Case Authoring: From Textual Reports to Knowledge-Rich Cases

Stella Asiimwe[1], Susan Craw[1], Bruce Taylor[2], and Nirmalie Wiratunga[1]

[1] School of Computing
[2] Scott Sutherland School
The Robert Gordon University, Aberdeen, Scotland, UK
{sa, S.Craw, nw}@comp.rgu.ac.uk, B.Taylor@rgu.ac.uk

Abstract. SmartCAT is a Case Authoring Tool that creates knowledge-rich cases from textual reports. Knowledge is extracted from the reports and used to learn a concept hierarchy. The reports are mapped onto domain-specific concepts and the resulting cases are used to create a hierarchically organised case-based system. Indexing knowledge is acquired automatically unlike most textual case-based reasoning systems. Components of a solution are attached to nodes and relevant parts of a solution are retrieved and reused at different levels of abstraction. We evaluate SmartCAT on the SmartHouse domain looking at the usefulness of the cases, the structure of the case-base and the retrieval strategy in problem-solving. The system generated solutions compare well with those of a domain expert.

1 Introduction

Creating a case-based reasoning system can be quite challenging if the problem-solving experiences are captured as unstructured or semi-structured text [13]. This is because the system should be able to compare new problems with the textual case knowledge. Although IR-based techniques can be used to retrieve whole documents or snippets of documents, case comparison in this situation would only take place at word/phrase level. The features pertaining to the documents would still have to be compared using some domain/background knowledge or lexical source, in order to arrive at a useful ranking. Alternatively, a structured case representation can be created and the textual sources mapped onto it before they are used in reasoning [15]. This is quite difficult and the costs can be prohibitive if it is manually done by an expert.

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 [4]. For example, an occupational therapist that reads about a *wheelchair user* immediately thinks about the person's *mobility*. Hierarchical organisation of cases enables effective retrieval at different levels of problem abstraction [2]. The humans' ability to organise information into concepts, in order to extract meaning that is beyond the words they read, is what we attempt to mimic in our work.

Our Case Authoring Tool SmartCAT creates knowledge-rich cases from textual SmartHouse reports. Figure 1 shows the expert interacting with SmartCAT to sanction authored cases. SmartCAT uses the information embedded in text to learn a concept

R.O. Weber and M.M. Richter (Eds.): ICCBR 2007, LNAI 4626, pp. 179–193, 2007.

hierarchy. During case authoring, it maps the textual reports onto appropriate domain-specific concepts. SmartH-CBR is the resulting SmartHouse case-based reasoning system where the cases are organised in a hierarchy. In this paper, we examine the usefulness of the SmartCAT cases and the goodness of the SmartH-CBR retrieval strategy.

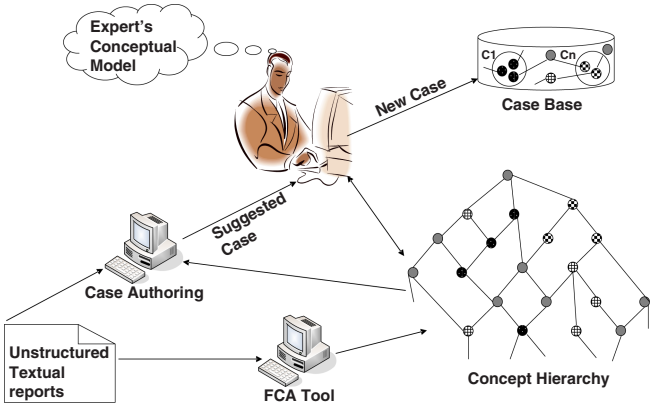

Fig. 1. Using the SmartCAT Tool

The rest of the paper is organised as follows. Related work is presented in Section 2 after which we describe the domain in Section 3. Section 4 details how relevant terms are extracted from the textual reports and used to represent the sub-problems. Section 5 presents the creation of the concept hierarchy which is used to organise the cases in Section 6. The usefulness of the cases and the adopted retrieval strategy are evaluated in Section 7 before our concluding remarks in Section 8.

2 Related Work

Early TCBR systems like FAQ Finder [5] retrieved whole unstructured documents or document snippets as cases. IR-based retrieval was typically employed and understanding case content was not a system requirement. More recent research has focused on creating more knowledge-rich case representations. One approach maps the textual case documents to a structured case representation, but in much of this work the case representations are acquired manually. This renders the systems more difficult to maintain. Examples include Wiratunga et. al.'s work [14] and systems created using frameworks like jCOLIBRI [10]. Further efforts have tried to acquire case representations automatically. Sophia [7] employs term distribution to create word-groups that co-occur in similar documents and the word clusters can represent the textual documents.

In domains like SmartHouse where adaptation knowledge is difficult to acquire, the effectiveness of the retrieval stage of CBR is also crucial. Both Bergmann & Wilke and Watson & Perera demonstrate that cases represented as a hierarchy of smaller cases lead to more efficient retrieval than using a simple flat case representation [2,12].

Déjà Vu [11] 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. We address this shortcoming by creating a concept hierarchy which enables us to automatically author cases and organise them in a hierarchy. Our earlier work was a first step towards building a concept hierarchy [1]. The availability of more reports allows us to incorporate Latent Semantic Indexing to identify relevant attributes for Formal Concept Analysis.

3 The SmartHouse Domain

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 2 is an excerpt from a typical report. First, it briefly summarises the person's disabilities which are referred to as a type of *problem*, *difficulty*, *impairment*, or a disabling medical condition like *dementia*. Thus, a person with *impaired hearing* is referred to as having a *hearing difficulty*, *problem* or *impairment*. To distinguish disabilities from other terms in the text we refer to them as *disability terms*.

Ms M was a powered wheelchair user with very limited mobility. She had severe curvature of the spine, and this disability created difficulty when she attempted to carry out everyday tasks around her home. A number of problems were identified:

Door Opening

Ms M did have a door-key fitted to a special fob that allowed her to unlock or lock her door, but once it was locked, she found it physically impossible to open the door due to her ***mobility problem****... ... and the final choice of equipment consisted of:*

Lighting Controls

The use of lighting controls was limited to two table lamps in the livingroom and one lamp in the bedroom... The lights were simply switched off or on by the operation of the GEWA control unit.

Door Opening Motor and Lock Release

These devices were intended to allow Ms M to unlock and open her door unaided...

Fig. 2. Report Excerpt

Next, sub-problem sections describe ways in which the person's disabilities manifest themselves. Each is dedicated to a given area of difficulty. The excerpt shows a sub-problem where the person found *door opening* to be cumbersome. Every sub-problem is given a summary heading, but they do not always accurately describe the area of difficulty. *Intercom operation* may refer to a person's difficulty in using their intercom but this could be due to a *hearing impairment* or a *mobility problem*. Typically, the person's disabilities are mentioned in the summary but these need not be repeated in the sub-problem text where symptoms and problem-areas are elaborated. The summary and all sub-problems make up the problem-part of each SmartHouse case. Each section

that describes an area of difficulty is called a *sub-problem* since they describe only a part of the SmartHouse case problem.

Lastly, the report mentions the solution package which lists the SmartHouse devices, 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. In Figure 2 *Door Opening Motor and Lock Release* is the solution for the *Door Opening* sub-problem. A report records one or more sub-problems with accompanying solutions.

4 Identifying Relevant Terms

It is important for SmartHouse cases to capture useful problem features if they are to be re-used in problem-solving. However, the effectiveness of a case-based reasoning system also depends on its ability to compare cases. This requirement influences the choice of case representation. It has been observed that humans interpret text at a concept level and not merely at the level of words they read [4]. In fact, an occupational therapist will list problem features under predefined domain-specific concepts. For example the problem-features *uses walking sticks* and *abnormal gait* will be recorded under a *mobility* concept. Therefore, not only do our cases need to capture relevant domain knowledge, we also need to be able to map the knowledge onto domain-specific concepts where concepts are useful groupings of disabilities, disabling conditions, or areas of difficulty.

In order to author cases that capture the relevant domain knowledge and allow for effective comparison and retrieval, we need to do the following steps:

1. extract knowledge embedded in the text and represent the textual reports with knowledge-rich terms;
2. use the knowledge to create a concept hierarchy;
3. map the problem representations onto the domain-specific concepts; and
4. attach solutions to the appropriate parts of the hierarchy.

In the real world, queries are lists of problem descriptors for which a list of SmartHouse devices is sought. Therefore, we focus on creating problem concepts for the SmartHouse domain. Thus cases will be represented as groups of domain-specific problem concepts and a corresponding list of SmartHouse devices that solve the problems. We start by trying to identify and extract terms that are meaningful with respect to SmartHouse problems. We will then use these terms to represent the problem parts of the textual reports.

4.1 Term Extraction from Textual Reports

We extract information in the form of trigrams, bigrams and a few necessary unigrams, since single words do not generally carry useful knowledge about the SmartHouse domain. This is to ensure that only potentially knowledge-rich text is mined for domain-specific concepts. First, we carry out some pre-processing where, each problem part of a

report is separated into sentences. A sentence is taken to be a group of words that is separated by a period, comma, bracket or other delimiting punctuation. Non-alphanumeric and numeric characters are removed next and the final pre-processing step applies the UEA-Lite [6] conservative stemmer that ensures words are reduced to forms that are complete words. The main aim is to make the case representations and the nodes in the concept hierarchy easy to comprehend and to allow manual refinement. Next, terms are 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 or are substrings of trigrams.
3. All non-stopword unigrams are extracted, discarding all that are sub-strings of trigrams or bigrams.

It should be noted that we discard only those terms that are substrings of other terms in the **same** sentence. This is to avoid unnecessary duplication, but at the same time ensuring that short terms are not 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 every word in the document can be represented as itself or as a 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 extracted and the single word "*case*" will be ignored 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 extracted if it appears in a sentence like "*A case is made up of a problem and solution part*". The problem part of each SmartHouse report is transformed into a set of terms containing stemmed words. The terms may contain stopwords but will not start or end with one. Extracted terms are meaningful because they have not been distorted by the removal of stopwords. We obtain 731 terms from 38 problems. These terms will be filtered to obtain those that actually contain useful knowledge.

4.2 Latent Semantic Indexing

We make use of Latent Semantic Indexing (LSI) [3] to identify useful terms out of the trigrams, bigrams and unigrams we have extracted. The problem part of a SmartHouse report is regarded as a *document* because we identify useful terms by learning their associations at case level. We represent the terms and the documents as an incidence term $\times$ document matrix A. Entry a_{ij} is the product of a local log frequency weighting and a global log entropy weighting of a term *i* in document *j*.

$$a_{ij} = \log_2(f_{ij} + 1)(1 - \sum_{k=1}^{N} \frac{P_{ik} \log_2(P_{ik})}{\log_2(N)})$$

where f_{ij} is the frequency of term *i* in document *j*, and P_{ik} is the relative frequency of term *i* in document *k*, compared to the collection of N documents.

LSI employs Singular Value Decomposition to decompose the term $\times$ document $m \times n$ matrix A as:

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

U_O represents the term matrix, V^T_O the document matrix, and S_O is a diagonal matrix containing singular values arranged in descending order. Each column in U_O represents a *topic* or *concept* and it captures terms that appear in that concept. The r highest singular values identify the r most important concepts in U_O. These r concepts are referred to as *LSI-concepts* in order to disambiguate them from later concepts in the paper. Keeping only the r highest singular values removes noisy dimensions and gives the lower rank approximation $\tilde{A}$, of the original matrix A.

$$\tilde{A}_{(m\times n)} = U_{(m\times r)} \times S_{(r\times r)} \times V^T_{(r\times n)} \qquad \text{(shaded in Figure 3.)}$$

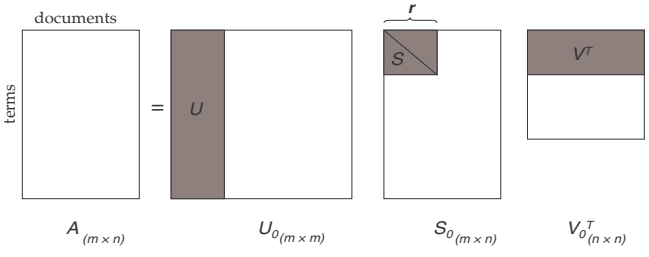

Fig. 3. Singular Value Decomposition and Latent Semantic Indexing

We accentuate the entries of $U_{(m\times r)}$ by multiplying it by $S_{(r\times r)}$ to obtain a $term \times concept$ matrix. The weights of terms in the matrix are a measure of the importance of the individual terms to the key LSI-concepts in the document collection. We use the top ten singular values ($r = 10$). Figure 4 illustrates a portion of the $term \times concept$ matrix obtained for the SmartHouse domain. The term *mobility problem* is most important in concept *C4* (weight *8.02*) and least important in concept *C2* (weight *-1.74*). It is the '*importance*' score and the groupings of terms as *LSI-concepts* that we exploit in order to identify knowledge-rich terms.

4.3 Term Filtering

We reduce the search space for finding relevant terms by making use of term weights in key LSI-concepts. In this domain, the areas of difficulty described in each sub-problem are a result of the person's disabilities and this helps us to target our search for relevant terms to only those terms that are related to disability terms. Pattern-matching with the words *difficulty*, *problem* and *impairment* and a list of disabling conditions are used to identify the disability terms. The list is compiled using brochures from the website of Tunstall, a leading provider of telecare solutions. The assumption we make here is that

	C1	C2	C3	C4	…	Cn
risk from fire	7.29	1.79	0.85	-1.93	…	0.76
weak grip	-0.36	1.28	1.48	6.55	…	0.87
unable to hear	0.31	6.11	0.71	-1.47	…	-1.98
→ **mobility problem**	**0.92**	**-1.74**	**1.12**	**8.02**	…	**2.51**
wheelchair	0.87	0.36	1.08	7.53	…	2.46
dementia	7.48	1.69	0.96	0.86	…	0.69
door open	1.39	1.30	1.87	6.95	…	2.30

Fig. 4. Term-Concept Matrix Showing Term Importance

"in a linear combination of terms in which the disability term is important, all other terms that are nearly as important, will be relevant to the disability." Thus disability terms are used as anchor terms to identify relevant terms.

We shall use the example in Figure 2 to illustrate the term filtering process. Consider the section describing the *door opening* sub-problem. The underlying disability is mentioned here as a *mobility problem*. Therefore terms that are relevant to the *door opening* problem are closely related to the disability term *mobility problem*. In Figure 4 we take the row in the $term \times concept$ matrix that represents the term *mobility problem* and look for the LSI-concept in which it has the highest weight. In the example, this is concept *C4*. We then set a threshold and extract terms whose weights are nearly as high in this concept. So the terms *wheelchair*, *weak grip* and *door open* are identified as being important. We look for those terms that *actually* appear in the sub-problem text. These will be the representative terms for this sub-problem. The effect is that we extract terms that are 'important' in LSI-concepts in which the disability term is *most important*. Thus we extract terms that are as informative as the disability term. This leaves only 238 terms rather than the 731 we had originally.

A sample of the discovered relevant terms is compared to text where an expert was asked to highlight key phrases. Figure 5 shows (stemmed) text highlighted in bold by the expert and LSI respectively for one sub-problem. The different n-grams are underlined in the latter case. 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 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.

We use the relevant terms obtained to represent the sub-problems. We also include the disability terms among the sub-problem representative terms since they may not be explicitly repeated in the description.

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

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.*

Fig. 5. Text Highlighted by Expert and LSI

5 Creation of a Concept Hierarchy

Concept-superconcept relationships do not exist in the semantic structure captured by LSI. However, Formal Concept Analysis (FCA) yields a concept hierarchy where the concepts are ordered according to their concept-superconcept relationships. We employ FCA to create a concept hierarchy using the knowledge we have extracted using LSI. A brief description of FCA and how its use to create the concept hierarchy now follows.

5.1 Formal Concept Analysis

FCA is used to represent and analyse data in information science [8,9]. 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. A formal context is often represented as in Figure 6. The different SmartHouse sub-problems form the set of objects, and some possible features of the sub-problems form the set of attributes. For example *telephone operation* has an attribute *hearing impairment* arising from a person with a *hearing impairment* who has difficulties operating their telephone. FCA uses a formal context to produce 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. In Figure 6, 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.

The concept lattice resulting from the context in Figure 6 is shown in Figure 7. 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. So as you descend the hierarchy and therefore

OBJECTS	ATTRIBUTES							
	hearing impairment	wheelchair	mobility problem	poor flexibility in the joints	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				

Fig. 6. Context for some SmartHouse Problems

become more specific (and 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 attributes reduces as one ascends the hierarchy.

The objects associated with a concept are called its extent, and the attributes describing the concept are called its intent. Node 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*}.

To prevent cluttering, reduced labeling is used. An attribute is attached to the top-most concept that has the attribute in its intent. The attribute occurs in all intents of concepts that are reachable by descending the subtree from which it is attached. Node 3 represents a concept whose intent is {*mobility problem*, *lack of strength in hands*, *multiple sclerosis*}. Conversely, an object is attached to the bottom-most concept where it is part of the extent. Every concept that is reachable by ascending from this point to the top-most concept has the object in its extent. Node 1 represents a concept whose extent is {*intercom operation*, *window operation*, *door operation*}. A description of how we obtain the objects and attributes for use in FCA now follows.

5.2 FCA Objects and Attributes

Each FCA object and associated attributes result in a concept and super-concepts 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 we use each sub-problem as an FCA object. For easy identification, we name the FCA objects using the problem summary heading

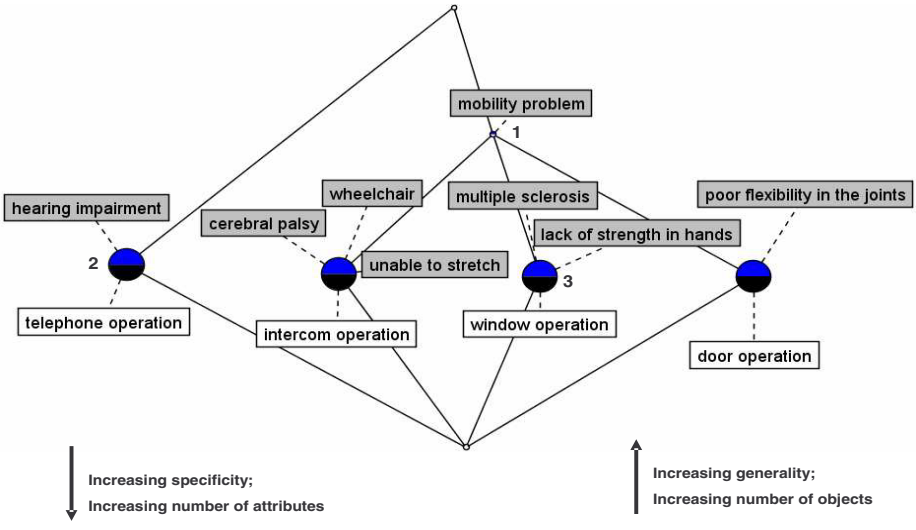

Fig. 7. Example Lattice

and the underlying disability. Thus the objects can be easily identified in the lattice which in turn, makes it easier for an expert to refine the hierarchy as necessary.

FCA attributes are features of the FCA objects. Each sub-problem is an FCA object and it follows that the terms describing the objects are used as FCA attributes. Thus the sub-problem representative terms identified using LSI, and the corresponding disability term, become the attributes for the sub-problem FCA objects. FCA is applied to a context of attributes and their corresponding objects to create a concept hierarchy.

6 Case Representation and Organisation

The concept hierarchy is used to define and organise cases in the SmartHouse domain. Figure 8 shows a portion of the SmartHouse problem concept hierarchy we have built. Normally, an occupational therapist would record a person's disabilities, problem areas and symptoms, under pre-defined groupings: *wheelchair* would be recorded under *mobility*; *learning difficulties* under *cognitive problems*. Similarly, the case representation task involves mapping the problem-representative terms onto the discovered concepts in the hierarchy and attaching a set of solutions that assists with the problem.

We map each sub-problem on to a concept by finding one whose *whole* intent is *all* of the sub-problem's representative terms. Node 5 in Figure 8 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 5 plus the corresponding disability term. Thus the *door opening* sub-problem is mapped on to the concept represented by node 5. We shall use the term *concrete* to refer to those concepts onto which a sub-problem is

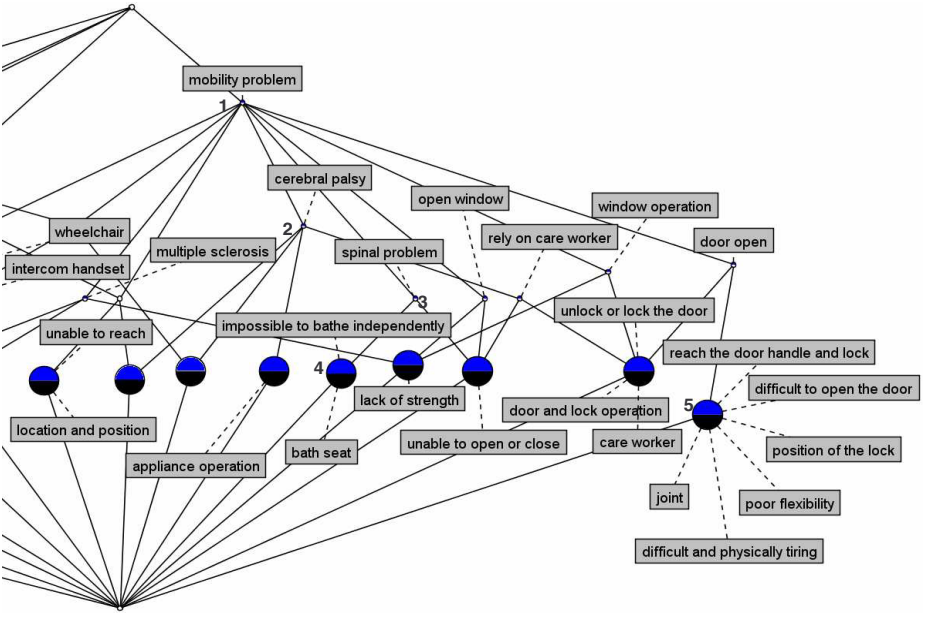

Fig. 8. Concept Activation During Retrieval

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. We also refer to abstract concepts on to which no sub-problem is mapped, as *completely-abstract*. Mapping each sub-problem on to a concept transforms the problem-part of the original textual case to a list of concrete concepts. A case that contains n sub-problems is mapped onto n concrete concepts in the concept hierarchy.

We map each sub-problem on to a corresponding list of SmartHouse devices by making use of overlaps between the words in the sub-problem and solution description text. We also take advantage of device names because sometimes they reflect the sub-problems they assist with. For example, *Door Opening Motor and Lock Release* in Figure 2 is the solution to the *door opening* sub-problem mentioned in the report excerpt. Thus we map each sub-problem and invariably each concrete concept on to a list of SmartHouse devices. Consequently, a case becomes a list of concrete concepts each of which is tagged with a list of SmartHouse devices that solve the different sub-problems. All this is implemented in a case-based reasoning system called SmartH-CBR.

During problem-solving, the concept hierarchy is searched for the most specific concepts matching the query terms. Ideally, a concept is activated by a 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, SmartH-CBR returns

the **disjunction** of all solutions of concrete concepts that are reachable by descending the subtree from which the completely-abstract concept is attached. If all we knew about a problem was that the person has *cerebral palsy* and a *mobility problem*, nodes 1 and 2 in Figure 8 would be activated. Without any further knowledge, it is difficult to predict if the person also has a *door opening* problem, an *appliance operation* problem 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.

7 Evaluation

We judge the usefulness of the authored cases by testing whether they capture knowledge that is useful for finding solutions. We also test whether the retrieval strategy employed by organising the cases in a hierarchy results in retrieval of useful solutions. We compare SmartH-CBR's and the expert's solution packages for four problems shown in Figure 9. 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, we have no cases whose problems are completely described by the same terms. Problem *C* is a problem part of a report that is excluded in the creation of the hierarchy. It is a test of whether the rest of the cases in the case base capture enough knowledge about the domain in order to give useful solutions to this problem that has not influenced the case representation. It is also a test of SmartH-CBR's ability to find appropriate parts of useful cases in order to reuse them for problem-solving. Problem *D* is another test of the same sort as Problem *C*. 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

Fig. 9. 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. Therefore, we test the system'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 enumerated. This makes the problems harder to solve than those where the specific needs can be targeted in the search for a solution. Thus problem *B* is harder to solve than *A* and problem *D* is harder than *C*.

SmartH-CBR attempts to index each problem in order to retrieve the appropriate solutions. The possession of features that are familiar to SmartH-CBR makes problems *A* and *B* easier to index than problems *C* and *D*. Figure 10 illustrates the solution packages offered by SmartH-CBR and the expert for problems *A*, *B*, *C* and *D*. The similarity of solutions for SmartH-CBR and the expert are compared using precision and recall. In the SmartHouse domain, precision is the proportion of SmartHouse devices proposed by SmartH-CBR that occur in the expert solution package; recall is the proportion of devices proposed by the expert that are also proposed by SmartH-CBR. 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 38 composite cases (reports) and 90 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**	

Fig. 10. SmartHouse Devices for Test Problems

Problem *A* was the easiest to solve and this is confirmed by the high values of recall obtained by SmartH-CBR. It activates nodes 3, and 4 in Figure 8. Node 4 is a concrete node which results in the return of the attached solution *shower with sitting facility*. However, two of the paths end at completely-abstract concept nodes which results in the generation of two additional devices. Nevertheless, SmartH-CBR recommends all the solutions that are proposed by the expert hence obtaining high recall for this problem.

Problem *B* was harder to solve as SmartH-CBR was required to find devices that would help the person, without knowledge of the person's specific needs. It has a wider space to search and was therefore more prone to returning solutions for completely-abstract concept nodes. Hence the poorer values of precision and recall.

In its search for a solution to the *intercom operation* sub-problem in problem *C*, SmartH-CBR activates a concrete node that results in the return of the solution *visual doorbell*. One interesting thing to note though is that, for the *television* sub-problem, SmartH-CBR 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 SmartH-CBR recommends the right solution by returning solutions of sub-cases attached to a completely-abstract node at which the search path ends.

Problem D was the most challenging. The fact that SmartH-CBR 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. However, the hierarchy has 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.

8 Conclusions and Future Work

We have presented SmartCAT, a case authoring tool that creates knowledge-rich cases from semi-structured textual reports. SmartCAT uses SmartHouse problem-solving experiences to learn a concept hierarchy. It then organises the knowledge-rich cases into a structure based on concept-superconcept relationships in the concept hierarchy. The result is SmartH-CBR, a hierarchically structured case-base where abstract cases and sub-cases exist at several levels of abstraction and all nodes whose intents completely represent a sub-problem are tagged with sub-solutions.

We obtain good results for precision and recall on the test cases. This is partly because tagging sub-problems with their solutions helps SmartH-CBR to retrieve only the relevant part of an otherwise composite solution. SmartH-CBR's ability to recommend sensible solutions can also be attributed to the retrieval mechanism. The use of the hierarchy as the basis for retrieval ensures the return of some form of solution. This is particularly important in domains where high recall is preferred to high precision.

Unlike most textual case-based reasoning systems, SmartH-CBR's case knowledge and the structure of the case-base are generated automatically. This feature makes the case knowledge for SmartH-CBR easy to acquire and maintain from textual records. This is the main novelty in our work. The use of FCA and LSI in CBR is not novel but using LSI to enrich FCA in order to exploit the resulting FCA hierarchy in CBR has not been explored by others.

The concept hierarchy could 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. The extents in the concept hierarchy have been named using both a sub-problem header and the discovered disability term. These should be informative descriptions of the underlying problem for the expert. The intents consist of terms that an expert is likely to have chosen as key phrases. Therefore it is easy for an expert to see the attributes in context and amend the hierarchy as necessary.

References

1. Asiimwe, S., Craw, S., Taylor, B.: Discovering a concept hierarchy from SmartHouse reports. In: Proc 3rd Textual Case-Based Reasoning Workshop, 8th European Conference on Case-Based Reasoning (2006)
2. Bergmann, R., Wilke, W.: On the role of abstraction in case-based reasoning. In: Proc 3rd European Workshop on Advances in Case-Based Reasoning, pp. 28–43. Springer, Heidelberg (1996)
3. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
4. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proc 20th International Joint Conference for Artificial Intelligence (2007)
5. Hammond, K., Burke, R., Martin, C., Lytinen, S.: FAQ Finder: a case-based approach to knowledge navigation. In: Proc 11th Conference on Artificial Intelligence for Applications, p. 80. IEEE Computer Society, Los Alamitos (1995)
6. Jenkins, M.-C., Smith, D.: Conservative stemming for search and indexing. Special Interest Group on Information Retrieval (2005)
7. Patterson, D., Rooney, N., Dobrynin, V., Galushka, M.: Sophia: A novel approach for textual case-based reasoning. In: Proc 19th International Joint Conference on Artificial Intelligence, pp. 15–20 (2005)
8. Petersen, W.: A set-theoretical approach for the induction of inheritance hierarchies. Electronic Notes in Theoretical Computer Science 53, 296–308 (2004)
9. Priss, U.: Formal concept analysis in information science. Annual Review of Information Science and Technology, 40 (2006)
10. Recio, J.A., Díaz-Agudo, B., Gomez-Martin, M., Wiratunga, N.: Extending jCOLIBRI for textual CBR. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 421–435. Springer, Heidelberg (2005)
11. Smyth, B., Keane, M.T., Cunningham, P.: 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 (2001)
12. Watson, I.D., Perera, S.: A hierarchical case representation using context guided retrieval. Knowledge Based Systems 11(5-6), 285–292 (1998)
13. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual Case-Based Reasoning. Knowledge Engineering Review 20(3), 255–260 (2005)
14. Wiratunga, N., Craw, S., Taylor, B., Davis, G.: Case-based reasoning for matching SmartHouse technology to people's needs. Knowledge Based Systems 17(2-4), 139–146 (2004)
15. Wiratunga, N., Koychev, I., Massie, S.: Feature selection and generalisation for textual case retrieval. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 806–820. Springer, Heidelberg (2004)