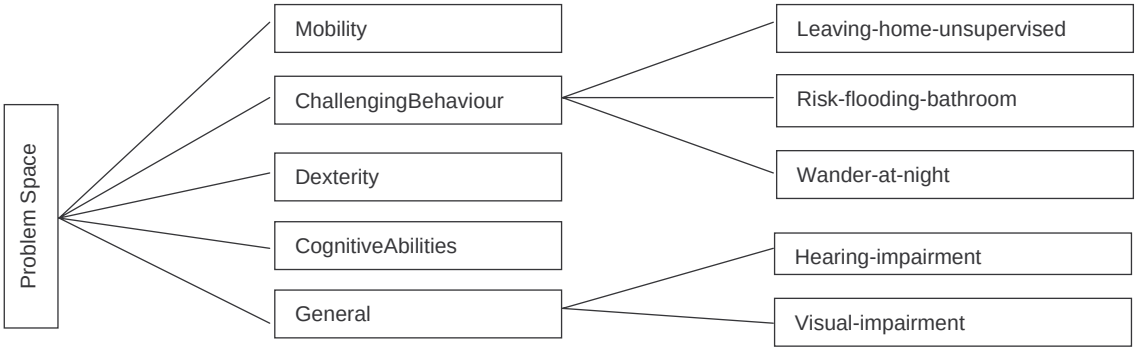


Figure 6.3: Manually Crafted Hierarchy

Figure 6.4 shows part of the SmartHouse conceptual model. Sub-problem descriptor headings have been used to refer to the sub-problem objects for ease of understanding. It is for the same reason that case identifiers have been added to sub-problem descriptor headings in situations where the headings are not unique; e.g., *Telephone Operation (case 8)* differentiates this sub-problem from *Telephone Operation (case 3)*.

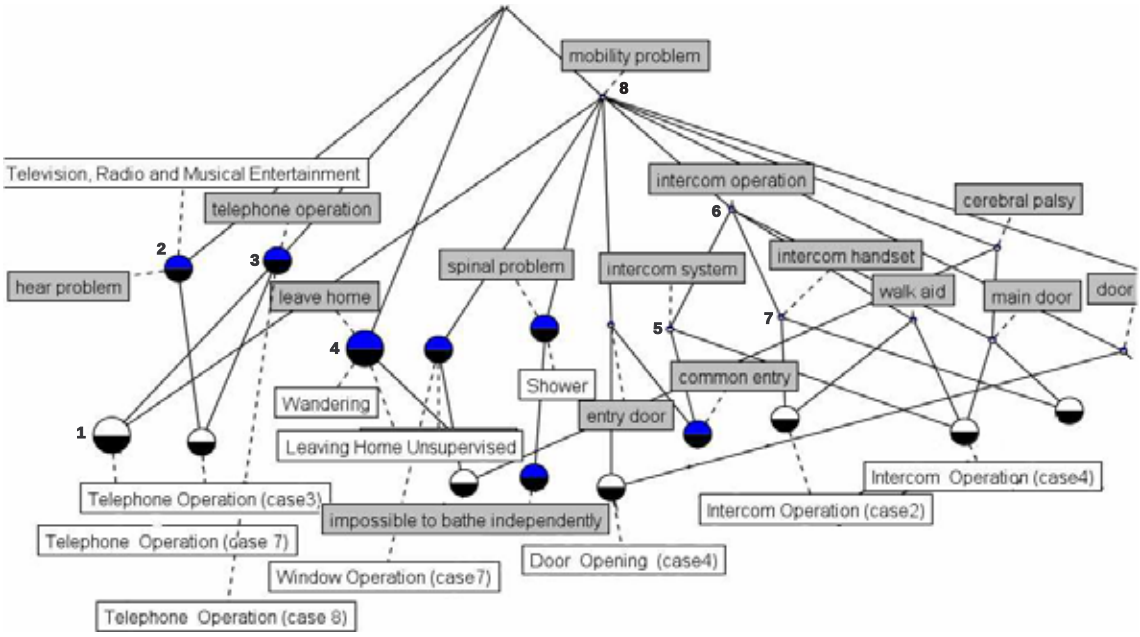


Figure 6.4: FCA-Generated Concepts

Consider the left section of the Figure 6.4 FCA conceptual model featuring *Telephone Operation* objects. There are three FCA concepts involving difficulties with telephone operation: those containing the key phrase *telephone operation* explicitly (node 3 and its children), and those that also mention *hear problem* (node 2 and its child node). A new problem described as *telephone operation* would result in the retrieval of solutions pertaining to a hearing problem or, to mobility problems as is the case in Figure 6.4 (node 1 is also a child of the *mobility problem* concept node labeled 8). Furthermore, solutions from cases for general loss of entertainment (e.g. the *Television, Radio and Musical Entertainment* case) could also be retrieved. In this case, FCA manages to create useful links between concepts that are not captured by knowledge representations like the manually created hierarchy (Asiimwe, Craw & Taylor 2006).

Looking at the *ChallengingBehaviour* class of the manual hierarchy, a wide variety of problems are seen, demonstrating different forms of challenging behaviour two of which are *leaving-home-unsupervised* and *wander-at-night*. Considering the FCA concept with intension *leave home* labeled as node 4 in Figure 6.4, the problem descriptor paragraphs *Leaving Home Unsupervised* and *Wandering* have formed a concept because in both cases the person had a tendency to *leave home*. This is captured by the extension and intension

of the FCA concept node.

The FCA generated hierarchy shows several door related problems that are difficult to relate together without the key phrases in the text. Similarly there are several relevant attributes: different types of door, entry actions, and mention of a lock. In contrast, all the problems involving intercoms have been named as *Intercom Operation*. However, there are different ways that the problem is manifest: the handset, the intercom system, its operation, and even the need to manage a walking aid. These are all clarified in the FCA hierarchy. It is also worth noting that concepts in which *intercom handset* (node 7) and *intercom system* (node 5) are intents, have been captured as sub-concepts of concepts in which *intercom operation* (node 6) is an intent.

The concept hierarchy exhibits multiple inheritance, which makes it compact thus allowing for saving of storage space and for efficient navigation. However, multiple inheritance can lead to inference errors when using the hierarchy to interpret data. For example, the phrase *risk of fall(ing)* applies to both people with *mobility problems* and those suffering from *dementia*. Thus a concept node to which *risk of fall* is attached will be a child of nodes pertaining to *dementia* and *mobility problems*. This can lead to misleading inferences when dealing with one of the two disabilities and not the other.

When new knowledge is obtained, the concept hierarchy will have to be redrawn in order to have a correct picture of all the domain knowledge and the relations between the different cases. Consequently, cases that share terms with the new case knowledge will need to be amended since such terms will now attain a higher position in the hierarchy. Effecting these changes ensures that new knowledge is not omitted from cases when new facts are encountered as would happen if the concept hierarchy were used as a fixed template for case authoring.

6.2 Creating Cases Using SmartCAT's Conceptual Model

SmartCAT uses the graphical conceptual model to define and organise cases. Normally, an occupational therapist would record a person's disabilities, problem areas and symptoms, under pre-defined groupings: *wheelchair* would be recorded under *mobility problem*; *unable to hear buzzer* under *hearing problem*. Similarly, the case authoring task involves mapping

the sub-problem representative terms onto the discovered concepts in the hierarchy and attaching a set of solutions that assists with the sub-problem.

Two types of case representations can be obtained by using SmartCAT's conceptual model. One option is to represent each sub-problem as a sub-graph of the conceptual model, by identifying in the model, concepts that pertain to the sub-problem. The second is to create flat-structured cases with the conceptual model acting as an underlying structure that can be utilised for similarity based retrieval during problem-solving.

6.2.1 Graphically Structured Cases

Graphically-structured cases can be obtained by instantiating as present, all concept nodes whose whole intent appears in the sub-problem's representation. Figure 6.5 shows the case with the attributes {*dementia*, *lost at night*, *wander*, *likely to leave the home and wander*, *challenging behaviour*, *disorientation*, *lose track of time*, *unable to walk in familiar neighbourhood*}.

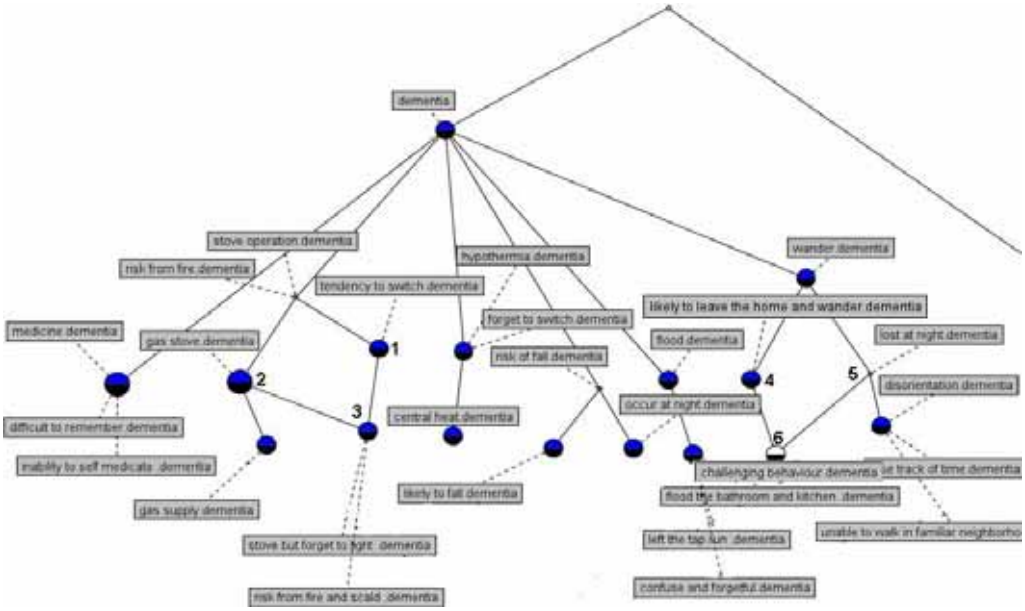


Figure 6.5: Structured SmartCAT Case

Each case becomes a sub-graph of the conceptual model. Problem-solving with such cases entails the comparison of common sub-graphs between cases and the query. This is a potentially complex task because it is NP-hard in the number of concept nodes. Moreover, an erroneous query structure, resulting from multiple inheritance in the conceptual model,

is potentially harmful to retrieval effectiveness as cases with the erroneously inherited features could be flagged up as "similar" to the query. It is for these reasons that graph-structured cases were not exploited in this project.

6.2.2 Flat Structured Cases

The second and less complex option is to author cases as flat structures as was done in earlier work by Asiimwe et al. (2007). Each sub-problem is mapped on to a particular concept by finding one whose *whole* intent is *all* of the sub-problem's representative terms. For example, Node 7 in Figure 6.6 (repeated from Figure 6.2 on Page 96) represents a concept whose intent is {*mobility problem*, *door open*, *reach the door handle and lock*, *poor flexibility*, *joint*, *difficult and physically tiring*, *position of the lock*, *difficult to open the door*}. This intent is also *all* the terms identified using LSI in the *door opening* sub-problem illustrated in Figure 6.7 (repeated from Figure 4.14 on Page 77). *Mobility problem* is the disability term in the document added to the problem-descriptor terms in order to complete the sub-problem's representation. Thus the *door opening* sub-problem is mapped on to the concept represented by node 7.

The term *concrete* shall be used to refer to those concepts onto which a sub-problem is mapped. Concrete concepts include all of the most specific concepts in the hierarchy, and some abstract concepts whose intent *completely* represents a sub-problem in the document collection. Abstract concepts on to which no sub-problem is mapped will be referred to as *completely-abstract*. Mapping each sub-problem on to a concept transforms the document into a set of concrete and abstract concepts. Therefore, a document that contains n sub-problems is mapped onto n concepts in the concept hierarchy. Figure 6.8 illustrates case authoring for four documents containing 3, 4, 4 and 2 sub-problems respectively.

Thus each sub-problem is transformed into a flat-structured, problem part of a case. However, these still have the conceptual model as an underlying structure. Figures 6.9 and 6.10 show the transformation from lemmatised text to graphical structure for the running example of the *wandering* sub-problem in Figure 1.2. Figure 6.9 (a) and (b) show the lemmatised version of the text and the sub-problem's representative terms respectively. Figure 6.10 shows the graphical case corresponding to the sub-problem. All the nodes show part of the underlying structure. The highlighted nodes and paths all pertain to the sub-

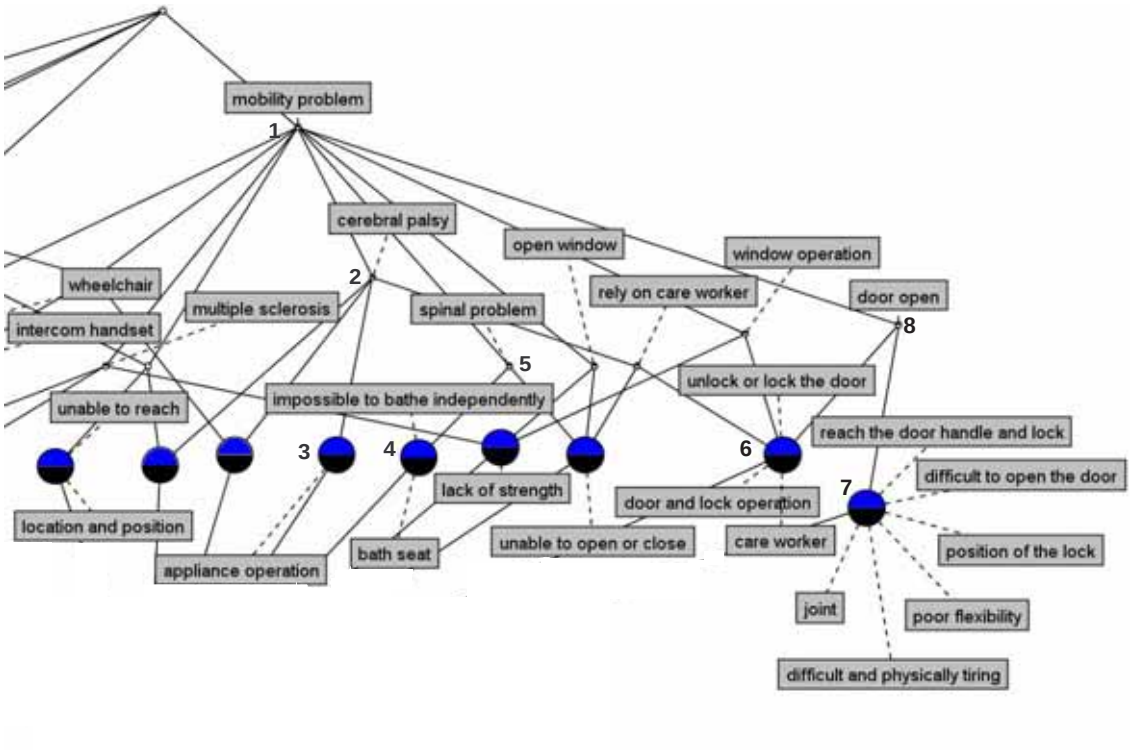


Figure 6.6: Graphical Conceptual Model for the Mobility Problem Sub-tree

LSI Highlighting Using Term Importance

door open

when mr. M wish to open the front door, he had to project himself forward to ***reach the door handle and lock****. because of* ***poor flexibility*** *in the* ***joint****, he found this task to be extremely* ***difficult and physically tiring****. also, the* ***position*** *of the* ***lock*** *made it both* ***difficult to open the door*** *from the inside upon leave, or from the outside when return home.*

Figure 6.7: Problem-descriptors Highlighted Using Term Importance

problem. The top-most node has the term *dementia* as its intent and the most specific node's intent contains the term *husband*. It is worth noting that this structure is potentially confusing because of the high level of dependencies between the various attributes in the context table. Addressing this will be the subject of Chapter 7.

Each problem part is tagged with its corresponding SmartHouse devices (and their descriptions) using the problem-solution mappings obtained in Chapter 5. The resulting cases are organised according to the conceptual model which permits the viewing of relationships between the different features in the problem space. This is implemented in the

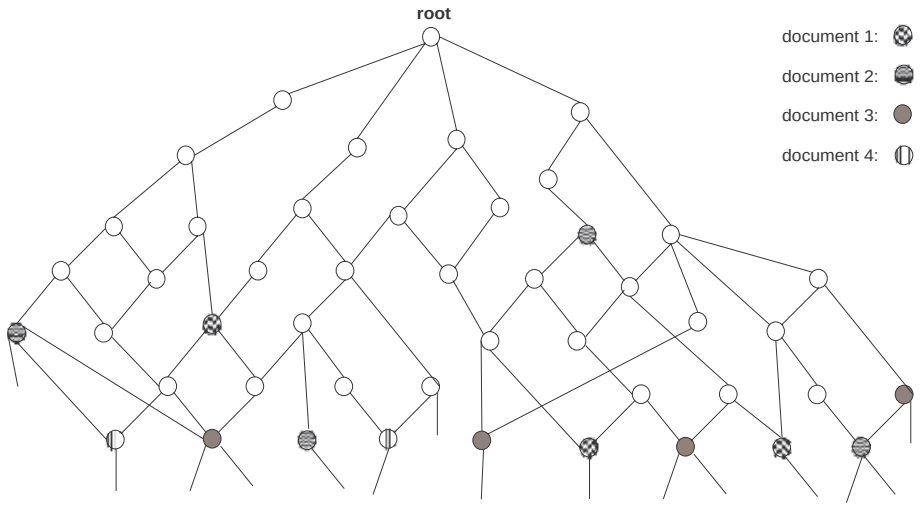


Figure 6.8: Case Authoring Using the Graphical Conceptual Model

(a) Lemmatised Text

wandering mrs m had a tendency to leave the house and wander off. she had recently wander away from the house in the middle of the night. her husband had found her outside, dress only in a thin nightgown, try to open the side door to a neighbor home...

(b) Sub-problem Representation

dementia, alzheimers disease, wandering, leave the home, home, wander, away, night, husband, open, door

Figure 6.9: Lemmatised Text and Representative Terms for the *Wandering* Sub-problem

case-based reasoning system, SMART. The next section presents the process of measuring similarity between queries and flat cases in SMART.

6.3 Similarity Measurement of Flat Structured Cases

Case retrieval in SMART makes use of the conceptual structure to determine similarity between cases in the case base and the query. The retrieval mechanism aims to find the most specific concepts that match the new problem description. A phrase in the intent that is shared by a previously solved problem can be used to determine the solution to the current one. Conversely, unshared terms suggest possible distinctions between a current and previous cases. During problem-solving, the concept hierarchy is searched for the most specific concepts matching the query terms. Ideally, a concept is activated by a

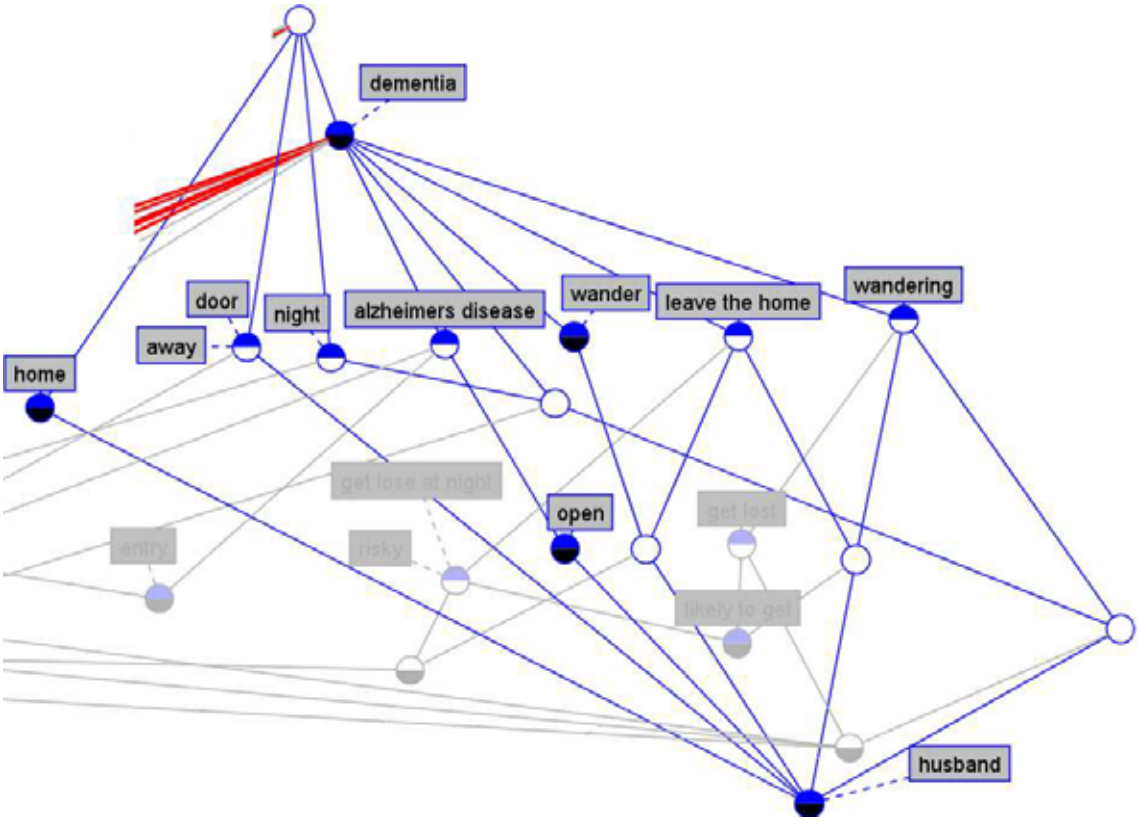


Figure 6.10: Graphical Case for the Wandering Sub-problem

query term that forms part of its intent. However, since a query term cannot always be the same as that in the intent, substring matching of the query and intent strings is used to ensure that the two need not be exactly the same for a concept to be activated. A path ending at a concrete concept node leads to retrieval of the attached solution. However, when a path ends at a completely-abstract node, it is not possible to predict the next point in the path without further knowledge about the problem. In this situation, SMART returns the **disjunction** of all solutions of concrete and abstract concepts that are reachable by descending the subtree from which the completely-abstract concept is attached. If all that was known about a problem was that the person has *cerebral palsy* and a *mobility problem*, nodes 1 and 2 in Figure 6.6 would be activated. Without any further knowledge, it is difficult to predict if the person also has a *door opening* problem (inherited at node 6 from node 8), an *appliance operation* problem (node 3) or any other problem whose concepts are reachable by descending from node 2. Generally, the retrieved solution depends on how much of the problem is described in the query.

A simple similarity measure was adopted. The measure deliberately favours cases with long phrases in common with the query to cases with none. It also correctly favours concrete nodes to abstract ones because the lower the concept node, the more specific it is and therefore, the more similar it is to the new problem. Let $P(q, n)$ be the set of single-word tokens the concept node n has in common with the query q, that also make up a two-word phrase. Let also $T(q, n)$ be the set of single-word tokens the concept node n has in common with the query q, that do not have to be part of a phrase. The tokens should not be stopwords. Overlapping phrases are better indicators of relatedness than overlapping words. Consequently, we wanted to assign a higher weight to overlapping phrases than to overlapping words. A number of coefficients from 1.2 through to 2 were used in turn and it was found that the coefficients 1.4, 1.5, and 1.6 resulted in reasonable weights for solution descriptions that matched sub-problems. A coefficient of 1.5 was selected because it was the median of the 3 equally good coefficients. Consequently, similarity $similarity(q, c)$ between a query q and a concrete node c is given by

$$similarity(q, c) = |T(q, c) \backslash P(q, c)| + 1.5\,|P(q, c)|$$

Concrete nodes generally represent specific concepts and would therefore be more similar to a query than abstract nodes. Therefore we wanted to assign a higher weight to concrete nodes than to abstract nodes. This was to be achieved by using the smallest coefficient that would give concrete nodes an advantage over abstract nodes. The values of 1.1 to 1.4 were tried out and 1.2 gave the smallest advantage and was thus chosen. Thus similarity $similarity(q, a)$ between a query q and an abstract node a is calculated as

$$similarity(q, a) = \frac{|T(q, a) \backslash P(q, a)| + 1.5\,|P(q, a)|}{1.2}$$

The graph is traversed as far down as possible using all the non-stopword tokens in the problem description, until there are no specific concept nodes to be found. These are then ranked, with the concept node having the most number of common terms appearing at the top of the ranking. The more the number of terms in common with the query, the more similar the concept node is to the current problem. However, two or more word terms are

considered to weigh more than single words. Thus a concept node that shares with the query, the most number of long terms is ranked higher than those with fewer (long) terms in common. Similarly, a concept node with more single words in common with the query is ranked higher than those with few words in common. The words in this instance do not include those that are part of any long terms that have already been used for ranking.

Unmatched query terms are used for tie breaking when some of the concept nodes are ranked the same due to their having the same number of shared attributes with the query. For each concrete or abstract concept node in the tie, the corresponding solution description is used to determine that concept node's ranking with respect to the other concept nodes in the tie. A concept node whose solution's description has more overlapping terms with the query is ranked higher than one whose solution has fewer overlaps. Similarly, a completely-abstract node whose solution (i.e., solutions to cases whose concept nodes are reachable by descending the subtree from the abstract node) has the most terms in common with the query is ranked higher than those whose solutions have fewer terms in common. However, a completely-abstract node's solution is a conjunction of *all* solutions attached to concepts that are reachable by descending the hierarchy from the abstract node. This also creates a tie between these solutions because they appear to be equally plausible with respect to the abstract node. Overlaps between the set of unmatched terms and the abstract node's solution's descriptions can be used to break the tie between the solutions. The bigger the overlap between a solution's terms and the set of unmatched terms, the higher will its ranking be, with respect to the other solutions pertaining to the abstract node.

Unlike traditional graph-based representations such as CAPER (Sanders, Kettler & Hendler 1997) in which nodes are made of different bits of case knowledge, each SMART flat case corresponds to a sub-problem and thus comprises a single concept node which has all the information pertaining to the case and to all its super concepts. Thus all the retrieval mechanism needs is to traverse the graph in search of concrete or abstract nodes that match with the information in the query. Consequently, complex graph matching is avoided. However, this also implies that cases at the same level in the conceptual model cannot be compared at different levels of abstraction whereas graphically structured cases can.

6.4 Evaluation

The authored cases are evaluated by testing whether they capture knowledge that helps in finding solutions. An experiment is also carried out as to whether the retrieval strategy employed by organising the cases in a hierarchy results in retrieval of useful solutions. SMART and the expert's solution packages for four problems are compared.

Figure 6.11 contains four test problems. Problems *A* and *B* were handcrafted by the expert who ensured the description terms were the same as the ones in the case base. Although these terms are familiar, SMART does not have cases whose problems are completely described by the same terms. Problems *C* and *D* are problem parts of actual reports that are excluded in the creation of the hierarchy and from the case base. They test whether the rest of the cases in the case base capture enough knowledge about the domain in order to give useful solutions to problems that have not only not influenced the case representation, but are also not part of the case base. They also test SMART's ability to find appropriate parts of useful cases in order to reuse them for problem-solving. Hence, problems *C* and *D* are more challenging.

Problem A	window opening, door opening, spinal problem, wheelchair user, unable to bathe independently
Problem B	cognitive problems, aphasia, confusion, disorientation, fully ambulant, perseveration, no insight into condition
Problem C	intercom operation, unable to hear buzzer, telephone operation, problem hearing the caller, unable to listen to television, can only watch pictures on television
Problem D	paralysis, ataxia, chair bound, body constricted, poor hand to eye coordination, left sided weakness, copious aspiration

Figure 6.11: Test Problems

Sometimes the occupational therapist has a list of terms describing the person's complications and she is required to *anticipate* the needs of the person. This situation is mimicked by testing SMART's ability to recognise terms as belonging to given sub-problems and subsequently retrieving their solutions. In problems *B* and *D*, the person's complications are given but the specific needs like *door* or *window opening* are not explicitly stated as is the case with problems *A* and *C*. This makes problems *B* and *D* particularly challenging. Thus problem *B* is more challenging than *A*, *C* is more challenging than *B*, and problem *D* is more challenging than *C*.

SMART attempts to index each problem in order to retrieve the appropriate solutions. Figure 6.12 shows the solution packages suggested by SMART and the expert for each test problem. The similarity of solutions for SMART and the expert are compared using *Precision* and *Recall*. In the SmartHouse domain, *Precision* is the proportion of SmartHouse devices proposed by SMART that occur in the expert solution package and *Recall* is the proportion of devices proposed by the expert that are correctly proposed by SMART. *Recall* is more important than *Precision* because SmartHouse solution recommendation is typically a supervised task. An occupational therapist prefers to be presented with a list of devices to choose from than to have a list of devices that perfectly solve only a part of the problem and be required to formulate the rest of the solution from scratch. Altogether, there are 102 sub-problems, each of which can be solved by one or more SmartHouse devices.

Solution A	CBR	Expert
powered windows	Yes	Yes
powered external doors	Yes	Yes
community alarm	Yes	No
electrically operated locks	Yes	Yes
environmental controls	Yes	No
shower with sitting facility	Yes	Yes
Precision = 0.7	**Recall = 1.0**	

Solution C	CBR	Expert
video intercom	No	Yes
visual doorbell	Yes	Yes
telephone amplifying unit	Yes	Yes
video interface to telephone	No	Yes
television/audio amplifying headset	Yes	Yes
Precision = 1.0	**Recall = 0.6**	

Solution B	CBR	Expert
smoke/heat/gas alarms	Yes	Yes
stove shutoff isolator	Yes	Yes
intelligent microwave	Yes	No
community alarm	Yes	Yes
environmental controls	Yes	Yes
out-of-house alert	Yes	No
flashing lights as prompts to check PC for next activity	No	Yes
Precision = 0.7	**Recall = 0.8**	

Solution D	CBR	Expert
smoke/heat/gas alarms	Yes	Yes
video entry phone	No	Yes
door entry system	Yes	No
community alarm	Yes	Yes
environmental controls	Yes	Yes
very sheltered accommodation	Yes	No
needs assistance with toileting and feeding	No	Yes
Precision = 0.6	**Recall = 0.6**	

Figure 6.12: SmartHouse Devices for Test Problems

Problem *A* was the easiest to solve and this is confirmed by the high values of recall obtained by SMART. Consider the portion of the hierarchy shown again in Figure 6.13. The problem features *spinal problem* and *unable to bathe independently* activate concept nodes 5 and 4 respectively. Node 4 is a concrete node which results in the return of the attached solution *shower with sitting facility*. Similarly, the feature *door opening* activates nodes 6, 7, 8 and other nodes not shown in the Figure. Two of the activated paths end at

completely-abstract concept nodes which results in the generation of two false positives namely, *community alarm* and *environmental controls*. Nevertheless, SMART recommends all the solutions that are proposed by the expert hence obtaining high recall for this problem.

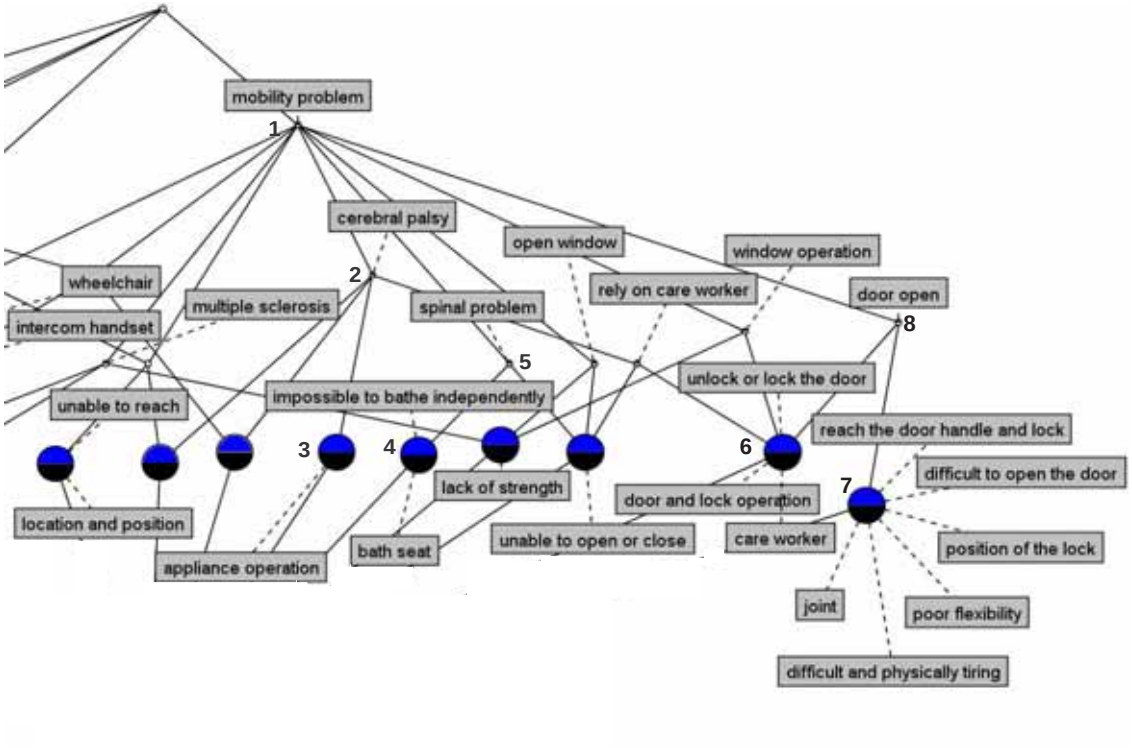


Figure 6.13: Concept Activation in the Graphical Conceptual Model

Problem *B* was harder to solve as SMART was required to find devices that would help the person, without knowledge of the person's specific needs. Problem *B*'s features are quite general and thus SMART was more prone to returning solutions for completely-abstract concept nodes than for concrete ones. Nevertheless, SMART recommends the same solutions as the expert in many instances. The expert ruled out the need for an *intelligent microwave* since the problem's features do not indicate severe disabilities. Furthermore, an *out-of-house alert* would be unnecessary once the *community alarm* was installed. Thus these solutions were only false positives in the sense that it would be "overkill" to incorporate them in the solution package.

In response to the *intercom operation* sub-problem in problem *C*, SMART activates a concrete node that results in the return of the solution *visual doorbell*. However, it fails

to suggest a *video intercom* and a *video interface to telephone*. The expert explained that he had suggested the two devices as a means of incorporating redundancy in the solution as is usually done in good engineering design. For example, a radio will have a mains inlet and a provision for batteries to be used solely for purposes of increasing the radio's reliability. Another interesting thing that was noted was that, for the *television* sub-problem, SMART returns the solution *television amplifier* OR *audio amplifying headset* because this particular path ends at a completely-abstract node. However, in real-life, either solution would assist with the sub-problem that is why the expert gives the solution as being either the *television amplifier* or the *audio amplifying headset*. Thus SMART recommends the right solution by returning solutions of cases attached to a completely-abstract node at which the search path ends.

Problem D was the most challenging. The fact that SMART obtains reasonable values of precision and recall shows that there is good coverage of cases in the case base and that the vocabulary used is fairly standard since previously unseen terms can activate concepts in the hierarchy. Again, because of the generality of features like ataxia (a disability), SMART suggests more devices than is necessary; *very sheltered accommodation* would only be required if the person were more severely disabled. Furthermore, a *video entry system* would be more effective than the *door entry system* that SMART suggested, because of the added advantage of the person being able to see the caller before they let them in. Thus a *video entry system* is a more specialised type of *door entry system*. It was also noted that there was a need for the hierarchy to expand its vocabulary and incorporate new terms. For example neither the query term *poor hand to eye coordination* nor its sub-strings activate any concept node. This could be done by the expert refining the concept hierarchy when she saw a need during problem-solving.

6.5 Summary

This chapter has presented SmartCAT, a case authoring tool that creates knowledge-rich cases from the sub-problem representations obtained in Chapter 5. SmartCAT uses the sub-problem representations to learn a conceptual model and then exploits the model to create knowledge-rich cases and to organise the cases into a structure based on concept-

superconcept relationships in the conceptual model. The result is SMART, a hierarchically structured case-base where cases exist at several levels of abstraction and all nodes whose intents completely represent a sub-problem are tagged with the sub-problem's solutions.

The FCA hierarchy compared well with the manual hierarchy from earlier work, capturing similar relationships, but also adding more levels to the hierarchy, thus distinguishing more finely between concepts. Unlike most textual case-based reasoning systems, SMART's case knowledge and the structure of the case-base are generated automatically. This feature makes the case knowledge for SMART easy to acquire and maintain from textual records. Flat-structured cases offer an advantage over graph-structured cases which are prone to comprising erroneous features as a result of multiple inheritance. Moreover, retrieval of flat cases is simpler than that for graph-structured ones. However, flat-structured cases require the whole of the conceptual model to be traversed in search of concrete nodes that match the new problem. In instances where the conceptual model is huge, the search process would make the use of such systems prohibitive. Moreover, cases at the same level in the conceptual model, cannot be compared at different levels of specificity.

Consequently, this project opted to explore the use of tree-structured cases instead. Tree structures do not have the problems associated with multiple inheritance as graph structures do, and they can be compared at different levels of abstraction which gives them an advantage over flat cases that use the conceptual model as an underlying structure. The next chapter presents the SmartCAT-T tool which uses the data to learn a tree-structured conceptual model and uses the model to create knowledge-rich hierarchically structured cases that allow for comparison at different levels of abstraction.

Chapter 7

SmartCAT-T: Exploiting Tree-structured Models

The previous chapter described the SmartCAT tool which learns a graphical conceptual model in order to create knowledge-rich cases. This chapter presents SmartCAT-T, a tool that creates a tree-structured conceptual model by overcoming multiple inheritance inherent in SmartCAT's graphical model. SmartCAT-T achieves this by changing the nature of sub-problem representations before it employs FCA to create the tree-structured conceptual model. The sub-problem representations are mapped onto concepts in the model in order to create knowledge-rich hierarchically structured cases. SmartCAT-T's cases do not have the problems associated with multiple inheritance and they can be compared at different levels of abstraction during problem solving. The conceptual model is created in Section 7.1 and compared to a manually obtained case representation in Section 7.2 after which SmartCAT-T uses it to create cases in Section 7.3. A refined similarity measurement that exploits the tree-structured model during problem-solving is discussed in Section 7.4.

7.1 Tree-structured Conceptual Models

A tree-structured conceptual model is obtained by changing the nature of the attributes used to create the formal context. Some of the terms do not discriminate between disabilities for example, a phrase like *intercom operation* can appear in documents, and

consequently in cases, where the person had a *hearing impairment* (and could therefore not hear the buzzer), or where the person had a *mobility problem* and had difficulty reaching the intercom because of its positioning. Disambiguating terms with respect to people's disabilities has the effect of removing some of the dependencies between domain concepts and consequently, a removal of these connections in the conceptual model. This in turn results in fewer instances of multiple inheritance.

Term disambiguation is achieved by tagging each term with the disability-term(s) in its parent document. Thus *intercom operation.mobility problem* will be different from *intercom operation.hearing problem* where the term after the period is the disability-term. Figure 7.1(c) shows representative terms for the *wandering* sub-problem; the terms have been tagged with the disability-term *dementia*. The disability-term is not tagged since it will only appear in documents pertaining to people with that disability.

(a) Key Phrases Using Term Importance

wandering, leave the home, home, wander, away, night, husband, open, door

(b) Sub-problem Representation

dementia, alzheimers disease, wandering, leave the home, home, wander, away, night, husband, open, door

(c) Tagged Key Phrases

dementia, alzheimers disease.dementia, wandering.dementia, leave the home.dementia, home.dementia, wander.dementia, away.dementia, night.dementia, husband.dementia, open.dementia, door.dementia

Figure 7.1: Tagged Representative Terms

FCA is applied to a context of sub-problems and all their now *tagged* representative terms in order to obtain formal concepts for the conceptual model. Figure 7.2 illustrates a portion of the resulting concept hierarchy. Using tagged attributes ensures that there are clear demarcations between the different disabilities. Each level 1 sub-tree of the conceptual model represents a disability and the shared concepts provide the different levels of abstraction in that sub-tree. Nodes 1, 2, 3, and 4 are concepts representing some of the most common disabilities. There is also a clear distinction between concepts that may be shared between disabilities as these may warrant different sets of SmartHouse solutions. Nodes 5 and 7 are *flooding* problems due to the disabilities *dementia* and

learn(ing) problem respectively. This concept hierarchy has more nodes than the graphical hierarchy but it is less confusing and more comprehensible.

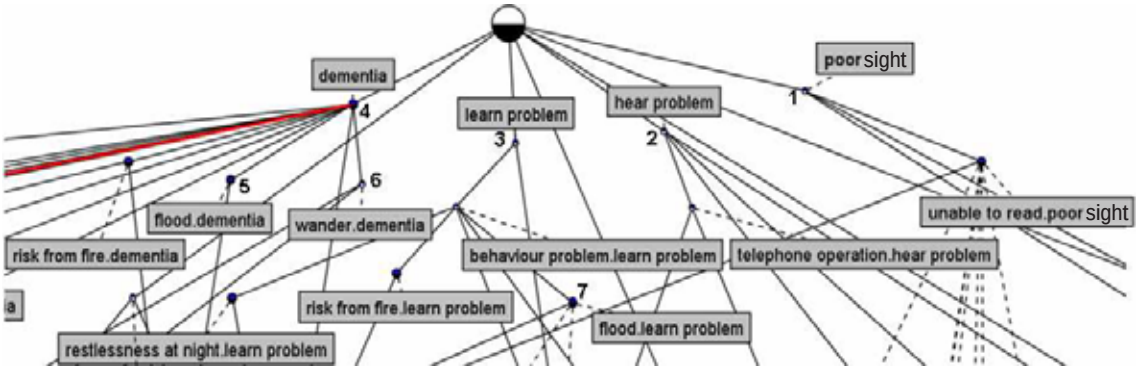


Figure 7.2: Concept Hierarchy with Tagged Intents

The last step in attaining a tree-structured concept hierarchy searches for any concepts exhibiting multiple inheritance. Tagging attributes with disability-terms makes multiple inheritance very rare but not completely impossible. Short phrases (typically those comprising single tokens) can occur in several sub-problems and so a sub-problem is likely to share different attributes with more than one sub-problem resulting in multiple inheritance. Consider Figure 7.3 which shows the hierarchy for the dementia sub-tree. Node 3 inherits the attribute *tendency to switch* from node 1, and inherits *gas stove* from node 2. Node 6 also inherits the attribute *likely to leave the house and wander* from node 4, and inherits *lost at night* from node 5. Multiple inheritance is overcome by creating new attributes and objects. The idea is to get rid of multiple inheritance whilst retaining the concept relations in the original hierarchy.

7.1.1 Multiple Inheritance

The concept hierarchy shown in Figure 7.4 has been borrowed from Priss (2006) to illustrate the task of removing multiple inheritance. The animals' concept hierarchy exhibits more multiple inheritance than the SmartHouse but it is easy to comprehend and it is thus used here to enable the reader to understand the ideas in the work, by applying them to more accessible concepts than those in the SmartHouse domain. In Figure 7.4, all of the animals save for *Harriet* the tortoise, exhibit multiple inheritance. For example, *Garfield* (node1) is a *cat* (node 2) and a *cartoon* character (node 3). *Garfield* shares the attribute

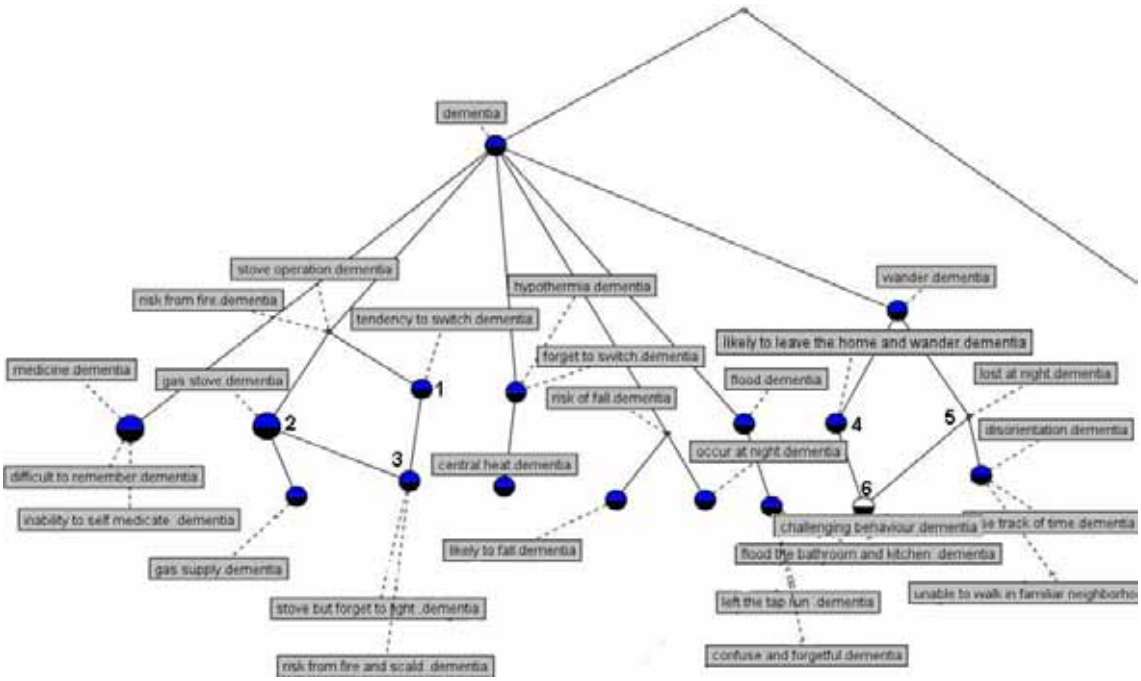


Figure 7.3: Intermediate Concept Hierarchy

cat with *Socks* and the attribute *cartoon* with *Snoopy*. At node 1, *Garfield* inherits the attributes *cartoon* and *cat*.

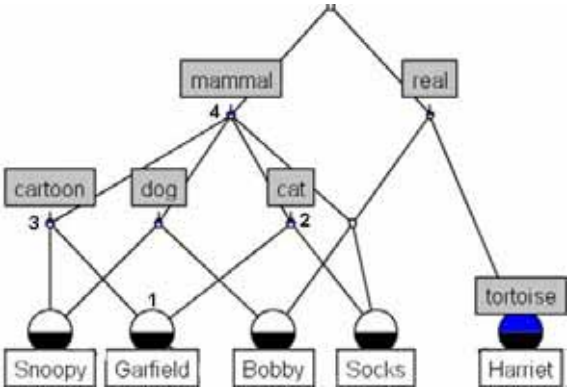


Figure 7.4: Animals Graphical Concept Hierarchy

Figure 7.5 shows the animals' version of a tree-structured hierarchy. Two new objects are created to replace the original Garfield object. The first, *Garfield1* (node 1), will have the original attributes *cat* and *mammal*, in addition to a new attribute *cartoon* that is different from the original *cartoon* attribute. The new and original *cartoon* attributes will appear "different" to the ConExp tool thus enabling the creation of a concept hierarchy

with 2 *cartoon* attributes. The second object, *Garfield2* (node 2), will have the original attributes *cartoon* and *mammal*, in addition to a new attribute *cat*. The process is repeated for all the other objects that exhibit multiple inheritance. The ellipses in Figure 7.5 highlight the new *cat* and *cartoon* attributes introduced to resolve *Garfield*'s multiple inheritance. The result is a slightly higher number of objects and attributes; the number of additional objects created to remove an object's multiple inheritance is equal to the number of objects from which the object is inheriting attributes. However, since the original inheriting objects are removed, the overall increase in number of objects is nearly half the number of objects from which the object is inheriting attributes. For instance, the original animals' concept hierarchy had 11 concept nodes (excluding the root node) where 4 of these nodes inherit attributes from 2 concept nodes. 8 new concept nodes are created but since the original objects exhibiting multiple inheritance are deleted, the overall increase in number of nodes is 4. Thus the tree-structured version has 15 nodes.

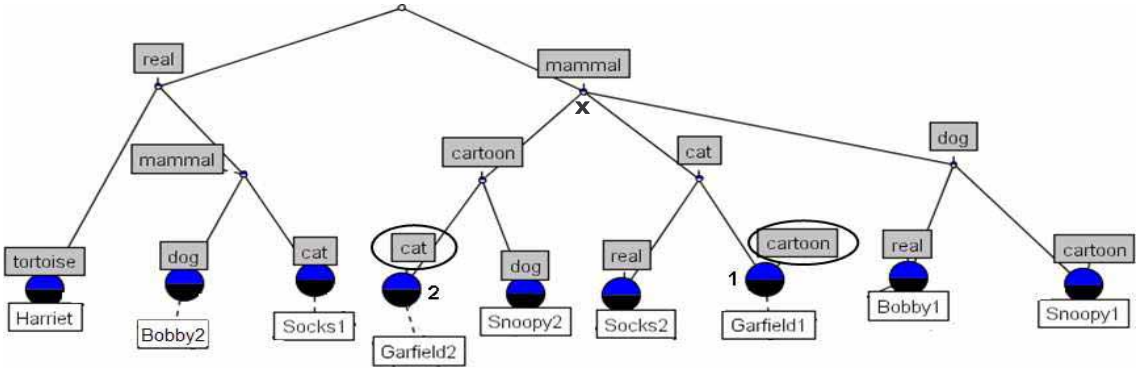


Figure 7.5: Animals Tree-structured Hierarchy

It is worth noting that the original relationships are maintained. For instance, *Garfield* (through *Garfield1*) shares the attribute *cat* with *Socks* through *Socks2*. *Garfield* through *Garfield2* also shares *cartoon* with *Snoopy* (through *Snoopy2*). A query containing only the abstract attribute *mammal* would activate concepts that are reachable by descending the sub-tree from the *mammal* node marked x, and a query with all *Garfield*'s attributes (*mammal*, *cartoon*, *cat*) would activate *Garfield1* and *Garfield2* objects.

7.1.2 Overcoming Multiple Inheritance in the SmartHouse Conceptual Model

In the SmartHouse domain's graphical conceptual model shown in Figure 7.6, node 3's multiple inheritance is resolved by replacing node 3 with two other nodes, *3x* and *3y*. Similarly, node 6 is replaced with nodes *6x* and *6y*. Figure 7.7 shows the resulting tree-structured conceptual model. Node *3x* has all the original attributes of node *3* except *tendency to switch* and a new *tendency to switch* attribute created for node *3x*. Node *3y* has all the original attributes of node *3* except *gas stove* in addition to a new *gas stove* attribute. Similarly, node *6x* has all the original attributes of node *6* except *likely to leave the house and wander* and a new *likely to leave the house and wander* attribute. Node *6y* has all the original attributes of node *6* except *lost at night*; it has a new *lost at night* attribute. The original concept nodes *3* and *6* are deleted.

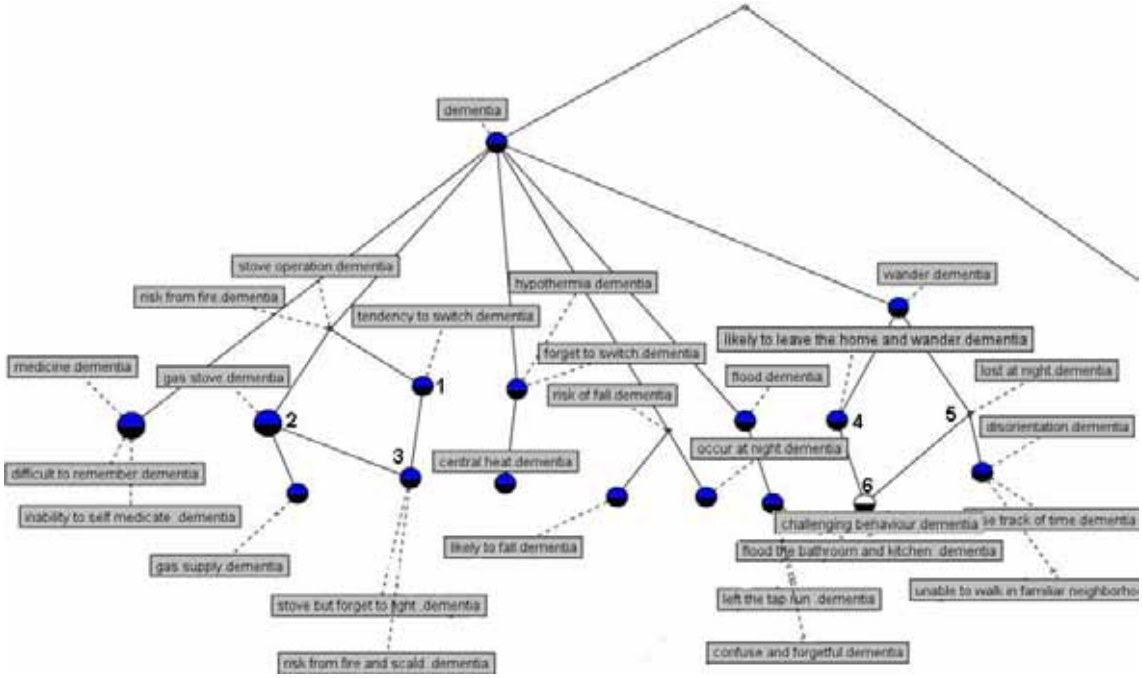


Figure 7.6: Conceptual Model for Dementia Showing Multiple Inheritance

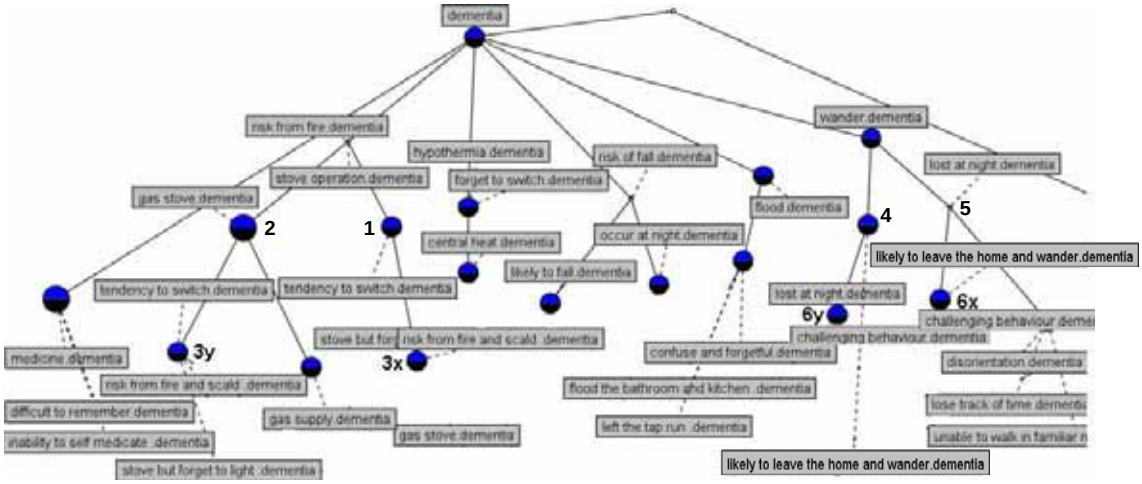


Figure 7.7: Final Tree-structured Hierarchy for Dementia

7.2 Comparison with a Manually Created Case Representation

The tree-structured conceptual model has also been compared to the one that was manually created by Wiratunga, Craw, Taylor & Davis (2004). Figure 7.8 which shows some of the manually obtained concepts is repeated from Figure 6.3. Consider the concept *challenging behaviour* under which the subconcepts *leaving-home-unsupervised*, *risk-flooding-bathroom* and *wander-at-night* are clustered. In *dementia* patients, *flooding* and *wandering* are due to *memory* problems while in people with *learning problems*, the acts are *intentional*. Different solutions may be recommended depending on the underlying disability. Thus cases with the *challenging behaviour* concept of Figure 7.8 will require the user to have knowledge of the previous people's disabilities before they can be re-used. On the other hand, the FCA generated tree-structured conceptual model offers a clear distinction between concepts that may be shared between disabilities. For example, nodes 5 and 7 in Figure 7.9 represent *flooding* problems due to the disabilities *dementia* and *learning problem* respectively. This goes to show that case representations that are crafted by humans are also prone to error. While the automatically created case representation will not be perfect, it is much easier for an expert to amend an imperfect representation than to create one from scratch.

The effectiveness of the tree-structured model will be tested on its ability in interpreting

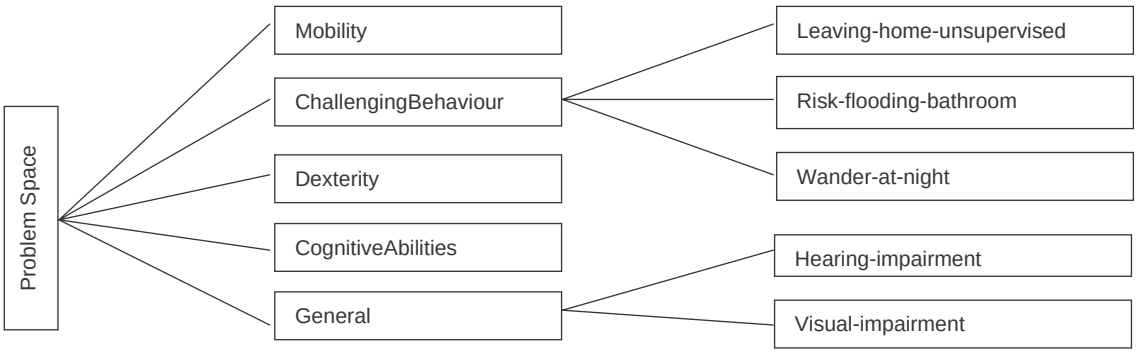


Figure 7.8: Manually Crafted Hierarchy

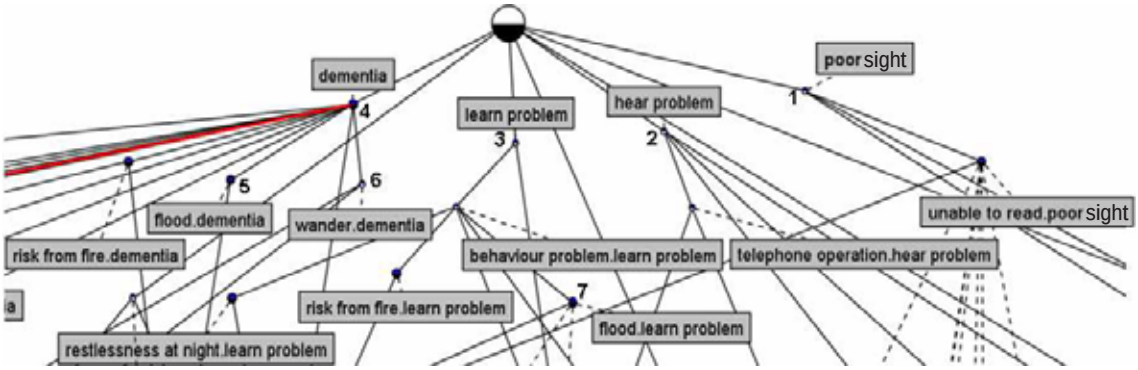


Figure 7.9: Concept Hierarchy with Tagged Intents

similar things as similar. The model would have to be redrawn when new features are discovered about the domain. Similarly, cases whose knowledge includes the new features would have to be amended accordingly in order to reflect the changes in the structure. Tree structures allow for comparison at different levels of specificity. Indeed, case retrieval can be fast if the search starts at the most general concept traversing the appropriate sub-trees until the leaves are reached or the knowledge about the problem is exhausted. Identifying the appropriate sub-tree should not be difficult since the sub-trees are essentially based on people's disabilities. Tree structures also have an advantage in that they do not suffer from problems associated with multiple inheritance as do graph structures.

7.3 Obtaining Cases Using SmartCAT-T's Conceptual Model

Figure 7.9 shows a part of the tree-structured conceptual model for the SmartHouse domain. Figure 7.10 is a portion of the conceptual model illustrating the *dementia* sub-

tree. The tags are removed from the attributes for purposes of aesthetics.

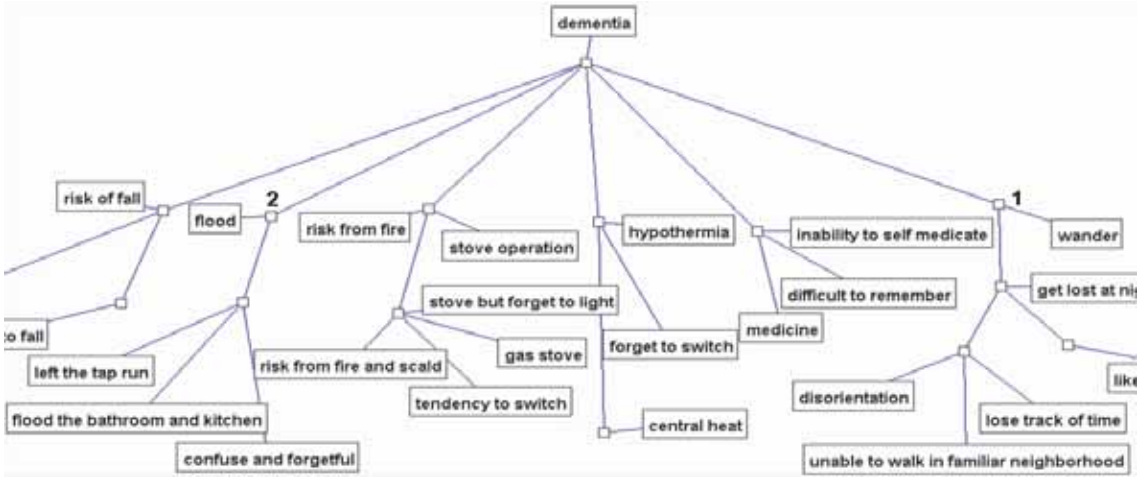


Figure 7.10: Portion of Case Representation Showing the Dementia Sub-tree

The task of creating structured cases from the sub-problem representations involves mapping the representations onto concepts in the case representation. For each sub-problem representation, concepts in which *all* elements of the intent are contained in the problem's representative terms, are instantiated as present. All remaining concepts in the case representation are instantiated as absent. The result is a hierarchically structured problem-part of a case, similar to the representation shown in Figure 7.10. Every sub-problem has a matching list of SmartHouse devices that solve that sub-problem, as obtained in section 5.3. Consequently, corresponding solutions are attached to leaf nodes of the sub-problems to enable problem solving to focus on solutions for individual complaints rather than retrieving complex solutions. Focusing on individual parts of the problem space contribute to retrieval effectiveness (Asiimwe et al. 2007).

Figure 7.11 illustrates a case with the features {*dementia*, *flooding*, *left the tap run*, *confuse*, *wander*, *disorientation*} and {*hear problem*, *intercom operation*, *unable to hear buzzer*}. Figure 7.12 shows the case corresponding to the running example of the *wandering* sub-problem in Figure 1.2. The case structure is clearer and less confusing than the graphical version illustrated in Figure 6.10 on page 103. Concept nodes 1 and 2 have been duplicated to remove multiple inheritance. All this is implemented in the SMARTER[1] system. The next Section discusses the process of comparing structured cases during

[1]**SMART** system with **E**nhanced **R**etrieval.

problem solving.

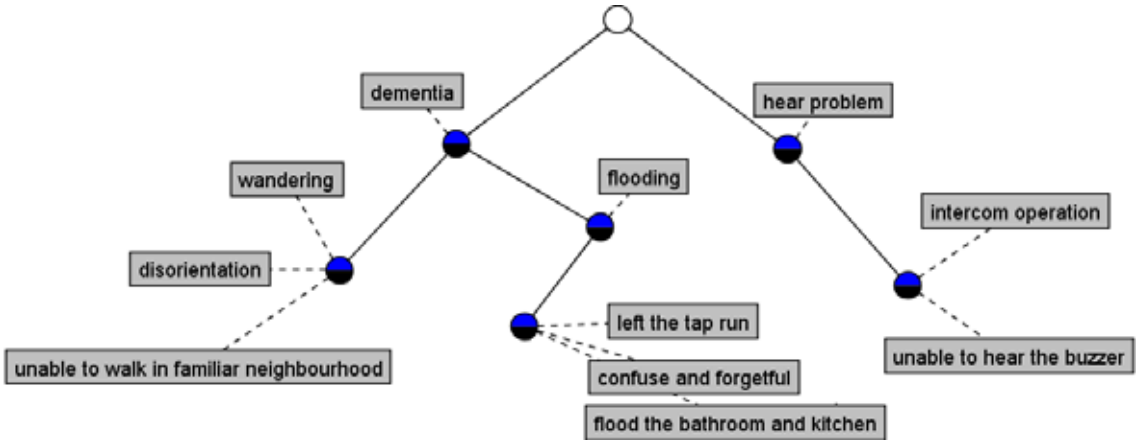


Figure 7.11: Structured SMARTER Case

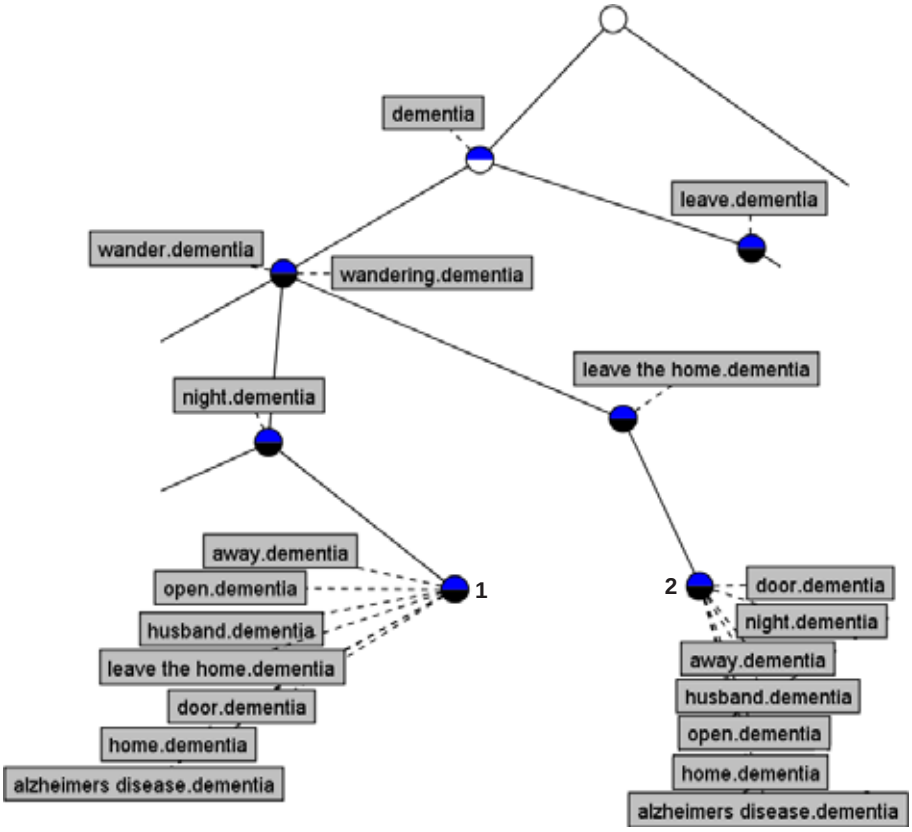


Figure 7.12: Tree-structured Case for the Wandering Sub-problem

7.4 Similarity Measurement of Structured Cases

During problem-solving, SMARTER structures the query by finding in the hierarchy, concepts whose intents contain the query terms. Cases that have concepts in common with the query structure are deemed to have a degree of similarity with the query and are thus retrieved for comparison. In order to obtain this measure of similarity between the query and retrieved cases, pair-wise comparisons are made between query concept nodes and nodes in the retrieved cases. The process is illustrated by use of a simple example in which the structures in Figures 7.13 and 7.14, representing the query and case structures respectively, are compared.

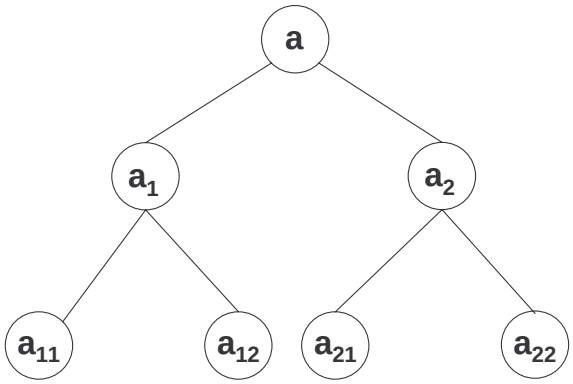


Figure 7.13: Query

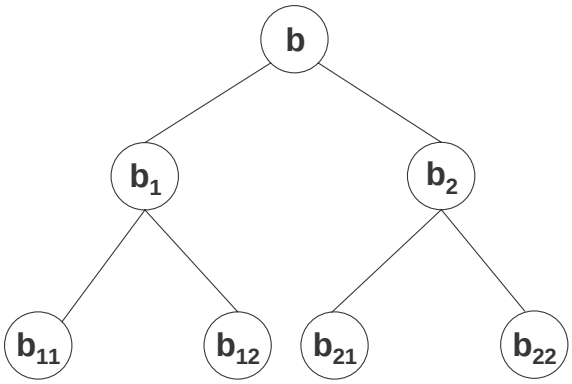


Figure 7.14: Case

Similarity $treeSim(a, b)$, between any two nodes a and b is given by:

$$treeSim(a, b) = 1 - \frac{D(a, ncp(a, b)) + D(b, ncp(a, b))}{D(a, root) + D(b, root)}$$

Where $D(x, a)$ is the distance between x and a, $ncp(x, y)$ is the nearest common parent node of x and y, and $root$ is the root node of the conceptual model. This captures some interesting intuitions of our notion of similarity. For instance, if two nodes are completely dissimilar, then their nearest common parent node will be the root node of the tree and their similarity score will thus be 0. On the other hand, if two nodes are siblings, then the distance to the nearest common parent node will be 1 for each of them. The similarity score will depend on their depth in the structure; the lower they are, the higher the similarity.

Similarity $sim(q, c)$ between any query structure q and a case structure c is given by:

$$sim(q, c) = \frac{\sum_{i=1}^{m} \max_{j=1}^{n} treeSim(q_i, c_j)}{m}$$

where m is the number of query nodes, n is the number of nodes in case c, and $sim(q_i, c_j)$ is the similarity between query node i and the case node to which it is most similar.

Each node in the query structure is compared to every node in the case structure, in turn. For each query node, we only consider the highest similarity score attained when comparing the query with each of the case nodes. Thus only the most similar pairs are considered in determining the overall similarity. Suppose that of all the nodes in Figure 7.14, node a of Figure 7.13 is most similar to node b_{12} and node a_1 is most similar to node b_{22} and so on. The overall similarity is calculated between the two structures by comparing the sum of the individual similarity scores to the maximum attainable score. The equation for similarity between a pair of nodes results in a score of 0 for completely dissimilar nodes and a score of 1 for completely similar nodes. Therefore, the maximum attainable similarity score between any 2 nodes is 1.

7.5 Summary

This chapter has presented SmartCAT-T, a tool that learns a tree-structured conceptual model by applying FCA to sub-problem representations. SmartCAT-T changes the nature of the representations in order to reduce dependencies between objects and subsequently, reduces the occurrence of multiple inheritance in the resulting hierarchy. The remaining occurrences of multiple inheritance are overcome by increasing redundancy in the context table and reapplying FCA to the formal context. The result is a tree structured conceptual model that is then used to create hierarchically structured cases. Structured cases are useful in matching problem descriptors at various levels of abstraction and do not suffer from the effects of multiple inheritance as SmartCAT's cases do. SmartCAT-T is the final version of the case authoring tool and it is the version that is used for the main comparative experiments.

Chapter 8

Evaluation

In this chapter, an experimental evaluation of the techniques developed in this project is presented. The evaluation is carried out using data from the SmartHouse domain and the objective is two fold:

- To establish the effectiveness of the developed techniques in mining semi-structured text for useful information that can be used in problem solving.
- To determine the effect on retrieval, of using hierarchically structured cases instead of flat-structured cases.

In Chapter 3, it was shown that in order for a computer to reasonably interpret text, it would need a language model of the domain in which the texts were written. Consequently, a domain-specific language model from texts in the SmartHouse domain was created in Chapter 3. The language model enabled the interpretation and thus, the extraction of knowledge from free-form texts in order to develop a conceptual model and hence create cases. Furthermore, the language model is used for query interpretation during problem solving. In Chapter 4, Latent Semantic Indexing was employed to highlight key phrases with which to represent the text documents. In Chapter 5, it was demonstrated how a conceptual model can be created by applying Formal Concept Analysis to a context of key phrase attributes and their sub-problem objects. Consequently, a conceptual model was used to create graphically structured cases in Chapter 6 and hierarchically-structured cases in Chapter 7. The use of the case structure to rank retrieved cases was also presented

in Chapter 7. This chapter will now present the evaluation of the SMARTER CBR tool, the different modules that comprise the case authoring tool and the retrieval strategy adopted as a result of the structure of the authored cases.

8.1 Measuring Accuracy

The authored cases are versions of the original documents and are typically stored in textual format. For example, cases where the knowledge is represented as attribute-value pairs may be stored in a database but still the stored strings can be viewed as textual representations of the original documents. Thus, a parameter for measuring retrieval effectiveness in document retrieval systems such as Google is relevant for measuring retrieval effectiveness of cases in a case-based reasoning system. A document would in the SmartHouse domain correspond to a hierarchically structured case as obtained by SmartCAT-T and stored in SMARTER's case base.

A popular way to measure the accuracy of a retrieval mechanism is to measure its *Precision* (P) and *Recall* (R).

$$P = \frac{|Rel \cap Ret|}{|Ret|} \tag{8.1}$$

$$R = \frac{|Rel \cap Ret|}{|Rel|} \tag{8.2}$$

Where, *Ret* and *Rel* are sets of *retrieved* and *relevant* documents with respect to a specific query, respectively. Relevance of a document is typically determined by a human or comparison with documents annotated by a human.

However, it is difficult to retrieve a large number of relevant documents without retrieving a good number of irrelevant ones (low precision) nd thus *Precision* is likely to fall as *Recall* rises. Thus, the *F-score* which is the harmonic mean of *Precision* and *Recall*, is a better judge of a retrieval system's accuracy. The F-score ranges between 0 and 1; it is 0 when *Rel* is 0 and 1 when all retrieved documents are relevant. The F-score (F) helps to determine the best possible measure of accuracy that would be attained at the corresponding values of *Precision* and *Recall*. The F-score is determined using equation 8.3.

$$F = \frac{2PR}{P+R} \tag{8.3}$$

Consequently, the different techniques are compared by taking the F-Measure of the system performance in each experiment.

In the SmartHouse domain, *Precision* is the proportion of SmartHouse devices proposed by the CBR system, that occur in the solution package proposed by the expert; *Recall* is the proportion of devices proposed by the expert that are correctly proposed by the CBR system. *Precision* and *Recall* are typically calculated by considering only a portion of the retrieved cases say, the top N retrieved cases. Thus in the SmartHouse domain, *Ret* is the set of devices in the retrieved cases in the top N slots of the ranking and *Rel* is the set of required SmartHouse devices proposed by the expert as the solution to the query. In this domain, *Recall* is more important than *Precision* because an occupational therapist prefers to be presented with a list of devices to choose from than to have a list of devices that perfectly solve only a part of the problem and so be required to formulate the rest of the solution from scratch. Indeed, as reported by Taylor, Robertson, Wiratunga, Craw, Mitchell & Stewart (2007), the solution proposed by the system should not be simply taken on board.

Complexity of the case retrieval task can be judged by the number of cases the retrieval module has to choose from in order to indentify the correct solution for a given problem. There are 102 cases in SMARTER's case base. Each of the 102 problems is solved by one or more SmartHouse devices (120 solutions in all). An intial evaluation to test the general performance of the SMARTER system and a more detailed evaluation of the developed techniques have been carried out. SMARTER has also been benchmarked against a standard IR tool. The details of the evaluations will be presented in the next sections.

8.2 Initial Evaluation

This evaluation was carried out to test the general performance of the SMARTER CBR system. The test probes were from the same source as the reports that were used to create the SMARTER system and thus are more likely to have the same vocabulary as the case knowedge. It was a general test of how the SMARTER system copes with problems it

has not seen before and it is carried out as a leave-one-out test. 10 cases were randomly picked and used in turn as the probe to the SMARTER CBR system. In each experiment, the probe case was excluded from the creation of the concept hierarchy. The test probes are shown in Table 8.1.

Query	**Features**
1	flooding. ms X be fascinate with water. and she commonly flooded bathroom and kitchen as a result. her activity include stuffing towel and other item down the bowl of the toilet.
2	door opening. when mr. X wish to open his front door he have to project himself forward to reach the door handle and lock. because of poor flexibility in his joint. he find this task to be extremely difficult and physically tiring. also. the positioning of the lock make it both difficult to open the door from the inside upon leave.
3	telephone operation. mrs. X have a sister who live in america and on a regular basis make contact by telephone to check on mrs. Xs health and wellbeing. X be unable to communicate effectively by telephone because of her hearing difficulty and in most instance. both mrs. X and her sister rely upon mr. X to relay message between the two. this have become a cause of great frustration and upset for both X and her sister. mr. X also become upset and frustrate by the lack of communication between his wife and her sister. especially as he be have to act as gobetween.
4	windows. in order to open window. miss X have to rely on care staff when available. this be because of the size and the weight of the window. they be of the sash and case type. and because of ms fultons lack of mobility due to her spinal problem.
5	operate set central heating control. mrs X report that she find it impossible to set or alter the setting of her central heating time clock. change from winter to summer time. she rely on her daughter and son to carry out such task for her.
6	wandering. ms X have start to leave her flat at odd hour. day and night. and whilst she could at time find her way to the staff resource facility. it be feel there be a risk that she may leave the site and get lost or become injured.
7	window operation ms X window be of the type that hinged at the bottom and have the opening handle at the very top. more than 2 metre above the ground. she find it impossible to be able to open and close these window herself. rely on care worker to do this for her.
8	telephone operation. as with her intercom. mrs X have difficulty use her telephone. she have problem hear caller. so use it infrequently.
9	door operation. mr. X find it difficult to negotiate door opening and closing. and operate doorlock. whilst manage his walking stick. also. the height of the doorhandle and lock make they inaccessible. the passage in mr. rosss home be narrow and this also add to the difficulty when it come to use the door. this be because when mr. ross be attempt to open the door from the inside. there be little way for he to get close enough to the lock and handle. whilst still hold on to his stick.
10	intercom operation. the intercom handset for mrs. X be locate right next to her front door. this mean to answer it to caller. who be buzz at the main door. she have to get from her chair with help of her stick. walk through to the hall. and then pick up the handset. by the time she have carry out this operation. more often than not the visitor have go. as well as this. the caller be quite often wish to get access to the building to visit someone else. which mrs. X find most frustrating.

Table 8.1: Lemmatised Test Probes for the Leave-One-Out Test

A comparison of an expert's solutions and those suggested by SMARTER for the 10

different leave-one-out test runs is shown in Table 8.2. It shows the general performance when the top 3 cases suggested by SMARTER are considered. The number of true positives (TP), false positives (FP) and false negatives (FN) are shown. In the SmartHouse domain, TP is the number of devices in the retrieved cases that also appear in the expert solution. FP is the number of devices that are in retrieved cases but not in the expert's solution. FN is the number of devices that are in the expert's solution but are missing in the retrieved cases. The solutions suggested by SMARTER, which match with expert-suggested solutions, are marked in bold. Precision and Recall can also be defined using equations 8.4 and 8.5.

$$P = \frac{TP}{TP + FP} \tag{8.4}$$

$$R = \frac{TP}{TP + FN} \tag{8.5}$$

Table 8.3 shows values for Precision, Recall and F-Scores as calculated using equations 8.4, 8.5 and 8.3 respectively.

The SMARTER system correctly recommends a *flood detector* for the *flooding* problem of query 1. However, the only other *flood detector* in the case base was recommended for a person with *Dementia* and not *Learning difficulties* which is the disability for the person in query 1. Thus this second *flood detector* does not appear in the top 3 devices.

Although the two people in the previously solved problems face the same (*flooding*) problem, the reasons behind the problem are dependent on the person's disability. Consequently, the second device although correct, appears with a much reduced ranking than it would if the corresponding person had the same disability as the one in the query. However, it is interesting to note that the devices ranking 2nd and 3rd (*paging unit*, *out of bedroom alert*, *out of house alert*) although do not solve the specific *flooding problem*, are very useful for a person with learning disabilities, who happens to exhibit the same symptoms.

In the second query, the first 2 devices match those suggested by the expert. The third device does not solve the door opening problem but is very likely to be useful for a person with a mobility problem, whose degree of disability is such that they are unable to open their doors unaided.

Query	TP	FP	FN	Top 3 Suggested Devices	Correct Device
1	1	3	0	**flood detector** paging unit,out of bedroom alert out of house alert	flood detector
2	2	1	1	**door opener** **door opening motor** window control	door opener,movement sensitive lighting
3	2	1	0	**telephone amplifying unit** visual doorbell **amplifying headset**	telephone amplifying unit
4	3	0	0	**window controls** **window control** **window control**	window openers
5	2	1	1	**registration pip** **text image magnifier** heat extreme sensor	registration pip,push button clock
6	3	0	1	**door sensor** **property exit sensor** **a bed occupancy sensor**	wandering alert
7	2	1	0	door opener **window controls** **window controls**	window control
8	2	1	0	visual doorbell **telephone amplifying unit** **television amplifier**	telephone amplifying unit
9	3	0	0	**door opening motor** **handsfree intercom unit** **door opener**	door opener
10	3	0	0	**handsfree intercom** **handsfree intercom unit** **door opener**	intercom operation
Total	23	8	3		

Table 8.2: A Comparison of SMARTER and Expert Suggested Solutions

In query 3, the solutions ranking 1st (*telephone amplifying unit* and 3rd *amplifying headset*) are both right since a person who requires a *telephone amplifying unit* will also need an *amplifying headset* to listen to radio or television. The second device is correct in that a person with a hearing problem will need a *visual door bell* because they will be

Query	Precision	Recall	F-Score
1	0.25	1.00	0.40
2	0.67	0.67	0.67
3	0.67	1.00	0.80
4	1.00	1.00	1.00
5	0.67	0.67	0.67
6	1.00	0.75	0.86
7	0.67	1.00	0.80
8	0.67	1.00	0.80
9	1.00	1.00	1.00
10	1.00	1.00	1.00

Table 8.3: Precision, Recall and F-Scores for 10 Leave-One-Out Test Runs for 3NN

unable to hear the buzzer when it sounds.

The 3rd recommended device in query 5 looks out of place at first glance. However, visually impaired people may need some fire protection in case they accidentally turned the cooker on at a level that is much higher than they had intended and consequently caused what they were cooking to be subjected to heat that is enough to cause a fire say, in the case of overheating oil.

In query 6, the *door*, *property exit* and *bed occupancy* sensors are all to ensure that the person's whereabout are known in case they wander. These three devices were ranked higher than *wandering client monitor*, which was ranked 4th. The only reason for this is the fewer numbr of terms the problem text corresponding to the *wandering client monitor* had in common with the query case. However, its similarity score was *10.44* which is comparable to a *bed occupancy sensor*'s (*10.79*) and not too far from *property exit sensor*'s which was *17.74*.

Generally, the SMARTER system performs remarkably well for the problems that have not been used to create the conceptual model and are not included in the case base. This is demonstrated by the high number of true positives and low number of false positives. Moreover, some of the false positives are still very useful recommendations for people exhibiting the same symptoms as the person in the query. An average F-Score of *0.80* is obtained for the 10 queries. This is very good performance because expecting the correct

solutions to appear in the top 3 ranks is quite demanding of the tool's effectiveness, as has been done here by considering only the top 3 retrieved solutions in the calculations for Precision, Recall and the F-Scores. This good performance can be attributed to the fact that case knowledge was mostly captured in key phrases, using a technique that also captures some semantics. Furthermore, the retrieval mechanism puts phrase information into account because key phrases provide more context than key words and thus a phrase that is key is more likely to be important in that given context than a key word. SMARTER deems overlapping phrases between queries and cases to be more significant indicators of similarity than overlapping single word terms.

8.3 Further Evaluation

A further evaluation was carried out to benchmark SMARTER and to analyse the role of the different modules on the effectiveness of the case authoring tool. The SMARTER CBR system was also tested on how well it would perform when required to make new predictions for data that was obtained from a completely different source and was thus likely to be unfamiliar. This is because in the real world, the CBR system would be expected to work well without forcing the user to adapt their queries to phrases that the tool has stored. Thus in this part of the evaluation, the test probes used are from new SmartHouse reports obtained from a different source from the ones used in the project. The new reports were produced seven years after the set that has been used throughout the project. Indeed, the new reports were written by an occupational therapist while the old ones were written by a knowledge engineer. Thus, finding solutions to these probes is a much more challenging task than the leave-one-out test. It is recognised that the proposed solutions may sometimes be simpler/outdated in comparison with solutions suggested in the new reports and therefore, the expert will be required to verify if these are still reasonable solutions. The test probes are shown in Figure 8.1.

Query	Features
1	mobility. window opening, door opening, spinal problem, wheelchair user, unable to bathe independently
2	mobility. risk of slip fall particularly when transfer unable to call for assistance (**help**) at all time when in house (**home**) environment ; Home environment get too hot , due to difficulty in control heating or be able to open window
3	mobility. Get trap in bathroom/kitchen area either due to carer not leave door ajar in correct position or due to Mrs need access to fridge/freezer and door become fix in shut position
4	mobility. Currently Mr X do not have the facility to call for assistance (**help**) from his lounge due to a black spot in existing transmitter . Mr X be liable to fall when transfer particularly in the toilet . Mr X be unable to open window without risk of fall, should Mr X fall Mrs X attempt to lift he without assistance (**help**) and already suffer from low back pain , both Mr and Mrs X be reluctant to request assistance (**help**), try to complete task on their own
5	mobility. Mrs X be leave for significant period of time on her own and do not have the physical capacity to operate current call system ; the front door to the tenancy be leave open to allow for ease of access for carer, and minimise security of property; Mrs X be unable to control all aspect of her environment , be leave watch television during her waking hour , without option (**choice**) of her entertainment . Mrs X be socially isolate be unable to communicate use the telephone , be dependent on carer and her husband ; Mr have a significant problem with use of alcohol , which have in past lead to inability to care for his wife and flooding of the flat . Mrs be liable to opportunistic infection , Mrs have a low appetite and require following a specific diet , which her husband have difficulty in adhere to
6	poor vision (**poor sight**). hear. Mr be at risk of stumble and fall within his house (**home**) and external environment ; Mr be at risk of flooding his house (**home**), be unable to hear the water flow and have leave the plug in the sink ; due to hear difficulty Mr would have significant difficulty in be alert to fire , especially at night
7	mobility. unable to control temperature in environment result in hot stuffy flat
8	ataxia. mobility. at risk of fall , particularly within the house (**home**) environment and on occasion in the shower ; issue with home (**home**) security , current system have result in misinterpretation and police interest ; at risk of damaging property due to physical difficulty in driving his wheelchair ; on occasion be leave for long period without support and have limit access to call system due to not wear pendant .
9	mobility. cognition. lack of security to property leave Ms X very vulnerable during the day ; front door of the property be lock by staff at night potentially create difficulty of access in the event (**case**) of fire ; Ms X be unable to use her current alert system , especially when tired and cognitive difficulty be more apparent ; Ms X have limit option (**choice**) regarding use of her environment , result in thing be do to she. this have lead to significant frustration and anger on Ms rs part .
10	mobility. Slip fall when transfer ; Vulnerable to intruder due to leave back door open during day time, when in her house (**home**) ; unable to control aspect of environment to alter (**change**) temperature such as , heating or opening window .

Figure 8.1: Lemmatised Test Probes from New SmartHouse Reports

The following methods are employed in the various comparisons:

- Lucene on whole documents (*Lucene*);
- Lucene on SMARTER's case knowledge i.e., key phrase vocabulary (*Lucene-plus*);
- SMARTER's retrieval module on SMARTER's structured cases with the following combinations:

    1. Retrieval carried out on cases created from unharmonised documents, using unharmonised queries (*U*-SMART-*U-Query*)
    2. Retrieval carried out on cases created from harmonised documents, using unharmonised queries (*H*-SMART-*U-Query*)
    3. Retrieval carried out on cases created from harmonised documents, using harmonised queries (*H*-SMART-*H-Query*)

The results of each experiment are discussed in the corresponding sections. The number of relevant cases and required devices for each query is shown in order to give an indication of how Precision and Recall have been calculated. The number of relevant cases is denoted as *Rel Cases* in the tables.

8.3.1 Calculating Precision and Recall

Precision and *Recall* are typically calculated on a portion of the retrieved cases. However, good *Recall* is obtained at the expense of *Precision* and the reverse is also true. Therefore, when the number of retrieved cases on which *Precision* and *Recall* are calculated is increased, *Recall* is likely to improve but *Precision* is likely to suffer. Thus it is important to strike a reasonable balance when deciding on the number of retrieved cases on which to calculate the system's performance.

The average number of required SmartHouse devices for the test probes in this part of the evaluation is 5. However, a SmartHouse device or device type could be a solution to a number of problems (in the case base) that differ in description, level of disability, and so on. Thus the same device type may appear a number of times in a ranking in response to a query because the (different) problems with which it assists are similar to

the query in some way(s). Consequently, even for a perfect system, it will be difficult for the top 5 slots of the ranking to be occupied by cases whose solutions comprise **all** the required devices for the query. Hence *Precision*, *Recall* and the F-Score are calculated on the top 12 ($\approx$ 5required devices $\times$ 2.5) retrieved cases in order to cater for device-type re-occurences. It is understood that this number may be large (good for *Recall* and bad for *Precision*) for some queries and too small for others (good for *Precision* and bad for *Recall*). However, the experiments are also compared by determining the F-Score for each query since the aim is to balance *Precision* and *Recall*.

8.3.2 Benchmarking SMARTER

SMARTER's retrieval mechanism is benchmarked against Lucene. This is a comparison of SMARTER in its default state i.e., the case knowledge as obtained with the SmartCAT-T system and utilising the case structure in retrieval, against using whole documents as case knowledge and Lucene as the tool for retrieval. Lucene treats each document as a bag-of-words and relies on frequency information of query terms in the document and document collection as a whole. SMARTER structures the query before making pair-wise comparison between nodes in the query structure and those in the structure of each activated case. The test is a comparison of *H-SMART-H-Query* and Lucene. Table 8.4 shows the obtained Precision and Recall values and Figure 8.2 shows a comparison of the F-Scores.

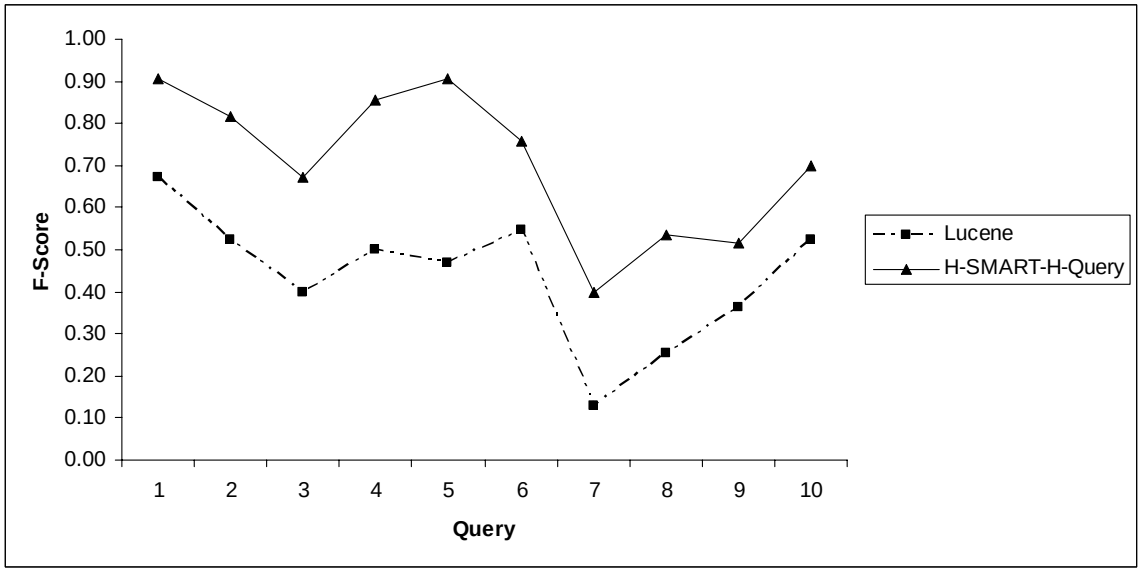


Figure 8.2: SMARTER Vs Lucene on Whole Documents

Query	Required Devices	Relevant Cases	Precision		Recall	
			H-SMART-H-Query	Lucene	H-SMART-H-Query	Lucene
1	3	21	0.83	0.67	1.00	0.67
2	5	29	0.83	0.75	0.80	0.40
3	3	17	0.67	0.50	0.67	0.33
4	4	18	0.75	0.50	1.00	0.50
5	5	33	1.00	0.33	0.83	0.80
6	10	32	0.83	0.50	0.70	0.60
7	3	18	0.25	0.08	1.00	0.33
8	4	21	0.58	0.17	0.50	0.50
9	3	25	0.42	0.25	0.67	0.67
10	5	32	0.83	0.75	0.60	0.40

Table 8.4: SMARTER's Structured Cases Vs Lucene on Whole Documents

SMARTER outperformed Lucene on all the queries for *Precision*, *Recall* and F-Score with an average F-Score of 0.71 in comparison to Lucene's score of 0.44. In both instances, the documents and the queries are harmonised. Therefore, this difference in performance can be attributed to the fact that SMARTER makes use of the case structure to compare cases at different levels of abstraction and Lucene only relies in word overlaps between the query and a case. Another contributing factor could be because Lucene makes use of all words in the documents and only relies on their statistical information to determine their importance and may therefore not be as focused on problem-solving as SMARTER which only makes use of words and phrases that SmartCAT-T had deemed to be important by relying on semantic information. Lucene is more likely than SMARTER to rank an important word as low, if that word is not appropriately supported by statistical information in the cases' text. SMARTER and Lucene perform poorest with query 7. This is because there were not enough words to describe the problem and so SMARTER and Lucene had to make do with the few words that were available. However, SMARTER's ability to relate the query to concepts whose intents comprise terms that may not be in the query, enabled SMARTER to expand the query and thus find more relevant cases than Lucene.

8.4 Testing the Different SMARTER Modules

The different modules that comprise SMARTER were evaluated in order to determine their contribution to the performance of the CBR tool. This involved evaluation of the modules comprising the SmartCAT-T case authoring tool, and the retrieval strategy adopted by the SMARTER CBR tool. The results of experiments to determine these individual contributions are presented in the next sub-sections.

8.4.1 The Effect of a Hierarchical Structure on Retrieval

This is a comparison of retrieval effectiveness when the cases' hierarchical structure is used for retrieval and when cases are used as flat structures. This test can also help to determine the goodness of the case structure and the conceptual model.

It is a little difficult to compare the structured cases with their corresponding flat-structured ones using SMARTER's retrieval mechanism. This is because SMARTER uses word overlaps between the queries and cases in conjunction with the case structures to obtain a final ranking for the retrieved cases. Thus applying SMARTER to both tree structured and flat structured cases would favour the tree structured cases and would thus result in an unfair comparison.

Lucene is a sophisticated IR tool and indeed, SMARTER does not apply any sophisticated weighting mechanism based on the word overlaps between the query and the cases as Lucene does. Thus, comparing retrieved cases using SMARTER with those obtained using Lucene is not a very fair comparison but since Lucene has the more sophisticated word-based technique, this test can still help us to judge the usefulness of using the case structure for retrieval. Consequently, the test is carried out by comparing SMARTER's retrieval on structured cases with Lucene's retrieval on case representations (*Lucene-plus*), for the same retrieval tasks. The Precision and Recall results of this comparison are shown in Table 8.5 and the F-Scores are shown in Figure 8.3. *Lucene-plus* is Lucene applied to case representations which are obtained from harmonised documents. Consequently, the comparison is made with cases obtained from harmonised documents and the queries are harmonised in each instance. This is a comparison of *H-SMART-H-Query* and *Lucene-Plus*.

Query	Required Devices	Relevant Cases	Precision		Recall	
			H-SMART-H-Query	Lucene-Plus	H-SMART-H-Query	Lucene-Plus
1	3	21	0.83	0.58	1.00	1.00
2	5	29	0.83	0.58	0.80	0.60
3	3	17	0.67	0.42	0.67	0.67
4	4	18	0.75	0.42	1.00	1.00
5	5	33	1.00	0.50	0.83	0.80
6	10	32	0.83	0.58	0.70	0.40
7	3	18	0.25	0.17	1.00	0.33
8	4	21	0.58	0.33	0.50	0.75
9	3	25	0.42	0.50	0.67	0.33
10	5	32	0.83	0.50	0.60	0.40

Table 8.5: SMARTER's Structured Cases Vs Lucene on Case Representations

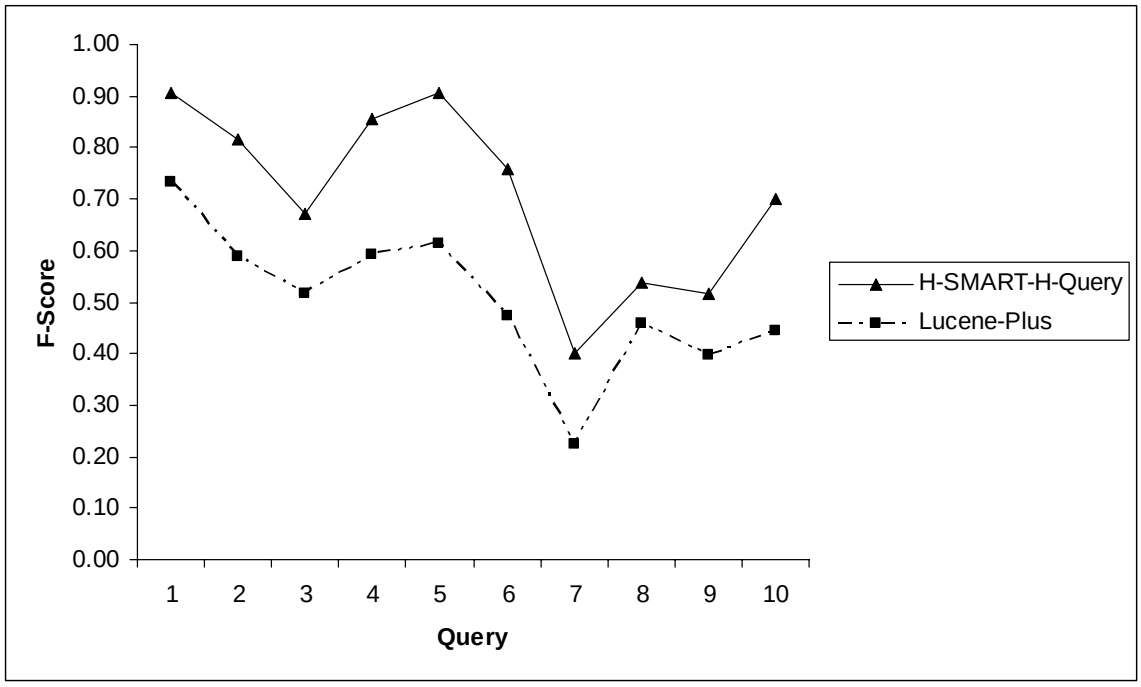


Figure 8.3: SMART on Structured Cases Vs Lucene on Case Representations

Better F-Scores, *Precision* and *Recall* (except for query 8) are obtained in all test probes when SMARTER is used for retrieval than when Lucene is utilised in retrieving the case representations. The structured cases result in an average F-Score of 0.71 while Lucene-plus has an average of 0.50. The flat and tree structured cases have the same case

content and thus the difference in performance of the two techniques can only be attributed to SMARTER's taking into account the relationships between concepts instead of using a bag-of-words approach, when calculating similarity. Lucene-Plus and *H-SMART-H-Query* both fair badly on query 7 again because the query was not described in enough detail.

8.4.2 Effectiveness of the Key Phrase Extraction Technique

The effectiveness of the key phrase extraction approach is evaluated by determining if the cases contain as much knowledge that is relevant for problem-solving, as the original documents. The key phrase extraction task essentially gets rid of terms that it considers unimportant and thus not useful for problem solving. Thus it is important to determine as to whether the phrases that are left as case knowledge contain knowledge that is enough to solve future problems.

Since this is a test of the knowledge inherent in the case representations compared to that in the whole documents, Lucene is used to carry out the ranking using the same queries in each instance. Hence Lucene is first applied to whole documents and then to case representations; the representations comprise key phrases extracted in Chapter 4. Thus it is a comparison of *Lucene* and *Lucene-Plus*. Table 8.6 shows Precision and Recall values and Figure 8.4 shows the obtained F-Scores for this experiment.

Query	Required Devices	Relevant Cases	Precision		Recall	
			Lucene	Lucene-Plus	Lucene	Lucene-Plus
1	3	21	0.67	0.58	0.67	1.00
2	5	29	0.75	0.58	0.40	0.60
3	3	17	0.50	0.42	0.33	0.67
4	4	18	0.50	0.42	0.50	1.00
5	5	33	0.33	0.50	0.80	0.80
6	10	32	0.50	0.58	0.60	0.40
7	3	18	0.08	0.17	0.33	0.33
8	4	21	0.17	0.33	0.50	0.75
9	3	25	0.25	0.50	0.67	0.33
10	5	32	0.75	0.50	0.40	0.40

Table 8.6: Case Representations Vs Whole Documents

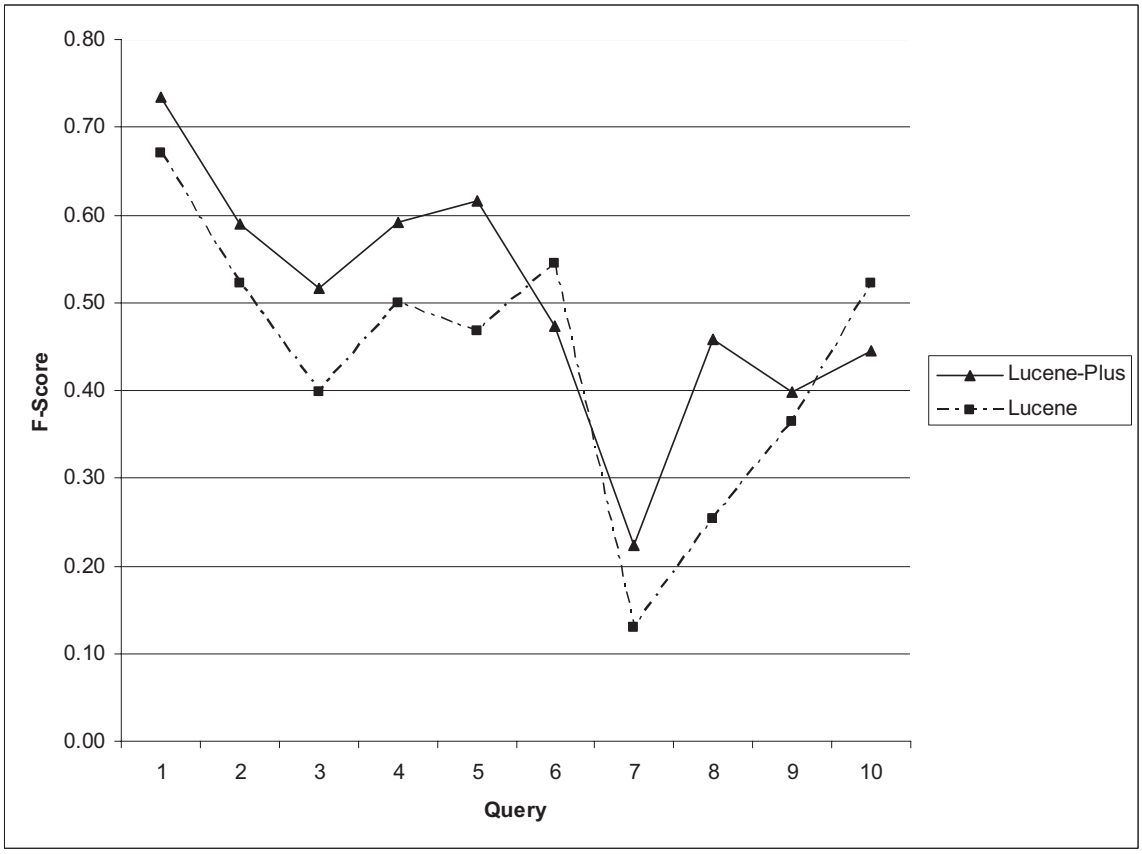


Figure 8.4: Lucene Vs Lucene-Plus

Lucene-plus shows better performance over Lucene in 8 of the 10 queries for *Precision*, *Recall* and F-Score, and has an average F-Score of 0.50 in comparison to Lucene's 0.44 for all the queries. This shows that for the 8 queries where Lucene-plus performed better, the case content is more focused to problem-solving, which in turn results in more effective retrieval than the original whole documents.

Lucene-Plus has better *Precision* than Lucene for query 6, having obtained 7 devices in cases occupying the top 12 slots whereas Lucene had 6 devices. However, Lucene-Plus had only 4 of the 10 devices required for this query but Lucene had 6 (required devices) thereby achieving better *Recall* and consequently, a better F-Score than Lucene-Plus. Lucene-Plus and Lucene have the same *Recall* for query 10. However, Lucene had 9 devices in retrieved cases occupying the top 12 slots but Lucene-Plus had 6 devices and consequently obtained a lower *Precision* and F-Score than Lucene. The reason for Lucene-Plus' poorer performance for these two queries could be that, the appropriate case

representations needed to have had a few more words in order to rank the relevant cases higher than was done. Lucene-Plus did flag up the relevant cases as important but their ranking was lower than what was taken into account when calculating *Precision*, *Recall* and F-Scores for this evaluation.

8.4.3 Effectiveness of Harmonisation

The purpose of this experiment is to establish the effect of using a uniform vocabulary on case authoring and on query processing. The experiment is essentially to evaluate the contribution of text harmonisation presented in Chapter 3. In Figure 8.1, The terms in bold are alternative (harmonised) versions of the preceding ones.

The Effect of Harmonisation on Key Phrase Extraction

In order to establish if harmonisation has an effect on the techniques used to acquire case knowledge, the test probes are first used on cases that have been created using the original non-harmonised documents and comparing results with those obtained when harmonised documents are used to create the cases. This experiment is carried out to determine the contribution of harmonisation (if any) to feature extraction by LSI. Thus in the first part of this evaluation, key phrases are extracted from original unharmonised data, these are used to create the conceptual model and the structured cases. In the second part of the evaluation, the documents are harmonised before key phrase extraction, the creation of a conceptual model and case authoring.

Table 8.7 shows results for Precision and Recall and Figure 8.5 shows a comparison of the F-Scores. *H-SMART-H-Query* shows results obtained when the documents were harmonised before feature extraction and case creation were carried out. *U-SMART-H-Query* shows results obtained when case knowledge is extracted from the original documents that have not been harmonised. In both instances, the queries were harmonised and as such, words forms that were most common in the vocabulary were used to replace query words where this was possible.

H-SMART-H-Query has better F-Scores than *U-SMART-H-Query* for all queries except query 8 where *U-SMART-H-Query* performs better. The two test methods achieve the same *Recall* scores for query 8 but *U-SMART-H-Query*'s top 12 retrieved cases com-

Query	Required Devices	Relevant Cases	Precision		Recall	
			U-SMART-H-Query	H-SMART-H-Query	U-SMART-H-Query	H-SMART-H-Query
1	3	21	0.83	0.83	0.67	1.00
2	5	29	1.00	0.83	0.60	0.80
3	3	17	0.58	0.67	0.33	0.67
4	4	18	0.50	0.75	0.50	1.00
5	5	33	1.00	1.00	0.80	0.83
6	10	32	0.92	0.83	0.30	0.70
7	3	18	0.08	0.25	0.33	1.00
8	4	21	0.75	0.58	0.50	0.50
9	3	25	0.42	0.42	0.33	0.67
10	5	32	1.00	0.83	0.40	0.60

Table 8.7: Harmonised Vs Unharmonised Documents

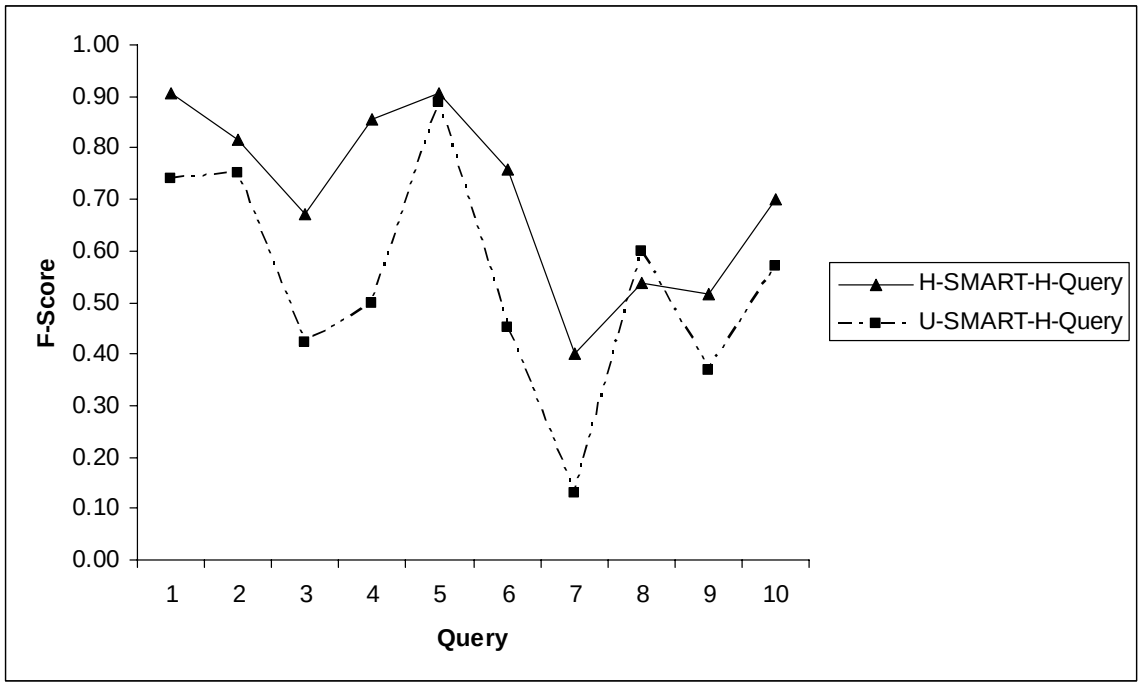


Figure 8.5: Harmonisation on Feature Extraction

prise two relevant cases more than *H-SMART-H-Query* hence achieving better *Precision*. *H-SMART-H-Query* has an average F-Score of 0.71 compared to *U-SMART-H-Query*'s score of 0.54. The difference in performance can be attributed to the fact that more key phrases are indentified when LSI is applied to terms in harmonised documents than when

it is applied to terms in original unharmonised documents. Harmonisation attempts to ensure that one word form is always used to refer to synonymous words thus creating a uniform vocabulary. LSI relies on co-occurence patterns of words to infer similarities between them. Thus, for large document collections, synonymous words have a "chance" to be seen as similar by occuring in the same documents. These word associations are not so obvious for small document collections. Thus, in this project, LSI had to be 'helped' to see similar words as similar (by restricting the vocabulary) and consequently, mine associations that were inherent in the document collection. The consequence of this action is evidenced from the better results obtained when harmonised documents are utilised in obtaining case knowledge.

The Effect of Harmonisation on Query Processing

In this experiment, harmonised and unharmonised queries are used in order to determine the effect of harmonisation on retrieval effectiveness. In order to harmonise the queries, 7 of the 10 queries were altered to ensure that the most common word forms were used. The changes made were very few because some of the query text was unfamiliar but nevertheless, a comparison of the results with harmonised and unharmonised queries will help to determine the effect of harmonising query terms, on retrieval effectiveness.

The obtained *Precision* and *Recall* values are shown in Tables 8.8 and 8.9 and the F-Scores are shown in Figures 8.6 and 8.7. Table 8.8 and Figure 8.6 shows results obtained for a comparison of retrieval effectiveness when cases are created from **harmonised** documents and the queries are harmonised and then unharmonised for each experiment. Thus it is a comparison of *H-SMART-H-Query* and *H-SMART-U-Query*. Table 8.9 and Figure 8.7 shows results obtained for a comparison of retrieval effectiveness when cases are created using original **unharmonised** documents, and the queries are harmonised and then unharmonised for each experiment; the experiments are referred to as *U-SMART-H-Query* and *U-SMART-U-Query* respectively.

Results show that when the documents are harmonised, SMARTER achieves an average F-Score of 0.71 for harmonised queries and 0.59 for unharmonised queries. This is considerably better than when the documents are unharmonised where the F-Scores for harmonised and unharmonised queries are 0.55 and 0.50 respectively. *H-SMART-H-*

Query	Required Devices	Relevant Cases	Precision		Recall	
			H-SMART-U-Query	H-SMART-H-Query	H-SMART-U-Query	H-SMART-H-Query
1	3	21	0.83	0.83	1.00	1.00
2	5	29	0.75	0.83	0.40	0.80
3	3	17	0.67	0.67	0.67	0.67
4	4	18	0.42	0.75	0.50	1.00
5	5	33	0.83	1.00	0.83	0.83
6	10	32	0.75	0.83	0.70	0.70
7	3	18	0.25	0.25	1.00	1.00
8	4	21	0.42	0.58	0.50	0.50
9	3	25	0.42	0.42	0.33	0.67
10	5	32	0.92	0.83	0.40	0.60

Table 8.8: Harmonised Vs Unharmonised Queries - Harmonised Case Knowledge

Query	Required Devices	Relevant Cases	Precision		Recall	
			U-SMART-H-Query	U-SMART-U-Query	U-SMART-H-Query	U-SMART-U-Query
1	3	21	0.83	0.83	0.67	0.67
2	5	29	1.00	0.83	0.60	0.40
3	3	17	0.58	0.58	0.33	0.33
4	4	18	0.50	0.33	0.50	0.50
5	5	33	1.00	1.00	0.80	0.80
6	10	32	0.92	0.92	0.30	0.30
7	3	18	0.08	0.08	0.33	0.33
8	4	21	0.58	0.50	0.75	0.50
9	3	25	0.42	0.42	0.33	0.33
10	5	32	1.00	1.00	0.40	0.40

Table 8.9: Harmonised Vs Unharmonised Queries - Unharmonised Case Knowledge

Query has better F-Scores than *H-SMART-U-Query* in all instances when the queries are harmonised. A similar pattern can be seen when the comparison is made using cases created from unharmonised documents; cases from unharmonised documents lead to better F-Scores for 3 queries and there is a tie in 4 of the instances when the queries are harmonised. This means that although the words that were harmonised were few, they

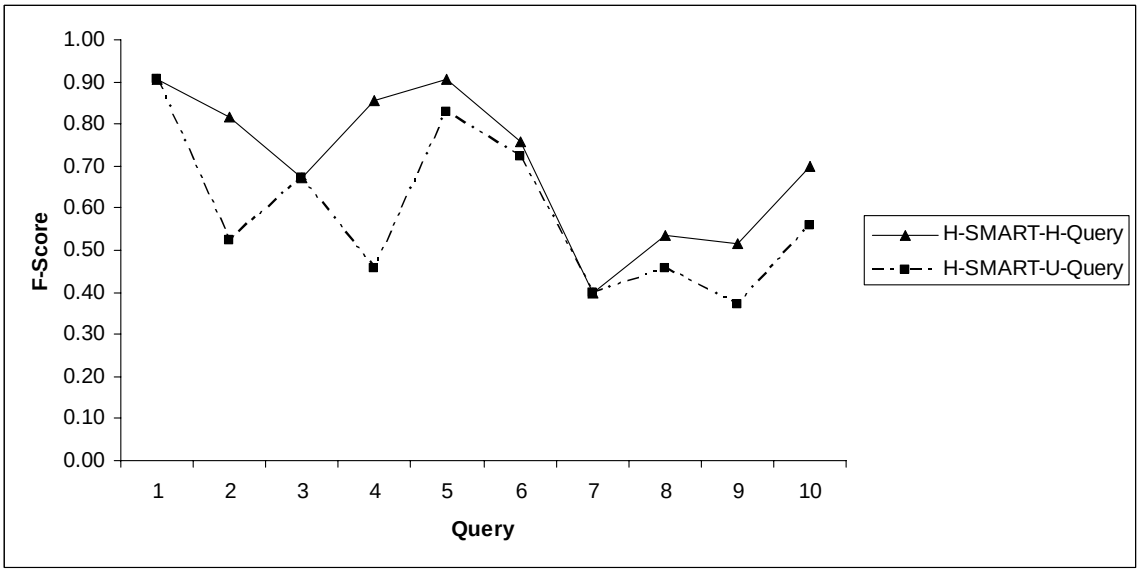


Figure 8.6: Harmonisation on Retrieval - Harmonised Documents

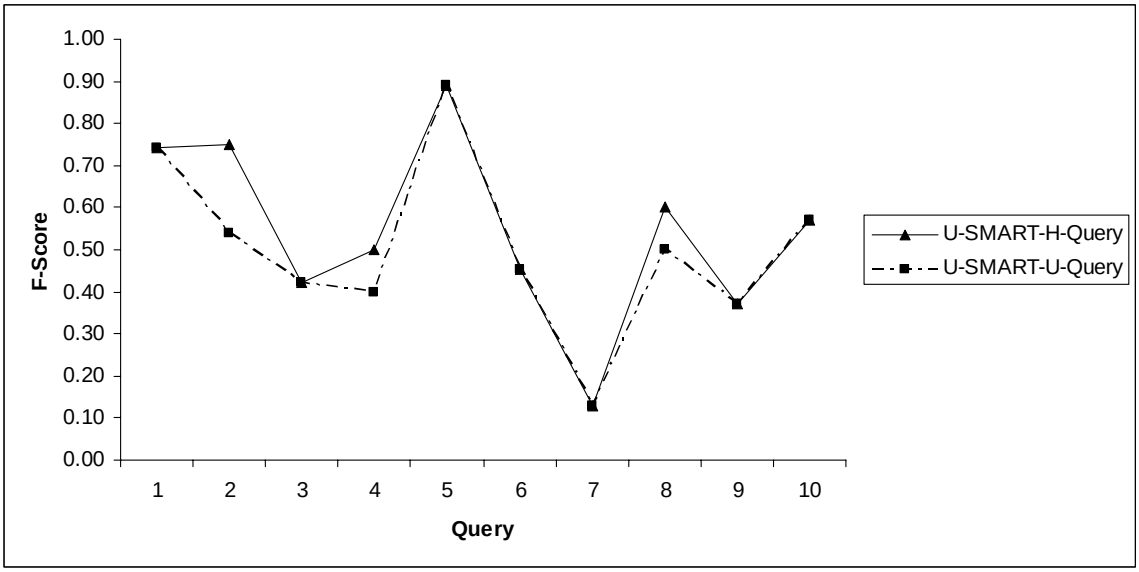


Figure 8.7: Harmonisation on Retrieval - unHarmonised Documents

were important and were thus instrumental in the activation of concepts in cases that are similar to the queries. When unfamiliar words were used instead of the harmonised versions, the cases in which they are important were identified but with a reduced ranking.

8.5 Overall Analysis

Figure 8.8 shows a comparison of F-Scores obtained for all the test regimes. It can be seen that *H*-SMART-*H-Query* outperformed all the other data-query combinations, for all queries except query 8. *H*-SMART-*H-Query* has an average F-Score of 0.71 and Lucene was the worst with an average F-Score of 0.44. This is mainly because hierarchically structured case representations allow for effective similarity matching even when cases are activated in a naïve way such as the bag-of-words approach used in these experiments. Comparing structures the way SMARTER does ensures that structures that are different are interpreted as such; a *mobility-related* case will get a similarity score of 0 when compared to a case regarding *hearing* problems since nodes in the two cases will have the root node as their nearest common parent.

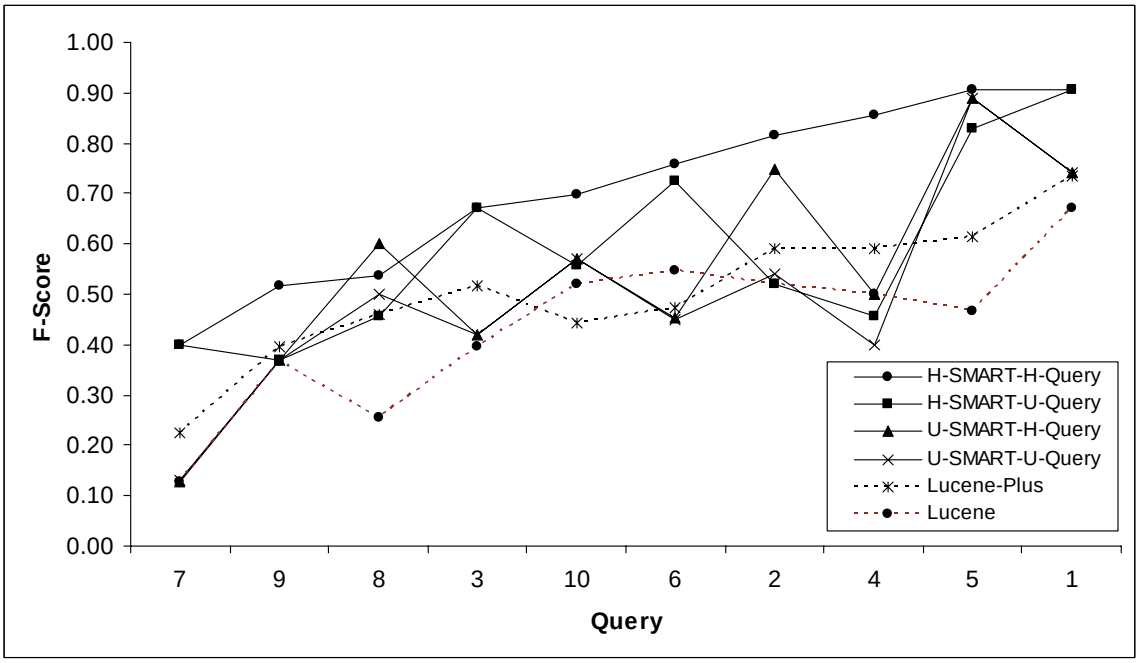


Figure 8.8: Comparison of all Combinations

Lucene-plus performs better than Lucene for most of the queries. This shows that the case representations generally contain as much useful knowledge as the original documents and in many instances, are more focused to problem-solving than whole documents. This further shows that the feature extraction technique is effective at identifying those features that are important in the domain.

Query 7 did not contain much of the problem detail. Thus the query was a challenge for all test regimes. Generally, better results are obtained with harmonised case knowledge and queries than with unharmonised case knowledge and queries. This can be attributed to the fact that more key phrases were identified when the documents were harmonised than when they were not. The cohesion created by harmonisation allows LSI to identify more key phrases because of the now enhanced degree of co-occurences. The identified key phrases in turn, lead to activation of concepts to which they are part of the intent. Thus more useful cases can be retrieved during problem solving.

Better results are obtained for the Leave-one-out test than for the new problems. This is partly because the test probes used in the Leave-one-out test use vocabulary that is familiar to SMARTER's case retrieval engine where as some of the vocabulary used in the queries is unfamiliar. Unfamiliar vocabulary in the new test problems could also be the reason why sometimes the difference in results obtained with harmonised queries is neglibly small; when the bulk of the text is unfamiliar, the use of a few familiar words may not make much difference on retrieval effectiveness. Nevertheless, good results are obtained which means that important terms such as disability-terms or problem-focused terms such as *door opening*, will be repeated in some fashion regardless of when or by whom reports are created. Hence the CBR system is able to find cases with similar knowledge even when some of the query terms are unfamiliar.

8.6 Summary

This chapter has presented an evaluation of the general performance of the SMARTER CBR system and benchmarked the system on a high performance IR tool Lucene. it has also presented an analysis of the contribution of the various modules to the overall performance of the SMART case authoring tool and of SMARTER's retrieval mechanism.

The results have shown SMARTER to perform remarkably well for new problems. Indeed, its performance was shown to be superior to that of Lucene even when concepts were activated in a naïve way where only overlapping words between the queries and case knowledge were considered. Results from further experiments have shown that the key phrase extraction technique was effective in identifying phrases that contain knowl-

edge that is useful for problem solving. Furthermore, better results were observed when case knowledge was obtained after the text was harmonised. This is because LSI employs co-occurrence patterns among terms in the document collection in order to identify key features inherent in the text. Text harmonisation makes those co-occurrences more aparent and consequently results in the identification of more key phrases than when the text is not harmonised.

Retrieval effectiveness depends on the ability of the retrieval engine to associate/match a query term with the ones used as case knowledge. The results show a slight improvement when harmonised query words are used. This can be attributed to the fact that word-synonyms are interpreted as such enabling the system to treat similar terms that are different strings, the same. Thus, high gains could potentially be obtained when the system is in use as there would be many more query words whose nature cannot be known apriori but which the system is designed to be able to handle with the use of the synonym mapper.

The experiments also illustrate the superiority of similarity measures that take into account relationships between concepts, over bag-of-words approaches. This is the reason behind SMARTER's ability to recommend more sensible solutions than IR-based engines such as Lucene.

In the next chapter, an experimental evaluation is carried out to establish how generic the developed approaches are by subjecting them to a domain other than the SmartHouse. The techniques are applied to data in the domain of air and marine safety investigation obtained from the Canadian Transport Safety Board (TSB 2007). Adjustments that are required in the application of the techniques in the domain of air and marine safety investigation are highlighted.

Chapter 9

Application of SmartCAT-T to a Different Domain

In the previous chapter, the SmartCAT-T case authoring tool was evaluated by testing the adequateness of the authored cases for problem-solving in the SmartHouse domain. SmartCAT-T was benchmarked against a high-standard IR tool and the modules comprising SmartCAT-T were also evaluated in order to determine their individual contributions to SmartCAT-T's effectiveness. The retrieval mechanism adopted in the resulting SMARTER CBR tool was also evaluated.

It is important to know how generic the developed techniques are by subjecting them to a domain other than SmartHouse. In this chapter, the case authoring techniques developed are applied to data in the domain of air and marine safety investigation obtained from the Canadian Transport Safety Board (TSB 2007). Adjustments that will be required in order to apply SmartCAT-T to this domain are highlighted. A short evaluation is also carried out to judge the usefulness of the created cases in problem-solving.

9.1 The Data

The data used in this part of the evaluation are in the form of records of investigations regarding transportation *incidents*. These reports are complex documents containing observations, findings and recommendations made after each incident. Figure 9.1 shows one report excerpt. The reports contain detailed descriptions of factors that might have led to

the incident and these (factors) fall in three broad categories: *environmental*, *human* and *equipment*.

The Transportation Safety Board of Canada (TSB) investigated this occurrence for the purpose of advancing transportation safety...

Summary

The pilot of an amateur-built Schreder HP 18 glider (registration C-GSTL, serial number 18105) was prepared for a winch-assisted take-off from the grass adjacent to Runway 02 at Stanley Airport...

Other Factual Information

The Schreder HP 18 is a single-place, high-performance glider. In the HP 18, the pilot is seated in a nearly horizontal (lying-down) position...

Analysis

Because of the severity of the breakup, it is unlikely that use of the shoulder harness straps would have reduced the severity of the injuries...

Findings as to Causes and Contributing Factors

Shortly after lifting off, the aircraft entered a steep climbing attitude and a wing stall ensued...

Findings as to Risk

The shoulder harness straps were not latched prior to take-off...

Figure 9.1: Safety Report Excerpt

Each report depicts the nature of the incident and this is recorded in the *Summary*, *Analysis*, *Synopsis*, and *Other factual information* sections of the report. The rest of the report depicts either some conclusions or recommendations, in the *Findings as to causes*,

Findings as to risk, *Other findings* and *Safety actions* sections of the report. One of the ways in which CBR could contribute to this domain would be in assisting in the process of drawing conclusions as to the cause of an incident, or making recommendations, for a new incident whose nature is described as a *Summary*, *Analysis*, *Synopsis*, and *Other factual information*. Thus it is assumed that the problem part of a case comprises the nature of an incident which is found in the *summary*, *other factual information* and *analysis* and the solution part to be the *findings*, *recommendation* or *conclusion*.

The database for the domain of air and marine safety comprises more than 2 million documents. For purposes of testing if the techniques developed in this research would be useful in this domain, we required only a few documents to roughly match the number that had been used in the SmartHouse domain. 52 documents were randomly selected for the purpose. Selection of anchors requires that the implementer be able to understand some aspects of the domain. Moreover, we wanted to be able to reasonably understand the outcomes, at the various stages, of applying SmartCAT-T to this domain. Some documents contain detailed descriptions of events that are not easy to understand for non-experts. This is especially so for incidents that were caused by human factors. However, it was easy to obtain aircraft parts from vendors and to determine if the incident in a given report was caused by weather conditions by reading the report summary. Hence only those reports in which incidents were either caused by equipment or environmental factors were selected for use. These amounted to 26 in total. The SmartHouse reports though few, feature common disabilities that affect people. Thus it is easy to reuse them in solving new problems. By their nature, incident reports will not have a lot in common since the aim is to try and ensure that a given incident does not occur again through the adaptation of appropriate safety techniques. However, causes of incidents such as bad weather conditions and equipment failures do reoccur. Thus report selection was dependent on reoccurence of the general causes (weather and equipment malfunction) such that the few that were selected could potentially be reused in a CBR system.

9.2 Applying SmartCAT-T

The SmartCAT-T approach was applied to documents in this domain as elaborated in the next Subsections.

9.2.1 Report Pre-processing

The first part of the pre-processing step was the removal of non-alphanumeric and numeric characters, followed by lemmatisation. A domain-specific synonym list was built and text standardisation was carried out. This was to ensure the few data we had were cohesive and that queries would be interpreted in a meaningful way. The techniques highlighted in Chapter 3 are domain independent and were therefore applied without any adjustments.

9.2.2 Key Phrase Extraction

The key phrase extraction technique relies on identification of a few key terms that are then used as anchors to obtain other key problem descriptors. Thus like in the Smart-House domain, background domain knowledge was used to extract key phrases from the investigation reports by assuming that, "*terms which are related to equipment parts and weather conditions will be important*". Since this was an unfamiliar domain, it was not possible to use seed words to obtain anchor terms. Instead, a frequency-based technique and syntactic information of the words in the documents, was used to automate as much as possible, the process of obtaining the anchors.

Identifying terms that are related to equipment and weather conditions is not a straightforward task, even for humans. First, a human would need to know helicopter parts before they can use them to find other problem-descriptors. The SmartCAT-T tool does this by making use of a list of helicopter parts in order to identify equipment-related words that are mentioned in the reports. A few terms relating to weather such as *snow* and *rain* were easily obtained manually from the summaries of some reports. Sixteen of the most frequent nouns and verbs were then selected from the equipment and weather related terms and used as anchor terms (see Figure 9.2). It should be noted that the term *ic* shown in Figures 9.2 and 9.4 is a lemmatised version of the terms *ice* and *icing*.

The anchor-terms were used to aid key phrase extraction from the problem parts of

rig, engine, governor, power, weather, ic, snow, rain, wind, water, winter, rotor, control, blade, nosewheel, axle

Figure 9.2: Chosen Anchors

the reports, using LSI. Although only a few terms are used as anchors, SmartCAT-T is able to identify more phrases that do not have the anchor words as their substrings or even as part of the sentences in which the phrases appear. This is because LSI makes use of co-occurence patterns between terms in order to identify important terms. Figure 9.3 shows key phrases extracted from one report.

...aircraft over speed, loss of control, aural stall warn, ambient sound level, sound level reduce the effectiveness, high engine power operation, improperly place airspeed range, place airspeed range mark, position the stall warn system, render the visual warn ineffective, system warn ineffective, visual indicator, defence, aerodynamic, recovery, raise the ambient noise level, mount the visual stall warn, visual stall warn ineffective...

Figure 9.3: Example of Key Phrases in the Domain of Air and Marine Safety

Most of the key phrases shown in Figure 9.3 are reasonably important. Phrases such as *aircraft over speed*, *loss of control*, *improperly place airspeed range* and *aural stall warn(ing)* are all important features of the problem part of the corresponding incident report.

9.2.3 Creation of a Conceptual Structure and Acquiring the Cases

Each problem part and its representative terms were regarded as an object and attributes respectively from which a formal context was created. FCA was applied to this formal context in order to obtain a concept hierarchy. A portion of the hierarchy is shown in Figure 9.4.

Examination of the hierarchy shows the most common features of incidents such as *collision with terrain* (node 1), *(height) feet above ground* (node 5) and *insufficient altitude* (node 7) appearing at the top of the hierarchy. It is also noteworthy that the created relationships are reasonable: for example *struck the ground* (node 2) is a feature of *collision with terrain* (node 1), *visibility* (node 6) can be a characteristic of *(height) feet about ground* (node 5) and, *failure* of a component (node 3) can result into/from an *accident* (node 4). When SmartCAT-T was applied to the SmartHouse domain, the complexity of the graph-

structured hierarchy was reduced before it was transformed into a tree structure. This was carried out by tagging terms with their corresponding disability-terms. It was not possible to do the same for this domain because there was a lack of knowledge if the complexity could be reduced without losing knowledge and how it could be done appropriately.

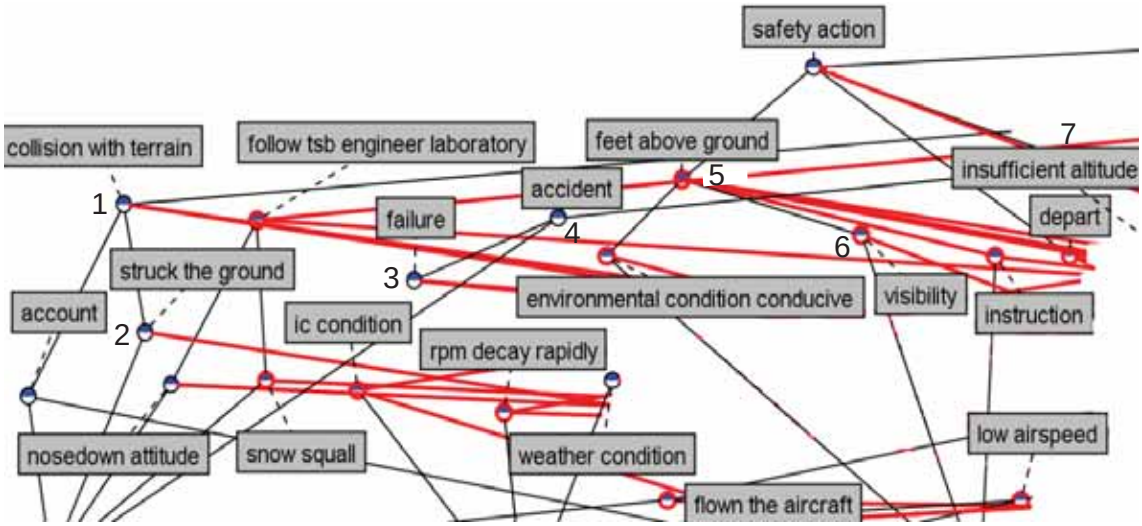


Figure 9.4: A Portion of the Concept Hierarchy for the Air and Marine Safety Domain

Consequently, the graph-tree transforming algorithm described in Chapter 7 was applied to the concept hierarchy to obtain a tree-structured conceptual model without first reducing the complexity of the conceptual model. Each textual document was then mapped on to the structured hierarchy by identifying concepts whose whole intent pertain to the document. The most specific concept node was tagged with the solution for that report and the authored cases stored in the SMARTER CBR system. It should be noted that the SmartHouse cases and cases in the domain of air and marine safety do not reside in the same database and so the user has to specify the domain from which cases should be retrieved in response to a query.

9.2.4 Problem-Solving

Case-based reasoning would assist in this domain by retrieving reports with the correct findings or conclusions given a set of features describing a problem i.e., *analysis* or *other*

factual information. A leave-one-out test was employed to determine the usefulness of the created cases in problem-solving. 5 cases were used in turn as test probes to the SMARTER CBR system. Thus for each test probe, the SMARTER CBR tool was expected to retrieve solutions that are similar to the ones corresponding to the test probes. The correctness of the suggested solutions was judged by a domain expert Brindley (Brindley & Bradley 2003). This is because the solutions suggested by SMARTER are texts that cannot be compared to other texts by a non-expert. The retrieved solutions will be referred to as *Smarter's solutions* and the solutions corresponding to the test probes as *probe solutions*.

The top 3 retrieved solutions for each of the 5 test probes i.e., 15 of SMARTER's solutions, were presented to the expert together with the 5 probe solutions. SMARTER's solutions were randomly ordered and the expert was asked to match them to each of the 5 probe solutions by deciding which, if any, of SMARTER's solutions were similar to the 5 probe solutions. He was not informed that there were 3 SMARTER solutions for each of the 5 probe solutions. This was a classification problem where the expert was required to classify each of the 15 SMARTER solutions under the 5 solutions for the problems being solved. The idea here is that, if SMARTER was 100% correct for all the 5 experiments, then the expert would match 3 of SMARTER's solutions to each of the 5 probe solutions, in the same way that SMARTER had. Table 9.1 shows results of this experiment. A *relevant* SMARTER solution is one that the expert selects for the test probe.

The first column of Table 9.1 represents a particular test probe. The second column shows the number of SMARTER solutions that the expert deemed to be relevant to solving the test probe. The third column shows the number of SMARTER solutions that SMARTER retrieved in response to the test probe and which were also judged by the expert to be relevant to solving that test probe. *Precision*, *Recall* and *F-Scores* corresponding to each test probe are shown in the subsequent columns; they are calculated using Equations 9.1, 9.2 and 9.3 (repeated here for convenience). *Ret* and *Rel* are the sets of retrieved and relevant solutions respectively. The number of retrieved cases for each query is 3.

$$Precision = \frac{|Ret \cap Rel|}{|Ret|} = \frac{|Ret \cap Rel|}{3} \tag{9.1}$$

Query	Rel	$Ret \cap Rel$	Precision	Recall	F-Score
1	2	1	0.33	0.50	0.40
2	3	2	0.67	0.67	0.67
3	3	2	0.67	0.67	0.67
4	4	3	1.00	0.75	0.86
5	3	2	0.67	0.67	0.67
Average	3	2	0.67	0.65	0.65

Table 9.1: Results for 5 Leave-one-out Test Runs for 3NN

$$Recall = \frac{|Ret \cap Rel|}{|Rel|} \tag{9.2}$$

$$F - Score = \frac{2PR}{P + R} \tag{9.3}$$

As Table 9.1 shows, the SMARTER CBR tool performs reasonably well for this domain. On average, 2 solutions were relevant out of the 3 retrieved SMARTER solutions. Good (better than 50%) *Precision* and *Recall* were obtained for 4 out of the 5 test probes. An average Precision of 0.67, average Recall of 0.65 and an average F-Score of 0.65 were obtained for the 5 test probes. This good performance can be attributed to the fact that the authored cases contain knowledge that is useful for problem solving.

Examination of probe solutions where there was disagreement between SMARTER and expert classification show a deficit in domain knowledge. Consider the SMARTER and probe solutions for query 1 shown in Figure 9.5. The term *rigging* is important and clearly, the cause of problems in both the SMARTER and probe cases was related to *rigging*. In the probe case, the governor was *abnormally rigged* and in the SMARTER case, the *rigging*

tolerance was inadequate. However, *rigging* in both instances refers to *setting*[1]. Thus according to the expert, rigging of the *governor* is unrelated to rigging of the *bellcrank to cam assembly*. This is because one is referring to the *engine* and the other to the *landing gear* which makes the term *rigging* unimportant in this context. If the SMARTER tool had knowledge of the fact that *rigging* was important, but that it is what was rigged that determined if the two solutions were similar, it would have given the SMARTER solution a lower similarity score than it did in this instance. The SMARTER solutions that were retrieved for probes 2, 3 and 5 but which the expert deemed to be irrelevant for solving those probes can be attributed to the same problem. SMARTER does not put the importance of query terms into consideration when calculating the similarity score. This was partly mitigated in the SmartHouse domain because the person's disability was taken into consideration when calculating the similarity scores. Hence the similarity of terms was automatically reduced if they were associated with different disability-terms. At the time of exploring the domain of air and marine safety, there was a lack of knowledge in deciding when terms were likely to harm retrieval effectiveness. This is also the reason behind SMARTER's mistake with query 3 where *severe icing* on the aircraft was erroneously linked to a *slippery surface* on the runway (in the probe case) that was due to *icing conditions*.

Overall, SMARTER performs reasonably well in this unfamiliar domain. Most of the false positives can be attributed to SMARTER's inability to distinguish terms that appear similar but occur in different contexts in determining similarity scores as was done in the SmartHouse domain when terms' corresponding disability-terms were considered. The other contributing factor could be the fact that SMARTER does not utilise importance scores of key phrases in the query and the activated concepts in the cases to determine similarity scores during problem-solving. However, the good performance still shows that the techniques developed in this research are generic enough to be useful in other domains.

[1]Setting in this instance means *"to put in the proper position"*.

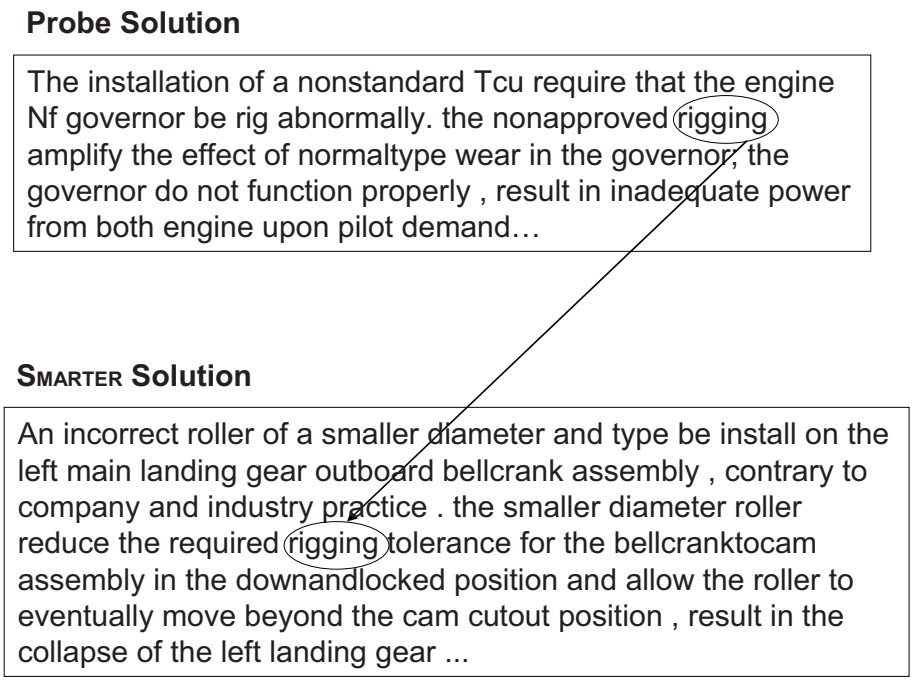


Figure 9.5: Example of Misclassified SMARTER and Probe Solution Pairs

9.3 Summary

This chapter has investigated the application of the techniques developed in the project, on the domain of air and marine safety investigation. The case authoring tool SmartCAT-T was applied to incident reports in this domain in the same way as was done for the SmartHouse reports except that the anchor terms were obtained using a frequency based approach and the complexity of the graph-structured hierarchy could not be reduced before transformation into a tree structure because of a lack of knowledge to carry out complexity reduction in an appropriate manner. An evaluation was carried out to judge the usefulness of the created cases in problem solving. Good results were obtained which shows that the cases contain enough knowledge to assist in problem-solving. Indeed, the average F-Score was quite good and the number of false positives was low considering this was an unfamiliar domain where relatively little background knowledge was assumed. This shows that the developed techniques are generic enough to be utilised in domains other than the SmartHouse. The next chapter highlights the important contributions made in this project and gives some directions for future research before concluding the thesis.

Chapter 10

Conclusions and Future Work

This thesis has investigated some of the techniques for mining knowledge from semi-structured text. The reliance of statistical approaches on large volumes of data to extract features from text, and the fact that the approaches typically result in knowledge-poor representations, made them unattractive for the SmartHouse domain. Consequently, they were employed to a very limited extent. Approaches based on Information Extraction systems were also ruled out because of the level of human intervention required to tune the information extractors. Hierarchically structured cases result in more effective retrieval than flat-structured cases because they allow for case comparison at various levels of problem abstraction. This project strikes a good balance between the need to automate the system and the desire for a knowledge-rich case representation that comprises various levels of knowledge abstraction. This is achieved by using background domain knowledge to mitigate some of the effects of data sparseness in order to automatically extract key phrases. The key phrases are used to construct a conceptual model for the domain which in turn, is used to create knowledge-rich structured cases from the text and for query interpretation during problem-solving. A language model is also developed to enable the case authoring process and retrieval mechanism to meaningfully interpret text by allowing the recognition of word synonyms in text. Thus this project mimics the way humans interpret text by making use of a conceptual model and by considering the different meanings a given word could have in the given text. The result is a case authoring tool that identifies important phrases in text and uses them to create hierarchically structured

cases, and a case-based problem-solving tool for the corresponding domain.

Eliciting case knowledge from textual sources depends on the nature of the text and the targeted case representation. Consequently, the knowledge elicitation techniques employed in this project were dependent on the fact that the SmartHouse reports are semi-structured with the bulk of the knowledge embedded in sub-problem descriptions but that the target knowledge was not known a priori. This chapter highlights some of the important contributions made in the research work, discusses possible extensions and concludes the thesis. In order to analyse the important contributions made, it is helpful to recall the project objectives which were as follows:

- to design a case structure onto which unstructured text reports will be mapped;
- to design and implement a semi-automated tool to map unstructured text to the case structure obtained above;
- to develop a concept hierarchy to enable structuring of cases with different levels of specificity; and
- to design and implement a case-based tool to assist in the design of SmartHouse solutions.

The contributions will mostly be analysed in the context of these objectives.

10.1 Major Contributions

This research had several outcomes the most important of which are listed below:

1. **A conceptual model in which concepts are interpretable by a human.** A hierarchically structured conceptual model has been developed using the textual problem-solving experiences in the SmartHouse domain. The tool SmartCAT-T makes use of background knowledge to determine a set of anchors that help to highlight key phrases in the text. The key phrases are then used to construct a hierarchically structured conceptual model. The structure of the conceptual model allows the creation of cases with different levels of specificity, thus achieving objective 3.

Automatically created conceptual models could typically benefit from interaction with a human expert who would refine the relations and thus supplement existing knowledge with more background knowledge. For this the knowledge determining the structure of the case base has to be comprehensible and should comprise informative descriptions of each underlying problem for the expert. The concepts in the developed conceptual model are interpretable by a human. The intents consist of terms that an expert is likely to have chosen as key phrases. Moreover, the intents mostly comprise phrases as opposed to words. Therefore it is easy for an expert to see the terms in context and amend the conceptual model as necessary. The conceptual model has been developed without the need for hand-tagged data.

2. **Automatic acquisition of cases.** SmartCAT-T makes use of the conceptual model to automatically identify in a textual document, those concepts that are important in the domain. The model has been created from texts, enabling mapping from text to structure. Consequently, the textual document is mapped onto the hierarchical structure of the conceptual model and a correspondingly structured case is created. Thus objective 1 was realised through the development of the conceptual model as it is the structure onto which the textual documents are mapped in order to facilitate reasoning that goes beyond simple key word matching. The queries are mapped onto the same model to enable more effective interpretation during problem solving. Evaluation showed the ability to map queries to a structure and thus achieve effective retrieval, even when the queries comprise unfamiliar phrases.

   The quality of the case content has been evaluated against original documents and the case knowledge was found to be as adequate and sometimes better focused for problem-solving, than whole documents. Better retrieval effectiveness was also obtained using the case structure than with a flat structure using a high performance IR tool, because the hierarchical structure allows case comparison at different levels of problem specificity. This is a useful feature for domains where the ability to match problem descriptors at various levels of abstraction is crucial, but more importantly, where case and adaptation knowledge are scarce.

   The second objective was realised through the automatic mapping of semi-structured

text to a structured representation and the second part of objective 3 through the fact that the cases are structured with different levels of specificity. Automatic acquisition of case knowledge made SMARTER the resulting case based reasoning system, easy to develop and makes it easy to maintain from textual records. Formal Concept Analysis and Latent Semantic Indexing have been employed in CBR before, and thus their individual use in CBR would not be novel. However, the use of Latent Semantic Indexing to generate features for use in Formal Concept Analysis in order to exploit the resulting conceptual model in CBR has not been explored by others.

3. **A case based reasoning tool.** The project has also developed a case based reasoning tool that matches SmartHouse solutions to the problems or difficulties faced by the elderly and people with disabilities. As seen from the evaluation, the SMARTER CBR tool recommends reasonable solutions for new problems. Thus objective 4 was realised.

4. **Unsupervised approach to Word Sense Disambiguation.** Another contribution of this project was the development of an unsupervised approach to Word Sense Disambiguation. WordNet is used to assign each polysemous word with a sense that is applicable in the domain. The Word Sense Disambiguation algorithm makes use of syntactic information of the sentence in which a target word appears and information from Google, to assign the correct sense to the target word, in the context in which it appears. Google is used as a large corpus that provides evidence that the disambiguated phrase is plausible in the English language. The use of rule-based systems for WSD yields systems that do not scale up because of the cost of manually creating the rules on which the systems rely to make inferences. Statistical techniques scale up more easily but the corpus may still not have sufficient information to support disambiguation and the use of frequency information to support disambiguation is not suitable for sparse domains such as SmartHouse. The word sense disambiguation technique developed in this research is automated but it neither relies on statistical information of the word in the document collection nor on hand-tagged data.

5. **Generic techniques that assist case authoring from text.** The techniques developed in this research were applied to the domain of air and marine safety in

order to explore the extent to which they were generic. It was found that minimal adjustments were required to apply the techniques to the new domain. Results from an evaluation also found the cases created from documents in the domain to be useful for problem-solving. This shows potential for the case authoring techniques to be useful in other domains where knowledge is embedded in text.

The next section describes maintenance of the conceptual model, a feature of the SmartCAT-T tool that had not been mentioned in the original objectives because it is not directly concerned with knowledge elicitation from text.

10.2 Maintaining the Conceptual Model

The conceptual model has been made editable, and so it is possible for the expert to remove attributes that she deems to be unimportant in describing the problem. It is also possible to add or change words in an attribute. However, these changes have to be limited for a number of reasons.

Consider the sub-problem illustrated in Figure 10.1. The terms in bold make up the intent of the concept that represents this sub-problem. The domain expert could decide that since *Telephone Operation* is part of the intent, there is no need to have *contact by telephone* as well. This term may then be eliminated from the conceptual model and invariably from the cases. On the other hand, suppose that the expert looks at the original text and decides that the term *unable to communicate* is not as good as *unable to communicate effectively by telephone*. The expert can amend the intent and the change will be reflected in the appropriate cases when the authoring module is run again. The task of determining the appropriateness of a phrase in an intent is an easy one if the original text is available to the expert, with the intent highlighted in some way.

Removal of a term in an abstract node's intent will change the relationships between the cases to which the node is attached. However, it is unlikely that a shared phrase will be considered unimportant and therefore qualify for removal. The decision has been left to the expert to decide whether a given relationship is important and therefore worth keeping. Shared attributes will most likely need to be changed instead of deleted; a 'bad' term could be replaced with a more informative one. However, the expert would have to

Telephone Operation. *Mrs. B had a sister who lives in America and on a regular basis made* ***contact by telephone*** *to check on Mrs. B's health and wellbeing. Mrs. B is* ***unable to communicate*** *effectively by telephone because of her* ***hearing difficulties*** *and in most instances, both Mrs. B and her sister relied upon Mr. B to* ***relay messages*** *between the two. This had become a cause of great* ***frustration and upset*** *for both Mrs. B and her sister. Mr. B also became upset and frustrated by the* ***lack of communication*** *between his wife and her sister, especially as he was having to act as go between.*

Figure 10.1: Deleting Vs. Amending Representative Terms

keep in mind that these changes will propagate to all the nodes that represent sub-concepts of the shared node.

All the changes made to the conceptual model are also reflected in the model's context table. This is because each new authored case will introduce a new set of attribute-terms and objects which need to be attached at appropriate positions in the hierarchy. However, it is impractical to expect a human to do this manually. Therefore, the new case knowledge is added to the context table and a new conceptual model generated. Thus, if the context table does not reflect the changes that were earlier made by the expert's editing of the conceptual model, the new model that is generated after adding a case will only reflect the changes due to the new added case and all the changes made to the hierarchy over time will be lost.

10.3 Future Work

This section highlights some of the limitations or shortcomings of the approaches used in this research and consequently identifies some directions for future research and improvement of the SmartCAT-T tool.

10.3.1 Pruning of the Conceptual Model

A technique for systematically pruning the conceptual model might be useful, especially as more knowledge is acquired. One reason for this is that even domain experts could disagree as to which attributes to delete, change, etc. Moreover, a good subset of the original concepts might be more useful than using all the concepts. Thus it is worth exploring pruning techniques that would be advantageous to the domain.

10.3.2 Word Sense Disambiguation

The approach presented in Chapter 3 could be extended as follows. Currently, each term is disambiguated independent of the other terms in the text. This means that for specialised domains, there may be limited evidence of the plausibility of the disambiguated phrase as obtained by Google. However, if the relationships between terms was taken into account, a term that has been disambiguated with high evidence from Google might "help" in disambiguating another term to which it is related. That way, evidence from Google regarding the plausibility of a phrase in which a given term is embedded, could be used to determine the plausibility of another phrase in which a related term is embedded and thus provide enough evidence for disambiguation of the related term. This approach is likely to result in the disambiguation of more terms since more evidence will be available to support disambiguation.

10.3.3 Retrieval Effectiveness

Currently during retrieval, all the query features are used to retrieve matching cases that are then ranked using the structure which exploits semantics to complement overlapping syntactic information between the query and the cases. However, some of the query features may be more important than others. Thus, it would be useful to identify this knowledge and incorporate it into the retrieval mechanism. This knowledge could be as simple as the importance weights obtained by LSI during key phrase extraction, or other feature ranking mechanisms such as those employed by Wiratunga et al. (2005) or in SOPHIA (Patterson et al. 2005) could be explored and utilised. These techniques could also make use of solution description texts to assign importance weights to key phrases in the problem text. These weights could then be used in ranking retrieved cases and so potentially improve on the quality of the similarity metric. As an alternative to automatically assigning importance weights to query features, it might be worth exploring whether a graphical representation of the query that allows the user to choose which parts of the structure to use for retrieval would improve on retrieval effectiveness. A mixed-initiative dialog similar to that employed in Conversational CBR could be explored in refining queries in order to improve on retrieval effectiveness.

Retrieval effectiveness has been evaluated using Precision, Recall and the F-Score. As further research, other means of determining retrieval effectiveness could be explored. One such measure is the Average Distance Measure (Mea & Mizzaro 2004) where the concept of *continuous relevance*[1] of the retrieved cases could be explored.

10.3.4 Editing the Conceptual Model

If all the terms in the intent are deleted then the corresponding sub-problem will have no description! Thus, deleting of terms could be limited in such a way that makes it impossible to delete all the terms in an intent. For instance, a term in an intent could only be removed if it does not contribute to the description of the problem and cannot be changed to do so. Currently, each intent is mostly composed of long phrases and so it is easy for the expert to grasp the context. As an extension to the tool, each intent's corresponding text could be provided at the concept node so as to further encourage the expert to view the terms in context and thus make informed decisions when editing a concept node's intent.

10.3.5 Exploring other domains

It is worth exploring the application of the developed techniques in domains other than SmartHouse. Sparse and dense domains could be explored. For sparse domains, other means of incorporating domain knowledge could be explored to aid the key phrase extraction task. For example in domains such as Medical where ontologies such as MESH[2] are available, the ontology can be exploited in providing domain knowledge that can in turn, be used to extract key phrases. SmartCAT-T makes use of background domain knowledge to obtain the seeds for anchor term identification. The anchor terms are then used to obtain other key phrases using term-term relationships obtained from LSI. An ontology could replace the use of LSI in SmartCAT-T. Thus an anchor term could be used to obtain other important terms by making use of relationships between the anchor term and other terms in the ontology. In dense domains, the anchor terms could be obtained using statistical means as was done in the domain of air and marine safety. In dense domains,

[1] In continuous relevance, the suitability of a retrieved solution is continuously assessed.
[2] http://www.nlm.nih.gov/mesh/meshhome.html

a purely statistical means of obtaining key phrases could be explored. For example, LSI could be exploited without the anchor terms to guide the key phrase extraction process. It would be interesting to observe the level of knowledge extracted from the texts without the use of additional domain knowledge. The reports in the domain of air and marine safety were long and verbose but the developed techniques still worked satisfactorily. Subjecting the techniques to a domain with longer documents or a much bigger corpus could help to determine their scalability limits.

10.3.6 Building the Conceptual Model

Other techniques of obtaining the conceptual model could be explored. Formal Concept Analysis is very good at arranging concepts according to their IS-A relationships and will thus be hard to compete with. However, other techniques such as that presented by Sanderson & Croft (1999) could be explored. Indeed, documents from the Web could be used to ascertain the validity of the obtained relationships as is done in the KnowItAll (Etzioni et al. 2005) system.

Chuang & Chien (2004) showed the usefulness of the subsumption hierarchies from Sanderson & Croft (1999) for interactive query expansion. These techniques typically do not put the meaning of the words or word order into consideration and as a result, relations that show a term as being an aspect of another term, e.g., *disease - outbreak* are extracted alongside hyponym/hypernym relations like *banana - fruit*. Thus although the approaches can be fully automated, many of the extracted relations are not useful in most applications.

10.3.7 Negation

Negation has not been handled in this research. It is unusual for a person's complaints to be listed and thereafter negated. Thus it is not common to encounter negation in the problem parts of the reports and none was encountered in this research. However, it is possible for a solution to be considered and then discarded due to the circumstances at the time. Thus it is possible to encounter negation in the solution part of the SmartHouse reports. Fortunately in this project, the inability to handle negation did not have serious repercussions because there were only two solution descriptions that had negation. How-

ever, as future work, it would be advantageous for the SmartCAT-T tool and SMARTER CBR system to have a means of recognising negating statements or terms in order to assign the appropriate similarity weights during retrieval. Terms such as "unfortunately or "however could be good indicators of negation and should thus be exploited.

10.4 Lessons Learned

A number of lessons were learned from the research questions, most notable of which were:

- Fully-automated systems require large amounts of data and still result in knowledge-poor representations. Domain knowledge needs to be incorporated to bridge the gaps in knowledge obtained from the data in order to obtain knowledge-rich cases. However, the available means of incorporating this knowledge require some degree of manual intervention. Thus, it is not possible at present, to create knowledge-rich cases from sparse sources without a certain level of manual intervention.

- It is possible to semi-automate the knowledge elicitation task whilst ensuring that the amount of required manual intervention does not inhibit system scalability. This was achieved with the use of seed words and domain-dependent assumptions to aid the key phrase extraction task.

- The resulting identified domain concepts can allow for knowledge refinement and system learning by ensuring that they are understandable by a human and by creating a provision for all new knowledge from new cases and knowledge refinement to be stored in a persistent manner.

10.5 Summary

The case authoring tool SmartCAT-T exploits co-occurrence patterns of terms in the different sub-problems to learn association rules in order to enrich the case representation just like systems such as SOPHIA (Patterson et al. 2005). However, SmartCAT-T goes further to identify domain concepts in the enriched representations and to learn their relationships in order to obtain a conceptual structure. Techniques like Latent Semantic Indexing

produce features with an underlying structure; the features are linear combinations of terms in the document collection. However, the interpretability of the representations and underlying concepts remains uncertain. Thus, expert initiated refinement of knowledge is difficult for these features. SmartCAT-T exploits the linear combinations of terms provided by LSI (in the form of topics) to identify important features that are used to create an interpretable structure using Formal Concept Analysis.

The approaches developed are generally applicable for knowledge modeling in semi-structured textual sources where domain knowledge is embedded in the free-form text of the section content. In domains like medicine and SmartHouse where concepts are shared between different entities, e.g. symptoms among different diseases, there is a requirement to have a conceptual structure that is interpretable in addition to a knowledge-rich case representation, while ensuring that the system is a cost-effective solution to problem-solving. SmartCAT-T fulfills these requirements by harnessing a number of techniques.

Common among the systems that employ information extraction tools to obtain structured cases is the significant amount of manual intervention required for tuning the information extractors. SmartCAT-T does not employ an information extraction system, but makes use of information extraction techniques and background knowledge to extract key phrases which are used to learn a conceptual structure in an unsupervised process. All this is carried out without the need for tagged data.

The use of manual intervention to create case representations from text ensures that correct features are selected from the text and semantics and domain knowledge are incorporated in the representations. However, developing these systems requires engaging an expert to provide the necessary knowledge for instance by tagging documents and the developed system can be unscalable because it is difficult to incorporate new knowledge. On the extreme end of the spectrum are techniques that are completely automated and use simple statistical analysis to obtain case representations from text. Unfortunately, the resulting representations are typically knowledge-poor, lack an underlying structure or if the structure such as that obtained using Latent Semantic Indexing exists, it is not interpretable by a human. SmartCAT-T strikes a healthy balance between these possibilities by making use of some background domain knowledge to inform the automatic key phrase extraction process and consequent creation of the conceptual model and the cases,

in an automated manner. Evaluation using a different domain showed that the techniques are not wholly tailored to the SmartHouse domain but that they can be applied to other domains with ease and the obtained cases are useful for problem-solving.

SmartCAT-T overcomes the short-comings in many of the systems mentioned in the literature by combining their complementary strengths. The result is an automatically created knowledge-rich, hierarchically structured, case representation from text. The case representation and its concepts are interpretable, allowing for expert refinement of knowledge.

Bibliography

Aamodt, A. (1994). Explanation-driven case-based reasoning, *Topics in case-based Reasoning* pp. 274–288.

Aamodt, A. & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches, *Artificial Intelligence Communications* **7**(1): 39–59.

Aleven, V. (2003). Using background knowledge in case-based legal reasoning: a computational model and an intelligent learning environment, *Artificial Intelligence* **150**(1-2): 183–237.

Aleven, V. & Ashley, K. D. (1997). Teaching case-based argumentation through a model and examples: Empirical evaluation of an intelligent learning environment, *Proceedings of Artificial Intelligence in Education, AI-ED 97 World Conference*, IOS Press, pp. 87–94.

Ashley, K. D. & Aleven, V. (1996). Evaluating a case argument tutorial program with a first year legal writing section, *Proceedings of the 4th International Conference on Substantive Technology in the Law School and Law Practice*, pp. 110–116.

Ashley, K. D. & Aleven, V. (1997). Reasoning symbolically about partially matched cases., *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 335–341.

Asiimwe, S., Craw, S. & Taylor, B. (2006). Discovering a concept hierarchy from SmartHouse reports, *Workshop Proceedings of the 8th European Conference on Case-Based reasoning*.

Asiimwe, S., Craw, S., Taylor, B. & Wiratunga, N. (2007). Case authoring: from textual reports to knowledge-rich cases, *Proceedings of the 7th International Conference on Case-Based Reasoning*, Springer, Berlin, Heidelberg, pp. 179–193.

Berry, M. W., Dumais, S. T. & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval, *SIAM Review* **37**(4): 573–595.

Brindley, G. & Bradley, R. (2003). Progress in the development of a versatile pilot model for the evaluation of rotorcraft performance, control strategy and pilot workload, *The Aeronautical Journal* **107**(1078): 731–738.

Briscoe, T., Carroll, J. & Watson, R. (2006). The second release of the rasp system, *Proceedings of the COLING/ACL on Interactive presentation sessions*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 77–80.

Brüninghaus, S. & Ashley, K. D. (1997). Using machine learning for assigning indices to textual cases, *Proceedings of the 2nd International Conference on Case-Based Reasoning*, Springer, pp. 303–314.

Brüninghaus, S. & Ashley, K. D. (1999). Bootstrapping case base development with annotated case summaries, *in* K.-D. Althoff, R. Bergmann & L. K. Branting (eds), *Proceedings of the 3rd International Conference on Case-Based Reasoning*, LNAI 1650, Springer, pp. 59–73.

Brüninghaus, S. & Ashley, K. D. (2001). The role of information extraction for textual CBR, *in* D. Aha, I. Watson & Q. Yang (eds), *Proceedings of the 4th International Conference on Case-Based Reasoning*, LNAI 2080, Springer, pp. 74–89.

Bunke, H. (2000). Recent developments in graph matching, *Pattern Recognition* **2**: 117–124.

Cañas, A., Leake, D. B. & Maguitman, A. (2001). Combining concept mapping with cbr: Towards experience-based support for knowledge modeling, *Proceedings of the 14th International Florida Artificial Intelligence Research Society Conference*, AAAI Press.

Caraballo, S. A. (1999). Automatic construction of a hypernym-labeled noun hierarchy from text, *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 120–126.

Chakraborti, S., Wiratunga, N., Lothian, R. & Watt, S. (2007). Acquiring word similarities with higher order association mining, *7th International Conference on Case-Based Reasoning*, LNCS 4626, Springer, pp. 61–76.

Charniak, E. (1994). *Statistical Language Learning*, MIT Press, Cambridge, MA, USA.

Christiansen, C. & Baum, C. M. (1997). *Occupational therapy: Enabling function and well-being*, SLACK Inc, Thorofare, NJ.

Chuang, S.-L. & Chien, L.-F. (2004). A practical web-based approach to generating topic hierarchy for text segments, *Proceedings of the 13th ACM international conference on Information and knowledge management*, ACM, New York, NY, USA, pp. 127–136.

Cimiano, P., Hotho, A. & Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis, *Journal of Artificial Intelligence Research* **24**: 305–339.

Cimiano, P., Ladwig, G. & Staab, S. (2005). Gimme' the context: context-driven automatic semantic annotation with c-pankow, *Proceedings of the 14th international conference on World Wide Web*, ACM, New York, NY, USA, pp. 332–341.

Cohen, W. W. & Singer, Y. (1996). Context-sensitive learning methods for text categorization, *SIGIR*, pp. 307–315.

Cox, D. R. (2006). *Principles of Statistical Inference*, Cambridge University Press, United Kingdom.

Craw, S. & Rowe, R. (2002). Case-based reasoning for tablet formulation: A novel technology to re-use past formulations, *Drug Delivery Companies Report*, Pharmaventures, pp. 14–17.

Cunningham, C., Weber, R., Proctor, J. M., Fowler, C. & Murphy, M. (2004). Investigating graphs in textual case-based reasoning., *in* P. Funk & P. A. Gonzlez-Calero (eds), *Advances in Case-Based Reasoning*, LNCS 3155, Springer, pp. 573–586.

Curran, J. R. & Moens, M. (2002). Improvements in automatic thesaurus extraction, *Proceedings of the Association for Computational Linguistics-02 workshop on Unsupervised lexical acquisition*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 59–66.

Daniels, J. J. & Rissland, E. L. (1997). What you saw is what you want: Using cases to seed information retrieval, *Proceedings of the 2nd International Conference on Case-Based Reasoning*, pp. 325–336.

Davis, R., Bruchanan, B. G. & Shortliffe, E. (1985). Production rules as a representation for a knowledge-based consultation program, *in* R. J. Brachman & H. J. Levesque (eds), *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA, pp. 371–387.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. (1990). Indexing by latent semantic analysis, *Journal of the American Society for Information Science* **41**(6): 391–407.

Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Mccurley, K. S., Rajagopalan, S. & Tomkins, A. (2003). A case for automated large-scale semantic annotation, *Web Semantics: Science, Services and Agents on the World Wide Web* **1**(1): 115–132.

Eibe, F., Paynter, G. W., Witten, I. H., Gutwin, C. & Nevill-Manning, C. G. (1999). Domain-specific keyphrase extraction, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 668–673.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S. & Yates, A. (2005). Unsupervised named-entity extraction from the web: an experimental study, *Artificial Intelligence* **165**(1): 91–134.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, The MIT Press.

Gabrilovich, E. & Markovitch, S. (2007). Computing semantic relatedness using

Wikipedia-based explicit semantic analysis, *Proceedings of the 20th International Joint Conference for Artificial Intelligence*, Hyderabad, India.

Gebhardt, F., Voß, A., Gräther, W. & Schmidt-Belz, B. (1997). *Reasoning with complex cases*, Kluwer Academic Publishers, Norwell, MA, USA.

Glover, E., Pennock, D. M., Lawrence, S. & Krovetz, R. (2002). Inferring hierarchical descriptions, *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, ACM, New York, NY, USA, pp. 507–514.

Golub, G. H. & Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix, *SIAM Journal on Numerical Analysis* **2**: 205–224.

Golub, G. & Loan, C. V. (1996). *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, USA.

Golub, G. & Van Loan, C. (1989). *Matrix Computations*, second edn, Baltimore, MD.

Gu, M. & Aamodt, A. (1994). A knowledge-intensive method for conversational CBR, *Topics in Case-based Reasoning* pp. 274–288.

H., Maynard, C. & Tablan (2002). GATE, a framework and graphical environment for robust NLP tools and applications, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 77–80.

Hammond, K., Burke, R., Martin, C. & Lytinen, S. (1995). FAQ finder: a case-based approach to knowledge navigation, *Working Notes of the AAAI Spring Symposium on Information Gathering in Heterogeneous Environments*, AAAI, pp. 69–73.

Hayes-Roth, F. (1985). Rule-based systems, *Communication of the ACM* **28**(9): 921–932.

Hearst, M. A. (1994). Automatic acquisition of hyponyms from large text corpora, *Proceedings of the 14th International Conference of the Association for Computational Linguistics*, pp. 539–545.

Hearst, M. A. (1998). Automated discovery of wordnet relations, *C. Fellbaum, WordNet: An Electronic Lexical Database*, MIT Press, pp. 131–153.

Hirst, G. (1987). *Semantic interpretation and the resolution of ambiguity*, Cambridge University Press, New York, NY, USA.

Klapaftis, I. P. & Manandhar, S. (2005). Google & WordNet based word sense disambiguation, *Proceedings of the 1st workshop on learning and extending ontologies by using machine learning methods, International Conference on Machine Learning, Bonn, Germany* .

Kokiopoulou, E. & Saad, Y. (2004). Polynomial filtering in latent semantic indexing for information retrieval, *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, New York, NY, USA, pp. 104–111.

Kolodner, J. (1993a). *Case-Based Reasoning*, Morgan Kaufmann, San Francisco, CA.

Kolodner, J. L. (1993b). *Case-Based Reasoning*, Morgan Kaufmann. page 8.

Kontostathis, A. & Pottenger, W. (2002). Detecting patterns in the lsi term-term matrix.

Lamontagne, L., Bentebibel, R., Miry, E. & Despres, S. (2006). Finding lexical relations for the reuse of investigation reports, *Proceedings of the 3rd Textual Case-Based Reasoning Workshop*, Fethiye, Turkey.

Lamontagne, L., Langlais, P. & Lapalme, G. (2003). Using statistical word associations for the retrieval of strongly-textual cases, *FLAIRS Conference*, pp. 124–128.

Leake, D. B. & Wilson, D. C. (1999). Combining CBR with interactive knowledge acquisition, manipulation, and reuse, *Proceedings of the 3rd International Conference on Case-Based Reasoning and Development*, LNCS 1650, Springer, London, UK, pp. 203–217.

Lehman, J. F. (1994). Toward the essential nature of statistical knowledge in sense resolution, *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 734–741.

Lenz, M. & Glintschert, A. (1999). On texts, cases, and concepts, *Proceedings of the 5th Biannual German Conference on Knowledge-Based Systems*, Springer, pp. 148–156.

Lenz, M., Hübner, A. & Kunze, M. (1998a). Question answering with textual CBR, *Proceedings of the 3rd International Conference on Flexible Query Answering Systems*, Springer, pp. 236–247.

Lenz, M., Hübner, A. & Kunze, M. (1998b). Textual CBR, *CBR Technology, From Foundations to Applications*, Springer, pp. 115–138.

Li, X., Szpakowicz, S. & Matwin, S. (1995). A WordNet-based algorithm for word sense disambiguation, *Proceedings of the 14th International Joint Conferences on Artificial Intelligence*, pp. 1368–1374.

Maddi, G. R., Velvadapu, C. S., Srivastava, S. & de Lamadrid, J. G. (2001). Ontology extraction from text documents by singular value decomposition, *2001 CADIP Research Symposium*.

Maedche, A. & Staab, S. (2000). Discovering conceptual relations from text, *Proceedings of the 14th European Conference on Artificial Intelligence*, IOS Press, Amsterdam.

Marshall, I. (1987). Tag selection using probabilistic methods, *in* R. Garside, G. Leech & G. Sampson (eds), *The Computational Analysis of English*, Longman, London, pp. 42–56.

Massie, S., Wiratunga, N., Donati, A. & Vicari (2007). From anomaly reports to cases, *in* R. Weber & M. M. Richter (eds), *Proceedings of the 7th International Conference on Case-Based Reasoning*, LNCS 4626, Springer, Belfast, Northern Ireland, pp. 359–373.

Matsuo, Y. & Ishizuka, M. (2004). Keyword extraction from a single document using word co-occurrence statistical information, *International Journal on Artificial Intelligence Tools* **13**(1): 157–169.

McCartney, R. & Sanders, K. (1990). The case for cases: a call for purity in case-based reasoning.

McLaren, B. M. & Ashley, K. D. (1999). Case representation, acquisition, and retrieval in sirocco, *Proceedings of the 3rd International Conference on Case-Based Reasoning and Development*, Springer, pp. 248–262.

Mea, V. D. & Mizzaro, S. (2004). Measuring retrieval effectiveness: A new proposal and a first experimental validation, *Journal of the American Society for Information Science and Technology* **55**(6): 530–543.

Miller, G. A. (1995). WordNet: a lexical database for English, *Communications of the ACM* **38**(11): 39–41.

Minor, M. (1999). Managing test specifications with case-based reasoning - application paper, *in* E. Melis (ed.), *7th German Workshop on Case-Based Reasoning*.

Nanas, N., Uren, V. & Roeck, A. D. (2003). Building and applying a concept hierarchy representation of a user profile, *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ACM, New York, USA, pp. 198–204.

Norreault, T., Koll, M. & McGill, M. J. (1977). Automatic ranked output from boolean searches in SIRE, *American Society for Information Science* **28**(11): 333–339.

Patterson, D., Anand, S. S., Dubitzky, D. & Hughes, J. (2000). A knowledge light approach to similarity maintenance for improving case-based competence, *Workshop on Flexible Strategies for Maintaining Knowledge Containers 14th European Conference on Artificial Intelligence*, pp. 65–77.

Patterson, D., Rooney, N., Dobrynin, V. & Galushka, M. (2005). Sophia: A novel approach for textual case-based reasoning, *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Professional Book Center, Edinburgh, Scotland, UK, pp. 15–20.

Pereira, F. C. N., Tishby, N. & Lee, L. (1993). Distributional clustering of english words, *Meeting of the Association for Computational Linguistics*, pp. 183–190.

Petersen, W. (2004). A set-theoretical approach for the induction of inheritance hierarchies, *Electronic Notes in Theoretical Computer Science* **53**: 296–308.

Porter, B. W., Bareiss, R. & Holte, R. C. (1990). Concept learning and heuristic classification in weak-theory domains, *Artificial Intelligence* **45**(1-2): 229–263.

Porter, M. F. (1980). An algorithm for suffix stripping, Program.

Priss, U. (2006). Formal concept analysis in information science, *Annual Review of Information Science and Technology* **40**(1): 521–543.

Quinlan, R. J. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc.

Richter, M. M. (1995). The knowledge contained in similarity measures, *in* M. Veloso & A. Aamodt (eds), *1st International Conference on Case-Based Reasoning*, Springer.

Riesbeck, C. & Schank, R. (1989). *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates.

Riloff, E. (1996a). Automatically generating extraction patterns from untagged text, *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 1044–1049.

Riloff, E. (1996b). An empirical study of automated dictionary construction for information extraction in three domains, *Artificial Intelligence* **85**(1-2): 101–134.

Rissland, E. L. & Ashley, K. D. (1987). A case-based system for trade secrets law, *Proceedings of the 1st International Conference on Artificial Intelligence and Law*, ACM Press, pp. 60–66.

Rissland, E. L. & Daniels, J. J. (1995). A hybrid CBR-IR approach to legal information retrieval, *Proceedings of the 5th International Conference on Artificial Intelligence and Law*, ACM Press, pp. 52–61.

Roth-Berghofer, T. R. (2004). Explanations and case-based reasoning: Foundational issues, *in* P. Funk & P. A. G. Calero (eds), *Advances in Case-Based Reasoning*, Springer, pp. 389–403.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval, *Information Processing and Management* **24**(5): 513–523.

Salton, G., Fox, E. A. & Wu, H. (1983). Extended boolean information retrieval, *Communication of the ACM* **26**(11): 1022–1036.

Salton, G. & McGill, M. J. (1986). *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., New York, NY, USA.

Sanders, K. E., Kettler, B. P. & Hendler, J. A. (1997). The case for graph-structured representations, *Proceedings of the 2nd International Conference on Case-Based Reasoning*, LNCS 1266, Springer, pp. 245–254.

Sanderson, M. (1994). Word sense disambiguation and information retrieval, *Proceedings of the 17th International Conference on Research and Development in Information Retrieval*, Dublin, IE, pp. 49–57.

Sanderson, M. & Croft, B. (1999). Deriving concept hierarchies from text, *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, New York, USA, pp. 206–213.

Schmidt, R. & Gierl, L. (2000). Case-based reasoning for medical knowledge-based systems, *German Workshop on Experience Management*, pp. 720–725.

Schutze, H. & Pedersen, J. O. (1995). Information retrieval based on word senses, *Symposium on Document Analysis and Information Retrieval (SDAIR)*, pp. 161–175.

Simpson, G. B. (1984). Lexical ambiguity and its role in models of word recognition, *Psychological Bulletin* **96**: 316–340.

Sleator, D. & Temperley, D. (1993). Parsing English with a link grammar, *Proceedings of the 3rd International Workshop on Parsing Technologies*.

Small, S. & Rieger, C. (1982). Parsing and comprehending with word experts (a theory and its realization), *in* W. G. Lehnert & M. H. Ringle (eds), *Strategies for Natural Language Processing*, Erlbaum, Hillsdale, NJ, pp. 89–147.

Smyth, B., Keane, M. T. & Cunningham, P. (2001). Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control

software design, *IEEE Transactions on Knowledge and Data Engineering* **13**(5): 793–812.

Soderland, S., Fisher, D., Aseltine, J. & Lehnert, W. (1995). CRYSTAL: Inducing a conceptual dictionary, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*.

Taylor, B., Robertson, D., Wiratunga, N., Craw, S., Mitchell, D. & Stewart, E. (2007). Using computer aided case based reasoning to support clinical reasoning in community occupational therapy, *Computer Methods and Programs in Biomedicine* **87**(2): 170–179.

Thompson, P. (2001). Automatic categorization of case law, *Proceedings of the 8th international conference on Artificial intelligence and law*, ACM Press, New York, NY, USA, pp. 70–77.

TSB (2007). Air investigation reports, http://www.tsb.gc.ca/en/reports/.

Valencia-García, R., Ruiz-Sánchez, J. M., Vicente, P. J. V., Fernández-Breis, J. T. & Martínez-Béjar, R. (2004). An incremental approach for discovering medical knowledge from texts, *Expert Systems with Application* **26**(3): 291–299.

van der Plas, L. & Bouma, G. (2005). Syntactic contexts for finding semantically related words, *Computational Linguistics in the Netherlands*, LOT Utrecht.

van der Plas, L. & Tiedemann, J. (2006). Finding synonyms using automatic word alignment and measures of distributional similarity, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, ACL.

van Leeuwen, J. (ed.) (1990). *Graph algorithms*, MIT Press, Cambridge, MA, USA.

Volz, R., Handschuh, S., Staab, S., Stojanovic, L. & Stojanovic, N. (2004). Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the semantic web, *Journal of Web Semantics* **1**(2): 187–206.

Watson, I. (1997). Survey of CBR application areas, *Proceedings of the 2nd International Conference on Case-Based Reasoning*, Springer-Verlag, London, UK.

Watson, I. D. & Perera, S. (1998). A hierarchical case representation using context guided retrieval., *Knowledge Based Systems* **11**(5-6): 285–292.

Weaver, W. (1949/1955). Translation, *in* W. N. Locke & A. D. Boothe (eds), *Machine Translation of Languages*, MIT Press, Cambridge, MA, pp. 15–23. Reprinted from a memorandum written by Weaver in 1949.

Weber, R., Aha, D. W., Branting, L. K., Lucas, J. R. & Fernandez, I. (2000). Active case-based reasoning for lessons delivery systems, *Proceedings of the AAAI-2000 Workshop on Intelligent Lessons Learned*, AAAI Press.

Weber, R., Evanco, W. M., Waller, M. & Verner, J. M. (2004). Identifying critical factors in case-based prediction, *in* V. Barr & Z. Markov (eds), *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference*, AAAI Press.

Weber, R. O., Ashley, K. D. & Brüninghaus, S. (2005). Textual Case-Based Reasoning, *Knowledge Engineering Review* **20**(3): 255–260.

Wheeler, R. & Aitken, S. (2000). Multiple algorithms for fraud detection, *Knowledge-Based Systems* **13**(2-3): 93–99.

Wille, R. (1997). Introduction to formal concept analysis, *in* G. Negrini (ed.), *Modelli e modellizzazione. Models and modelling*, Consiglio Nazionale delle Ricerche, Instituto di Studi sulli Ricerca e Documentazione Scientifica, Roma, pp. 39–51.

Wilson, D. C. & Bradshaw, S. (2000). CBR textuality, *Expert Update* **3**(1): 28–37.

Wiratunga, N., Craw, S., Taylor, B. & Davis, G. (2004). Case-based reasoning for matching SmartHouse technology to people's needs, *Knowledge Based Systems* **17**(2-4): 139–146.

Wiratunga, N., Koychev, I. & Massie, S. (2004). Feature selection and generalisation for textual case retrieval, *7th European Conference on Case-Based Reasoning*, LNAI 3155, Springer, pp. 806–820.

Wiratunga, N., Lothian, R., Chakraborti, S. & Koychev, I. (2005). A propositional approach to textual case indexing, *Principles and Practice of Knowledge Discovery in Databases*, pp. 380–391.

Wiratunga, N., Lothian, R. & Massie, S. (2006). Unsupervised feature selection for text data, *Proceedings of the 8th European Conference on Case-Based Reasoning*, pp. 340–354.

Yong, N. U. (2004). *Text mining with information extraction*, PhD thesis.

Zaluski, M., Japkowicz, N. & Matwin, S. (2003). Case authoring from text and historical experiences, *Canadian Conference on Artificial Intelligence*, pp. 222–236.

Appendix A

Published Papers

- S. Asiimwe, S. Craw, N. Wiratunga and B. Taylor. Automatically acquiring structured case representations: The SMART way. In *Applications and Innovations in Intelligent Systems XV: Proceedings of the 27th BCS SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Springer, pages 45–49, 2007.

- S. Asiimwe, S. Craw, N. Wiratunga and B. Taylor. Automatic text standardisation by synonym mapping. In *Proceedings of the 12th UK Workshop on Case-Based Reasoning*, ISBN 978-1-904521-48-8, pages 88–98, 2007.

- S. Asiimwe, S. Craw, B. Taylor and N. Wiratunga. Case Authoring: from Textual Reports to Knowledge-rich Cases. In *Proceedings of the 7th International Conference on Case-Based Reasoning*, Springer, pages 179–193, 2007.

- S. Asiimwe, S. Craw, and B. Taylor. Discovering a concept hierarchy from SmartHouse Reports. In *Workshop Proceedings of the 8th European Conference on Case-based reasoning*, pages 88–97, 2006.