Knowledge Sources for Textual CBR Applications

Mario Lenz*

Abstract

Textual CBR applications address issues that have traditionally been dealt with in the Information Retrieval community, namely the handling of textual documents. As CBR is a knowledge-based technique, the question arises where items of knowledge may come from and how they might contribute to the implementation of a Textual CBR system. In this paper, we will show how various pieces of knowledge available in a specific domain can be utilized for acquiring the knowledge required for a CBR system.

Keywords: Case-Based Reasoning, Textual CBR, Information Retrieval.

Introduction

The fundamental idea behind case-based reasoning (CBR) is to reuse knowledge obtained from earlier problem solving situations in a similar context. In practice, however, these *experiences* are very often stored in textual documents. Probably the best-known examples are collections of Frequently Asked Questions (FAQ) which are used in virtually any area. Other examples are documentations, manuals of technical equipment, reports by physicians etc.

Consequently, CBR researchers started to address issues of textual documents under the heading of *Textual CBR*. A key property distinguishing Textual CBR from other techniques, such as more traditional Information Retrieval, is the possibility to consider *knowledge* during the reasoning process and, thus, to capture domain-specific expertise.

For specific projects, however, the question arises what kind of knowledge can be acquired and how this should be encoded within the CBR system. In this paper, we will address this question; we will show how the required knowledge may be obtained and how it can be represented in such a way that a Textual CBR system can benefit from it. To ease the understanding, we will describe a typical application area of Textual CBR first.

*AI Lab, Dept. of Computer Science, Humboldt University, D-10099 Berlin, lenz@informatik.hu-berlin.de

An Application Scenario

Before describing in detail how domain-specific knowledge can be used in a Textual CBR system, let us consider a particular application scenario in which Textual CBR is a promising technique.

The Hotline Scenario

Today it is becoming more and more difficult to sell products without a reliable and efficient customer support. This is true both, for industrial equipment as well as for highly complex software systems. The reason for this is that maintenance costs often exceed the initial value and thus become a decision criterion of customers (Lenz *et al.* 1996). Consequently, many companies establish so-called *help desks* and *hotlines* which the customer may contact in case of a problem.

These applications can be characterized as follows:

- A hotline will always focus on a specific domain, e.g., on some type of technical devices, on a set of software components etc. This implies that no topics outside this domain will be dealt with. Also, a number of highly specific terms will be used such as names of components, functions and modules.
- The term *hotline* does not mean that customer enquiries have to be handled within seconds. Rather, finding the answers to problems usually involves a complex problem analysis and thus may take some time. Nevertheless, questions concerning problems that have been solved before should, of course, be answered rapidly.
- Naturally, a hotline will rely very much on textual documents such as FAQs, documentations, and error reports. Also, the customers' queries will be given in free text plus some additional information such as names of hardware components, software release numbers etc.

Consequently, there is an urgent need for techniques which can support the hotline staff in retrieving relevant documents in a specific problem context. Obviously, assessing relevance somehow requires capturing the meaning of the corresponding documents. This is not possible by simply matching common keywords.

For this type of application, Textual CBR seems promising because CBR systems can in principle take into account

- domain-specific knowledge helping to relate different terms;
- knowledge about the structure of documents, e.g., for weighting different components;
- knowledge about the structure of the domain, e.g., for focusing the search on a specific sub-area.

On the other hand, we only consider applications here which focus very much on the *retrieval* aspect rather than performing any deep reasoning based on the documents / cases; also no adaptation is performed.

Knowledge for Textual CBR

A major argument in favor of Textual CBR is that this paradigm can utilize virtually any kind of knowledge available in order to improve the system behavior. In general, when dealing with the implementation of a knowledge-based system, one has to answer at least the following three questions:

1. Where can knowledge be added to the system in order to obtain maximal benefits; that is, what are appropriate *knowledge containers*?
2. How should this knowledge be represented in the chosen *knowledge container*?
3. How can the required knowledge be acquired; i.e. what are the *knowledge sources*?

We will give possible answers to these questions in the following. These answers summarize the experiences collected from a number of projects which will be used throughout the paper for illustration purposes.

Knowledge Containers

In terms of the knowledge container model introduced by Richter (Richter 1995), every piece of knowledge can be represented in one (or more) of the following categories:

(a) the collection of cases;

(b) the definition of an index vocabulary for cases;

(c) the construction of a similarity measure;

(d) the specification of adaptation knowledge.

For the purpose of this article, we assume that **(a)** documents are available which can be considered as cases, and **(d)** adaptation is of limited use only. Consequently, the main containers in which knowledge can be filled are **(b)** the terminology of case representation and **(c)** the similarity measure used to compare cases.

Knowledge Layers

According to our experiences from the performed projects, it appears to be advantageous to separate the knowledge in several *knowledge layers* each of which is responsible for handling specific types of knowledge that may occur in the documents:

Keyword Layer: contains some kind of keyword dictionary which is used for recognizing simple keywords, for ignoring stop-words etc.;

Phrase Layer: contains a dictionary of domain-specific expressions used for recognizing more complex phrases not normally used in general-purpose documents, such as names of modules and devices;

Thesaurus Layer: contains information about how various keywords relate to each other in terms of (linguistic) similarity;

Glossary Layer: contains information about how elements of the keyword and phrase layers relate to each other in terms of domain-specific similarity;

Feature Value Layer: contains a set of features and their values as they might occur in the specific domain, such as names and release numbers of operating systems, physical measures etc.;

Domain Structure Layer: contains a description of the domain structure allowing some *clustering* of documents, an example would be the distinction of printers in laser and ink-jet printers;

Information Extraction Layer: contains an IE module which is used to automatically extract structured information, feature values etc. from the textual descriptions.

Note that in the above order the knowledge layers become more and more knowledge-intensive; in general each layer is set on top of the previous ones.

In terms of the knowledge container model, the keyword, phrase, and feature value layers belong to the terminology of case representation whereas the remaining layers describe a piece of the similarity measure.

Knowledge Sources

After having established the above layers as a desired architecture for encoding knowledge of a Textual CBR system, the question arises as to how each of these layers can be filled for a specific application. In general, a careful knowledge engineering process is necessary for this purpose and each layer requires specific procedures and techniques.

Keyword Layer: As mentioned above, the keyword layer basically consists of a keyword dictionary plus a keyword parser using this dictionary for recognizing simple expressions in the documents. For building the keyword dictionary, techniques known from the Information Retrieval community (Salton & McGill 1983) can be used, such as:

- statistics about frequency of terms;
- stop-word lists;
- information on stemming of words.

To go beyond these statistical techniques and include linguistic knowledge, we also used *part-of-speech tagging* to obtain information about semantic categories of

words as well as more advanced stemming information. This is particularly useful as some of our applications deal with German texts and we, thus, have to cope with an even more irregular grammar.

Phrase Layer: For the phrase layer, too, a parser is required which runs on a defined phrase dictionary and recognizes domain-specific phrases and a given document. The major differences to the above keyword layer are that

(a) phrases can not normally be obtained in sufficient quality by pure statistical analysis of a document collection;

(b) recognizing phrases is more complicated than parsing simple keywords because parts of a phrase may occur separated in a sentence;

(c) the keyword layer may be reused in other applications whereas the phrase layer is highly domain-specific.

To construct the phrase dictionary, application-specific knowledge sources have to be used. These include

- documentations and manuals of (software) products from which names of modules, components, menus etc. can be extracted;
- product catalogues containing detailed descriptions of products, releases, versions etc.
- more general dictionaries containing expressions not specific to a particular application but to some application area; for example, FOLDOC[1] provides a glossary of computer science terms.

During the knowledge acquisition process, these knowledge sources have to be scanned for relevant terms, i.e. for terms and phrases that might occur in the documents. In this phase, the integration of domain-experts is essential.

Furthermore, a more advanced parser is required which is able to recognize phrases in the texts.

Thesaurus Layer: The task of the thesaurus layer is to relate different keywords to each other. For example, information about synonyms, more abstract, and more specific terms appears to be highly useful.

For this, a general purpose thesaurus can be used, such as WORDNET[2] (Miller 1995). Simply speaking, such a tool can be used for extracting various types of relations between a given pair of keywords and to assign a certain similarity value depending on this relation.

Unfortunately, WORDNET is currently available for English only whereas some of our projects have to deal with German texts. To obtain a certain amount of thesaurus information for these applications, we utilized the fact that German texts very often use composite nouns. From these an abstraction hierarchy can be derived semi-automatically. More precisely, composite nouns can be used to automatically construct lists of related terms which afterwards have to be scanned and checked manually. Based on our experiences, about 80% of the relationships derived this way are, indeed, meaningful.

Glossary Layer: Similarly to the thesaurus layer, the glossary layer is used to relate application-specific terms to each other by specifying similarity relations. Major differences to the thesaurus layer are that application-specific terms will hardly be contained in general purpose thesauri and even if they are, the actual meaning, or relations, between two terms may change for a specific application.

Consequently, an application-specific thesaurus has to be built for which glossaries and other sources in which terms are explained can be utilized. To a large extent, knowledge engineering techniques are required here, i.e. in cooperation with domain experts a model has to be built which relates application-specific terms to each other and thus provides the basis for a similarity measure.

Feature Value Layer: The feature value layer is designed to contain information about attributes and their values relevant to an application. These can be obtained by discussing with domain experts which attributes are normally used to describe the products and services of the application at hand. Also, an existing document collection can be scanned to see which feature values actually occur. According to our experiences, such feature values are often separated from other parts of the documents in that special sections are used to encode these.

Given the knowledge about relevant feature values one has to consult the domain experts again in order to construct an appropriate similarity measure which should include weightings of the various features as well as descriptions of similarity between the various feature values.

Domain Structure Layer: The domain structure layer is built on top of the feature value layer in the sense that very often particular features exist which classify documents with respect to some application area, or *topic*, they belong to. This can be a very useful information in particular if there are areas which appear to be disjoint. Using the *topic* classification, entire sets of documents can be safely ignored and thus precision can be improved greatly.

Information Extraction Layer: Though documents often contain a special section from which feature values can be obtained, sometimes the textual parts of documents also contain expressions which should better be encoded in terms of a more structured representation, such as attribute-value pairs. Also, queries posed by users normally lack an appropriate structuring. Consequently, a certain amount of structural information should be extracted directly from the texts. For this we

[1] *The Free On-line Dictionary of Computing* http://wagner.princeton.edu/foldoc/

[2] http://www.cogsci.princeton.edu/~wn

utilized Information Extraction (IE) techniques (Riloff & Lehnert 1994) which scan the textual components for specific triggers and try to recognize additional information in the form of attribute-value pairs.

Knowledge Acquisition Effort

Filling all the above knowledge layers obviously requires considerable effort once a particular Textual CBR system has to be built. Fortunately, some parts of knowledge acquisition can be reused for a number of applications whereas other parts remain highly domain-specific.

In particular, the keyword and the thesaurus layer should be reusable across domains as long as the language of documents remains the same. All other layers, however, are highly domain-specific and, hence, require additional effort for each new application. According to our experience, however, building the keyword and thesaurus layers is most time-consuming due to the vast number of terms involved.

Case Studies

After having described in principle how the various knowledge layers can be filled, we will now briefly sketch how the required information has been obtained in some projects performed. For details to these projects the reader is referred to the corresponding publications.

FALLQ

FALLQ has been a prototypical implementation for LHS, a leading developer of customer care and billing software in the telecommunication market (Lenz & Burkhard 1997; Kunze & Hübner 1998). The objective of the system was to improve in-house knowledge management by making previously observed (and solved) problems available to the entire staff, in particular to the customer support group. FALLQ is prototypical in so far as integration and maintenance issues are not addressed sufficiently yet whereas the Textual CBR part has been fully implemented.

A major advantage at LHS was that a number of document types had been defined which served as guidelines for describing the various business processes, such as a customer request or a defect description. Based on these, a case base could be built directly. The knowledge has been obtained as follows:

Keyword Layer: derived by analyzing a document collection, using statistics about term frequencies and such, plus linguistic tools and tables to include various word forms

Phrase Layer: filled by analyzing documents with respect to multi-word expressions occurring often, plus databases available at LHS containing names of devices, modules, functions etc.

Thesaurus Layer: based on some machine-readable thesauri, manually converted and corrected

Glossary Layer: based on a glossary provided by LHS, manually converted

Feature Value Layer: based on features that occurred in the document collection, LHS experts decided about inclusion

Domain Structure Layer: currently not used

Information Extraction Layer: currently not used

ExperienceBook

The EXPERIENCEBOOK (Kunze & Hübner 1998) is an in-house system developed for supporting system administration at Humboldt University Berlin. In contrast to FALLQ, documents that could serve as a starting point for constructing a case base had to be collected first. This was achieved by searching appropriate news groups as well as reusing the personal documentations of system administrators.

The various knowledge layers had then been filled by resuing parts of the FALLQ project:

Keyword Layer: built based on FALLQ keyword layer, extended and adjusted by analyzing document collections

Phrase Layer: inclusion of FOLDOC terms, manually filtered and corrected

Thesaurus Layer: utilization of WORDNET, manually filtered and corrected

Glossary Layer: currently not used

Feature Value Layer: simply included attributes such as operation systems and machine types

Domain Structure Layer: currently not used

Information Extraction Layer: currently not used

SIMATIC Knowledge Manager

The SIMATIC KNOWLEDGE MANAGER[3] is the result of a project performed in cooperation with tecInno GmbH, Kaiserslautern, for Siemens AG, in particular for the customer support group of *Automation & Drives*. The main emphasis here was to develop a tool that would support the hotline of *Siemens Automation & Drives* as well as enable customers (who are technicians dealing with SIMATIC equipment worldwide) to search certain document collections via the WWW.

Similarly to the FALLQ project, a number of document types, such as FAQs and Download Information, had been available a priori which could be used to construct a case base.

Keyword Layer: derived by analyzing a document collection, using statistics about term frequencies and such, plus linguistic tools and tables to include various word forms, different to FALLQ due to German documents

Phrase Layer: filled by manually analyzing product catalogues available at Siemens

[3] http://www.ad.siemens.de:8080/skm

Thesaurus Layer: mainly derived from analyzing German composite nouns

Glossary Layer: partly derived from product catalogues which also include a clustering of products, partly built by Siemens staff

Feature Value Layer: obtained from product catalogues and additional databases containing product descriptions, unique product numbers and such

Domain Structure Layer: built in discussion with Siemens staff

Information Extraction Layer: IE module directly built on top of feature value layer

Discussion

After having described how knowledge may be used in a Textual CBR application that is similar to the described hotline scenario, two major questions need to be discussed: Firstly, does the knowledge acquisition effort pay off? Secondly, what are related areas and techniques that can be used for the purpose of knowledge acquisition? As an answer to the first question, we will present the results of an preliminary evaluation performed. After that, we will discuss related work.

Evaluation

Obviously, for evaluating the performance of a Textual CBR system, measures similar to those known from the IR community should be used. However, there are some crucial differences with respect to the underlying assumptions:

Firstly, measures like *precision* and *recall* assume that for a set of queries relevance judgments are known. In our projects, this appeared to be a major drawback as these projects were performed in highly specialized domains where relevance judgment is possible only by a domain expert. What's more, one can easily imagine that it is hard, if not impossible, to get these experts perform a task which is mainly interesting from an academic point of view.

Secondly, the original *recall* measures the percentage of the retrieved relevant documents with respect to all relevant ones. In the hotline scenario, however, the goal is not to retrieve *all* relevant items but rather to answer a query successfully. Hence, *one* relevant document is sufficient (for a similar discussion see (Burke *et al.* 1997)).

Concerning the second problem, we modified the notion of *recall* such that for a single query *recall* is 1.0 if a relevant document has been retrieved, and 0 otherwise.

The first problem is harder to overcome. To construct a set of queries for evaluation, we utilized the following observation: While relevance judgments can only be given by a domain expert, virtually every one can determine whether two queries have a similar semantics. Consequently, we randomly selected a set of FAQs and changed their question components in several steps, from minor grammatical variations to complete reformulations. In the latter case, we changed as many expressions as possible but kept the names of devices, modules, components etc.

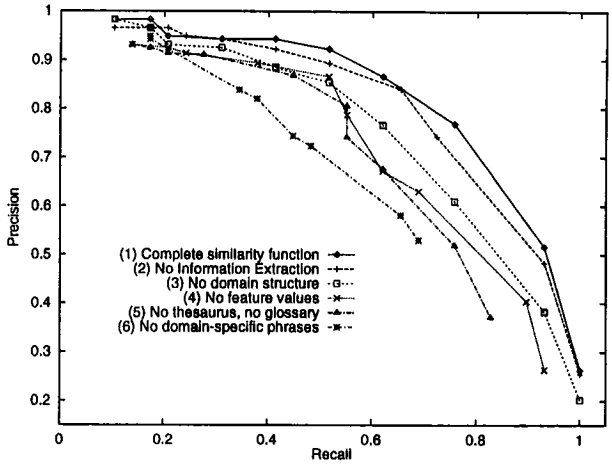

Figure 1: Results of the ablation study

An example of such a reformulation might be:

Original query: *In what modus do I have to shoot the 128 KByte EPROM for the CPU-944?*

Modified query: *How do I have to run the 944 CPU for installing a 128 KB EPROM?*

As this example shows, the semantics of the query may change slightly. But in any case, the answer to the original query will still answer the modified one.

When applying this procedure, one knows which FAQs are relevant for each of the modified queries. In most situations it was just the original document, but in about 25% a second relevant FAQ could be found as a highly similar FAQ occurred in a different context again.

Ablation Study: Recall that the similarity measure used to compare documents makes use of several knowledge layers as described above. To evaluate the contribution of each layer, we performed an ablation study by subsequently eliminating higher level layers.

Figure 1 shows the results obtained from evaluating out Textual CBR approach within the SIMATIC KNOWLEDGE MANAGER domain. As the *precision–recall* curve shows, the performance of the system degrades if less knowledge is integrated into the similarity measure. Also, the differences between the various curves are as expected:

- If no *Information Extraction* is performed, the system's performance degrades only slightly (1 vs. 2).
- If the domain structure is not used to filter out similar documents from completely other areas[4], the same

[4]Note that in the SIMATIC application documents may describe related observations – but as they apply to different components, such a document is of no use if it describes a behavior of a different component.

recall values are achieved at a lower level of *precision* (1 vs. 3).

- Another step can be observed if structural information contained in the documents is not considered (3 vs. 4).
- If similarity of terms is not considered, the performance further degrades (4 vs. 5).
- Finally, a major loss in performance results if the domain-specific phrases are removed from the set of IEs (5 vs. 6).

Related Work

FAQFinder applies CBR in combination with other techniques to document management (Burke *et al.* 1997). In particular, FAQFinder's goal is to answer questions by retrieving similar FAQs from USENET news groups FAQs. FAQFinder uses WORDNET (Miller 1995) to base its reasoning on a semantic knowledge base. It differs from our approach as it does not focus on a specific domain. Instead, it applies a two stage process:

- In the first step, a shallow analysis mainly of the keywords contained in the query is used to infer the most likely news groups related to the request.
- After the user has selected one of the presented news groups, a more sophisticated analysis of the related FAQ file starts to compare the contained FAQs with the user's query.

The approaches discussed in this paper are even more focussed on a specific domain in that the systems have been designed specifically for the particular application. For example, in such technical areas a lot of terms exist that would hardly be represented in WORDNET. Also, a careful knowledge engineering process has been undertaken to employ domain-specific knowledge for similarity assessment. This would not be possible in the scenario of FAQFinder where the semantic base (i.e. WORDNET) is the same for all news group topics.

SPIRE uses a completely different approach for dealing with textual cases (Daniels & Rissland 1997). Based on the observation from the field of IR that people have problems in formulating *good queries* to IR systems, the idea behind SPIRE is to use a combination of CBR and IR technology: A user's request for some information is analyzed by means of a CBR module. As a result, a small number of relevant cases representing text documents are selected and sent to the INQUERY retrieval engine. Consequently, CBR is in fact used as an interface to IR.

Information Extraction is another area closely related to the approaches discussed in this paper (Riloff & Lehnert 1994); in particular, for filling some of the knowledge layers. Constructing the MITA system (Glasgow *et al.* 1998) obviously required highly similar techniques as used during knowledge acquisition for Textual CBR. The major difference to the MITA system is that our objective is not to *classify* documents but to retrieve relevant documents when given a problem description.

Acknowledgments

The authors want to thank all the people who have been involved in the Textual CBR projects discussed in this paper, in particular Hans-Dieter Burkhard, Mirjam Kunze, André Hübner, Thomas Ritz, Alexander Glintschert, and Marlies Gollnick.

References

Burke, R.; Hammond, K.; Kulyukin, V.; Lytinen, S.; Tomuro, N.; and Schoenberg, S. 1997. Question Answering from Frequently Asked Question Files. *AI Magazine* 18(2):57–66.

Daniels, J. J., and Rissland, E. L. 1997. What You Saw Is What You Want: Using Cases to Seed Information. In Leake and Plaza (1997), 325–336.

Glasgow, B.; Mandell, A.; Binney, D.; Ghemri, L.; and Fisher, D. 1998. MITA: An information extraction approach to the analysis of free-form text in life insurance application. *AI Magazine* 19(1):59–71.

Kunze, M., and Hübner, A. 1998. CBR on Semi-structured Documents: The ExperienceBook and the FAllQ Project. In *Proceedings 6th German Workshop on CBR.*

Leake, D. B., and Plaza, E., eds. 1997. *Case-Based Reasoning Research and Development, Proceedings ICCBR-97*, Lecture Notes in Artificial Intelligence, 1266. Springer Verlag.

Lenz, M., and Burkhard, H.-D. 1997. CBR for Document Retrieval - The FALLQ Project. In Leake and Plaza (1997), 84–93.

Lenz, M.; Burkhard, H.-D.; Pirk, P.; Auriol, E.; and Manago, M. 1996. CBR for Diagnosis and Decision Support. *AI Communications* 9(3):138–146.

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

Richter, M. M. 1995. The knowledge contained in similarity measures. Invited Talk at ICCBR-95.

Riloff, E., and Lehnert, W. 1994. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems* 12(3):296–333.

Salton, G., and McGill, M. 1983. *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.