

5. Textual CBR

Mario Lenz, André Hübner, Mirjam Kunze

“All the evidence suggests that for end-user searching, the indexing language should be natural language, rather than controlled language oriented.”
Lewis and Sparck Jones (1996)

In this chapter, we will explore a fairly new direction of research in the case-based reasoning community, namely the handling of textual documents. We will explain first why the ability to deal with natural language texts is crucial. We will then discuss more traditional methods for these tasks. Following this, we present the approach of Textual CBR and, in particular, the CBR-ANSWERS project.

5.1 Introduction

In recent years, case-based reasoning researchers have started to address tasks that have traditionally been coped with by the Information Retrieval community, namely the handling of textual documents.

When considering the roots of CBR, this development is not surprising at all: As discussed already in Chapter 1, the fundamental idea of this problem solving paradigm is to collect *experiences* from earlier problem solving episodes and to explicitly reuse these for dealing with new tasks. In real life, many of the most valuable experiences are stored as textual documents, such as:

- reports by physicians;
- documentations and manuals of technical equipment;
- Frequently Asked Question (FAQ) collections;
- informal notes and comments on specific functions and observed behavior.

Consequently, it is only natural to address the issue of textual documents from a CBR perspective.

In this chapter, we will demonstrate how CBR systems can be designed which are able to handle textual documents. We will discuss more traditional approaches to text retrieval, describe in detail some of the existing projects and what problems still need to be solved. Before this, an example application scenario shall clarify the goals of a Textual CBR system.

5.2 A Typical Application Area: Hotline Support

As pointed out earlier in Chapters 3 and 4, an efficient customer support is becoming more and more important in today's business world. Consequently, many companies establish so-called *help desks* and *hotlines* in addition to tools shipped together with the products (cf. Section 3.6). In case of a problem, the customer may contact these hotlines and will be given additional support. Generally, applications such as help desks and hotlines 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 and a *general world understanding* is not required.
- Because of the limitation to a specific domain, a number of highly specific terms will be used, such as names of components, functions, and modules.
- 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.
- 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.

Consequently, there is an urgent need for systems which can support the hotline staff in retrieving relevant documents in a specific problem context. For this, techniques are required which support the search for documents given a user's question; i.e., on the one hand they have to handle natural language texts, on the other hand these techniques must be flexible enough to cope with paraphrased documents which have essentially the same semantics (Lenz et al. 1998).

5.3 Basic Ideas of Information Retrieval

When trying to recall relevant textual documents in a problem situation, the technique that is traditionally applied is Information Retrieval (IR) (Rijsbergen 1979; Salton and McGill 1983). Despite the name, however, IR systems do not really retrieve information in the sense that they deliver facts satisfying an information need. Rather, they search for documents which are somehow related to the information need as expressed in a query:

“An information retrieval system does not inform ... the user on the subject of his inquiry. It merely informs on the existence ... of documents relating to his request.” (Rijsbergen 1979)

Consequently, a better name for this area would be *Document Retrieval*. However, *Information Retrieval* has now become a widely accepted term.

5.3.1 Fundamental Models of IR

Quite a number of successful IR systems now exist on the market, including search engines for the WWW. Most of these systems are based on either the *Vector Space Model*, the *Probabilistic Model*, or the *Inference Network Model* of IR. To ease the understanding of the differences compared to Textual CBR, we will sketch these models in the following. For a more detailed comparison see Lenz (1998).

In the *Vector Space Model* (Salton et al. 1975), documents are encoded by means of a very large vector, the elements of which represent the weight of a specific term of the index vocabulary for that document:

$$D_i = [d_{i1}, \dots, d_{in}]$$

The weight d_{ij} of the j th index term is calculated on the basis of the frequency of that term in the considered document D_i as well as in the entire document collection. Once documents are represented this way, similarity of two documents may be assessed by comparing the corresponding vectors, e.g., by calculating the cosine measure.

In the *Probabilistic Model* (Fuhr 1989), too, documents are represented by means of term vectors:

$$D_i = [d_{i1}, \dots, d_{in}]$$

However, D_i is now a binary vector, i.e. $d_{ij} = 0$ or $d_{ij} = 1$ depending on the presence of the j th term d_j in the D_i . In this model, the probability that a certain document D_i is relevant to a given request:

$$P(\text{relevant}|D_i) \tag{5.1}$$

is calculated, again, from the observed term frequencies where Bayes' Theorem is used to reformulate Equation (5.1).

Finally, in the *Inference Network* model (Turtle and Croft 1990; Turtle 1991), documents are represented by certain nodes in a graph which also contains nodes for index terms, query terms, etc. These nodes are connected via arcs which may be weighted according to observed probabilities in a document collection. Given a query representing an information need, the query network is built characterizing the dependence of that query on the document collection, i.e., a ranking may be performed based on the information encoded in the network.

5.3.2 Advantages of IR Models

A major advantage of these IR models is that they are domain-independent. All the information required for building such a system may be obtained from a given document collection, including the weighting of index terms. Thus, it is appropriate to consider IR systems as tools which may be easily applied to virtually any domain in which textual documents have to be handled.

Also, IR systems may handle large document collections, i.e., the number of documents that can be searched efficiently is, by far, larger than what has traditionally been the number of cases in a CBR system.

Finally, the IR community provided a number of valuable contributions concerning the evaluation of their systems. Using these evaluation methods, it is possible to compare various systems and estimate various performance parameters.

5.3.3 Limitations of IR Models

Assumption of Independence. All of the above IR models assume that the various index terms used for representing documents are, in some sense, independent of each other. For the *Probabilistic Model*, for example, an explicit assumption is that index terms are *stochastically independent* (Fuhr 1989).

A similar assumption underlies the *Vector Space Model*; if two documents contain disjoint sets of index terms, then the vectors representing these documents are orthogonal to each other and any reasonable measure should indicate that they have zero similarity to each other. If, however, two or more terms are synonymous to each other, then the two documents should have non-zero similarity. In fact, the assumption of independence of the two index terms is violated then.

Such problems often occur because in many situations terms are related to each other and during search one wants to consider these relationships.

Statistics Rather than Knowledge. As sketched above, all major IR models heavily make use of statistics, counting of index terms, etc.:

1. The index vocabulary is determined based on the given document collection and applying some statistics and heuristics. For example, terms that occur too often are considered to be useless — as are terms that occur too seldom.
2. The weight of index terms is again based on counting frequencies in a given document collection. Consequently, the similarity of documents is based on these frequencies, too.

In terms of the knowledge container model introduced in Section 1.3.8, this means that, in fact, only the document collection may contain meaningful knowledge — while both, index vocabulary and similarity measure are fixed

for a given document set. Consequently, a number of problems arise when some kind of knowledge has to be integrated into an IR system to improve the problem solving behavior. In practice, this knowledge may contain both;

- descriptions of terms that are particularly relevant to a certain domain,
- specifications of relationships between different terms.

When attempting to work with textual documents without using any specific domain knowledge, one often has to struggle with the following two problems:

Ambiguity Problem: Although the sets of keywords in two compared documents may be similar, the actual meaning of both documents may differ significantly.

Paraphrase Problem: With natural language, the same meaning may be expressed using completely different expressions.

A further shortcoming of traditional IR methods is that they do not support the consideration of information that is represented in non-textual form, e.g. additional structured information, diagrams, etc.

5.3.4 An Even Simpler Approach: n -gram Matching

An alternative approach to the classical IR models has been implemented in some commercially available help desk systems as, for example, CBR EXPRESS from Inference Corp. Instead of representing documents as sets of index terms, CBR EXPRESS, uses an even simpler matching based on n -grams of common characters for comparing documents. More precisely, the text contained in a document is cut into sequences of n subsequent letters (most often $n = 3$) and the set of all the sequences is used as a representation of the original document.

Example 5.3.1. When using 3-gram matching, the textual description

The new printer does not work.

is represented in form of the following set of 3-grams:

{the,new,pri,rin,int,nte,ter,doe,oes,not,wor,ork}

Compared to the above models of IR, this is firstly less computationally expensive and secondly appears to be very robust against minor changes in the text as, for example, grammatical variations and misspellings.

As with the IR models, this kind of document matching does not permit the integration of additional knowledge sources, such as domain-specific thesauri, glossaries, etc. This is obvious from the fact that in the representation of a document not even the actual words occurring in that document are directly recognizable.

In a tool like CBR EXPRESS, however, such an approach is sufficient as the analysis of textual descriptions is intended to only give a first clue about

which cases to focus on. A further refinement is then made by answering questions and evaluating the cases with respect to the obtained answers. As far as the underlying principles are concerned, this second step is equivalent to querying a decision tree as explained in Section 3.2, i.e., for this phase, a properly structured case memory is necessary which requires a careful knowledge engineering process.

In particular, a *case authoring* process must be performed during which the cases as they occur in the problem domain are encoded by means of the formalisms provided by the system. In particular, for larger applications, this case authoring process requires a lot of man power and skill. In contrast, the approach of Textual CBR, that will be explored in the following, directly utilizes the cases; i.e., the textual documents as they are available in the problem domain.

5.4 Textual CBR

As discussed above, reuse of information contained in documents, such as manuals and FAQ collections seems to be a straightforward task for a CBR system. However, until recently, CBR research mainly focussed on areas where a more structured case representation is applicable, i.e., where cases describe certain situations and events in some predefined format.

For building a CBR system handling textual documents, we have to solve a number of tasks as they occur in other application areas too, such as case representation, knowledge acquisition, etc. However, as we deal with natural language texts here, the emphasis is a bit different. Before going into the details about these issues, we will explain why we think it is worthwhile to spend the additional effort required for Textual CBR.

5.4.1 The Knowledge of a CBR System

A crucial difference to the above sketched IR models is that CBR is a knowledge-based technique – hence any CBR system explicitly allows (*even: requires*) the integration of semantic knowledge. Consequently, during knowledge acquisition, a domain model has to be build which describes the essential properties of the domain, the entities which one is talking about, and their relationships (most importantly, similarity).

In terms of the knowledge container model (cf. Section 1.3.8), 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 chapter, we will assume that (a) documents are available which can serve as the basis for a case base and (d) adaptation is of limited use only. Consequently, the most important knowledge containers are (b) the index vocabulary describing which term are used in the domain and (c) the similarity measure relating the various terms as well as queries and documents to each other. In particular, a *semantic* similarity measure of CBR systems will contain *additional* knowledge as a supplement to the other knowledge containers. This is possible because a Textual CBR system is built for a specific domain only:

1. By performing a careful knowledge acquisition, features important in that domain can be identified (e.g., based on keywords, terms, expressions etc.) and thus a *term dictionary* can be constructed which is not limited to statistic evaluations. Often, these features may cover more than a single keyword; an example might be the expression *Customer Administration Module* which represents a certain component in the FALLQ project (see below).
2. Additional knowledge may be represented by means of the *structure* of documents; for Textual CBR, a document is not merely a single text but may consist of several components. A FAQ, for example, would naturally contain (at least) two separate parts, a question and an answer.
3. Due to the limitation to a specific domain, the *Ambiguity Problem* is avoided to a great extend.
4. By carefully analyzing the domain under consideration, a similarity measure can be constructed which goes beyond statistical term weighting, e.g., by considering a domain-specific thesaurus, ontologies underlying the domain, etc.

With respect to the latter point, it is important to recognize the difference to IR systems; though some of these also use thesauri, it is much easier if domain-specific knowledge can be utilized. Also, terms which describe objects (such as names of devices, processes, etc.) do not normally occur in general-purpose thesauri but are available in a specific context.

It is obvious that the major advantage of Textual CBR is the ability to make use of domain-specific expertise in order to go beyond pure keyword matching and thus improve the problem solving behavior. However, this benefit is not for free; rather, a well-designed knowledge acquisition process is required in order to encode existing knowledge and make it available to the CBR system. While the non-textual information in these environments has to be dealt with as in other Artificial Intelligence approaches, handling of textual data is novel and, hence, requires additional effort.

5.4.2 Knowledge Acquisition for Textual CBR

In order to obtain cases to reason about from a given document collection, it is required to *extract* cases from the given text documents. Obviously, a

first step towards this is to use keywords specific to a domain for building an appropriate index vocabulary. However, this allows only a rough estimation of the content of documents. If one wants to obtain a more meaningful representation of texts, Natural Language Processing (NLP) techniques are promising candidates.

In our projects, we experimented with a couple of NLP techniques to obtain a more sophisticated representation of textual documents. Unfortunately, most of the available NLP tools have a number of shortcomings when it comes to real-world texts:

- They are computationally expensive, i.e., the performance of the systems on larger text collections does not permit their integration into other systems.
- They only work with small dictionaries; even worse, they are not robust when unknown terms are encountered.

Besides these shortcomings, we experienced that sophisticated NLP is of limited use because in many situations, documents are in the form of short notes, remarks, etc. instead of properly structured sentences with correct grammar.

Consequently, we concentrated on so-called *shallow* NLP techniques, such as *Part-Of-Speech* tagging. These are both, efficient and robust as they do not attempt to build a highly structured representation of texts but merely *tag* each word with its probable class (i.e., whether it is a noun, a verb, etc.) and additional information such as the word stem.

For this purpose, we used the TREETAGGER¹ tool developed at the Institute of Natural Language Processing (IMS) at Stuttgart University and analyzed the result of tagging the document collection. In particular, the objective was to obtain both, lists of relevant keywords as well as stemming information.

More precisely, we applied the following procedure:

1. Let the NLP tool *tag* an example document collection.
2. *Normalize* the texts in the document collection by replacing particular words by their corresponding stems.
3. Obtain index terms by grouping together words with common stems.
4. Automatically classify the index terms on the documents as useful or useless according to the following heuristics:
 - a) Mark a term as *useless* if it is a determiner, an auxiliary or modal verb, a preposition, a pronomen, etc.
 - b) Mark a term as *useful* if it is an adverb, an adjective, a full verb, or a noun which is contained in the dictionary of the NLP tool.
 - c) Otherwise, mark the term as *potentially useful*.
5. For the *useful* and *potentially useful* terms, determine how often these are present in the document collection and discard those which appear too seldom to be of interest (i.e. once or twice only).

¹ <http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html>

6. Investigate the *useful* and *potentially useful* terms manually; in particular for those terms not present in the dictionary determine whether these are special terms of the domain or whether they are spelling errors, etc.

While the last step requires manual effort, this is limited to a relatively small set of terms. In the domain of SIMATIC customer support, for example, we had to manually investigate an index term list of less than 2,000 entries, which should be manageable in a few hours (cf. Section 5.7.2).

Concerning this procedure, the following issues are highly important:

1. Index term selection is not performed solely on the basis of statistics, keyword counting, etc. Rather, semantic information is used.
2. The above method only describes how index vocabulary can be obtained from a given document collection. It is, of course, possible to also utilize additional sources such as manuals, glossaries, etc. in which domain-specific terms are defined.
3. Every index term is not only classified according to its usefulness but also assigned a special tag indicating whether it represents a domain-specific or a general purpose term. In this way, development of index vocabulary for various domains is incremental in that the general purpose terms obtained from an analysis of one domain can be utilized without any further effort for another domain.

Information Extraction. While major parts of the contents of documents are given in plain text, some more structured information may be included in the documents too. In particular in technical domains, the text often contains expressions such as “220 Volt” or “CPU 991-999”. Obviously, a better representation for this kind of information is to use attribute-value pairs which, however, have to be obtained from the texts automatically.

For this, we applied techniques which originate in the area of *Information Extraction* (cf. Riloff and Lehnert (1994) for an overview). More precisely, we defined a number of *triggers* which indicate that a piece of text might contain an expression suitable for an attribute-value based representation.

As an example consider a phrase like

“*The CPU 999 is constructed to work with 12 Volt instead of 8.*”

Obviously, both *CPU 999* and *12 Volt* should be considered as features of the corresponding case. To achieve this, physical units (such as *Volt*) as well as other names (such as *CPU*) have been defined as triggers. Additional information is provided for these triggers as constraints, for example that voltage should have a numerical value which is usually written in front of the *Volt* string. Once such a trigger is observed, a more complex sentence analysis starts, the objective of which is to *extract* the information associated with that trigger. As a result of this Information Extraction process, a case does not only contain the index terms present in the corresponding documents but also some more structured information in the form of attribute-value pairs.

Machine-Readable Thesauri. Obviously, similarity of documents should indicate similarity of contents of these documents. Because of the *Paraphrase Problem*, however, it is clear that similar contents may be expressed in various ways. Consequently, a thesaurus relating synonymous terms to each other is highly useful.

For English, WORDNET² is such a tool. It provides information about similar words, antonyms, etc., based on a solid linguistic ground. We utilized parts of WORDNET for some aspects of similarity assessment. Of course, such a tool can only cover general purpose terms while domain-specific expressions will not be related to each other. For the latter terms, we manually analyzed documentations, glossaries, etc. and encoded the obtained knowledge in terms of the similarity measure.

For the applications dealing with German texts (as the SIMATIC application described in Section 5.7.2) we had the problem that a tool like WORDNET is not available. Consequently, we had to build our own thesaurus which was, however, limited to the specific domain. To do so, we combined a manual knowledge engineering process with some heuristic procedures which could be performed automatically. For example, we utilized the fact that German texts make extensive use of compound nouns. To make use of this, we automatically derived similarities between words which are part of each other.

5.4.3 Requirements for Textual CBR

In Section 5.3, we mentioned that a major advantage of traditional IR techniques is that they can be applied to virtually any domain. The only requirement is that a document collection is available. Thus, all the information which is necessary to build a running IR system can be computed on the basis of this document collection.

Unfortunately, this is not the case for Textual CBR systems. In addition to the aspects of knowledge engineering discussed in Section 5.4.2, some further requirements have to be satisfied. In particular, the CBR-ANSWERS system, which will be described in detail in Section 5.5, is based on the following explicit assumptions:

Availability of Textual Documents. As with the more traditional IR models, a crucial requirement is that documents are available which can be utilized for building a case base. Note that, in contrast to the systems briefly described in Section 5.3.4, we do not attempt to perform a separate *case authoring* process but rather make direct use of available documents.

Availability of Knowledge. In order to really make use of the knowledge-based approach of Textual CBR, domain-specific knowledge has to be available from which information concerning index vocabulary, similarity assessment, etc. can be obtained. According to our experiences from projects, part

² <http://www.cogsci.princeton.edu/~wn>

of the required knowledge is readily available. For example, companies often maintain databases which contain descriptions of all their products, including names, components, etc. As with other knowledge-based approaches, the expertise required beyond this has to be obtained in a knowledge acquisition process. For this, domain experts have to be available.

Domain-specific Application. The previous requirement implicitly implies that a Textual CBR application is built with a specific domain in mind. As similarity knowledge is used to express relationships between the various entities in the domain, an implemented system will not be easily protable to another problem domain.

Semi-structured Documents. A major advantage of Textual CBR is that it can handle *semi-structured* as well as unstructured documents. In our projects, we even assumed that, although all the major information is contained in the textual description, additional structured components are also available. It is even more beneficial if each particular document has a certain structure. This is the case, for example, with *Frequently Asked Questions*, where each document is, in fact, a question-answer pair. However, in other situations, there may even be more complex document structures.

5.5 The CBR-ANSWERS Project

As a result of our research activities in the area of Textual CBR, we have developed the CBR-ANSWERS system, a tool suitable for handling of knowledge contained in textual documents given the above requirements. In particular, CBR-ANSWERS addresses the hotline scenario sketched in Section 5.2. The system heavily relies on the model of Case Retrieval Nets as described in Section 3.4. In particular, concepts like *Information Entities* (IEs) will again be used here for describing the functionality of the system.

5.5.1 Overall Architecture of CBR-ANSWERS

Figure 5.1 gives an overview of the principle architecture of CBR-ANSWERS. The entire system is implemented in a Client-Server architecture. Using the system consists of two separate phases:

Off-line Preprocessing. During an off-line process, the textual documents are parsed in order to obtain an appropriate representation of the documents. Note, that *parsing* here refers to a much more sophisticated process than usually in IR systems (see below). The result of this process is a properly structured case memory in the form of a Case Retrieval Net which can be utilized by the *Retrieval Server*.

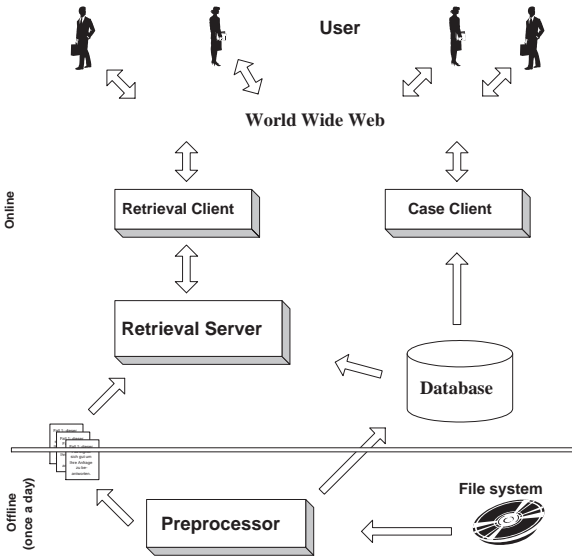


Fig. 5.1. Overall architecture of the CBR-ANSWERS system

Online Retrieval. For online retrieval, the *Retrieval Server* itself is permanently running. Users access the system, e.g., via the World Wide Web, and by submitting a query, the `httpd` server starts a *Retrieval Client* which is basically a CGI script³. This connects via TCP/IP sockets to the *Retrieval Server* to pose the query and receive the results. Finally, the *Retrieval Client* displays these results in an appropriate style (HTML, Java, etc.) to the user.

Besides the components required for the retrieval process itself, CBR-ANSWERS includes a number of additional modules. As an example, Figure 5.1 shows the *Case Client* which is utilized to display the documents which the user requested after having selected from the list of documents resulting from a retrieval process. This additional functionality has been implemented (a) to obtain a more modular structure of the system in which issues of retrieval are separated from displaying documents, and (b) to support situations in which the document collection used during preprocessing is different from the one used for displaying documents. The latter situation might occur, e.g., when CD-ROM catalogue are shipped to users; i.e., when preprocessing is probably based on company-internal documents while the customers should only see a public version of these.

5.5.2 Case Structure and Case Representation

In order to access the information contained in the textual documents, each document is *parsed* to obtain a properly structured case using the given IE

³ Of course, the *Retrieval Client* may also be any other kind of program appropriate for communicating with the *Retrieval Server* and providing a GUI to the user.

dictionary. Note, however, that *parsing* here refers to a much more sophisticated process than usually in IR systems as this includes

- knowledge about the structure of the documents, e.g., that FAQs should be represented as question-answer pairs;
- a mapping from texts to sets of IEs where an IE may be more than just a *keyword* — the expression *Customer Administration Module* mentioned in Section 5.4 is an example;
- shallow NLP techniques (*Part-of-Speech* tagging) to identify basic linguistic structures in the given documents;
- some simple Information Extraction techniques to recognize structured items in the text, such as certain attribute-value pairs.

Another important issue is that an IE is a kind of symbol representing a specific meaning in the application domain. Consequently, a single IE represents a certain keyword or phrase, including grammatical variations, abbreviations, and even foreign language expressions. The set of all IEs may be further distinguished according to specific types, such as:

- keywords, such as *billing*;
- domain-specific expressions, such as *Customer Administration Module*, which would not normally be considered as a single keyword but have a specific meaning;
- special codes, numbers etc. which refer to a particular object in the domain;
- attribute-value pairs, like *Version=1.1*, which can be obtained by an Information Extraction process;
- further attributes required for domain modeling.

An important aspect of this *parsing* process is that it is performed automatically; i.e., no manual case authoring has to be performed.

5.5.3 Similarity and Retrieval

Once the documents have been converted into cases, the query can be handled in the same manner. Then, similarity of queries and cases is computed based on the similarity of the constituent IEs, i.e., the similarity of a case F to a query Q is defined as:

$$SIM(Q, C) = \sum_{e_i \in Q} \sum_{e_j \in C} \text{sim}(e_i, e_j) \quad (5.2)$$

where e_i and e_j are the IEs used to represent the query Q and the case C , respectively. The function sim specifies the *local* similarity of any two IEs.

To get a better understanding of how the various types of IEs contribute to case similarity, it is helpful to sketch how the similarity function of Equation 5.2 is composed according to the types of IEs which really influence each other. In more detail, the overall similarity measure takes into account:

- similarity based on common IEs;
- similarity based on related (similar) IEs;
- similarity based on attribute values as specified in the documents;
- similarity based on attribute values obtained during automatic Information Extraction.

Note that the components of the similarity function become more and more knowledge-intensive in the order they are listed above. In Section 5.5.4, we will show how these components contribute to the overall performance of the system.

The retrieval process directly utilizes the IEs of cases and their similarity relationships as they are encoded by means of a Case Retrieval Net (cf. Section 3.4). In particular, efficiency and flexibility are valuable benefits of this model (Lenz and Burkhard 1996a).

5.5.4 Evaluation

Evaluation Methodology. Despite the users' acceptance of the systems implemented using CBR-ANSWERS (see Section 5.7), a theoretical evaluation is required to clearly show advantages and disadvantages of the approach. For this purpose, measures similar to those known from the IR community should obviously be used. However, there are some crucial differences with respect to the underlying assumptions:

Firstly, measures like *precision* and *recall* generally assume that relevance judgments are known for a set of test queries. 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. Furthermore, 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. 1997b).

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 was harder to overcome. To construct a set of queries for evaluation, we utilized the observation that, while relevance judgments can only be given by a domain expert, virtually anyone 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. 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 shooting a 128 KB EPROM?*

As this example shows, the semantics of the query may change slightly. But 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 percent, a second relevant FAQ could be found as nearly exactly the same FAQ occurred in a different context again. Based on this, we built a set of test queries from the SIMATIC KNOWLEDGE MANAGER domain and used this test set for a number of performance tests. In particular, the ablation study described in the following provides useful insights into the behavior of the system.

Ablation Study. As described in Section 5.5.3, the similarity measure implemented in CBR-ANSWERS generally consists of several components with varying degree of knowledge contained. To determine the contribution of each component, we performed an ablation study by subsequently eliminating higher level components.

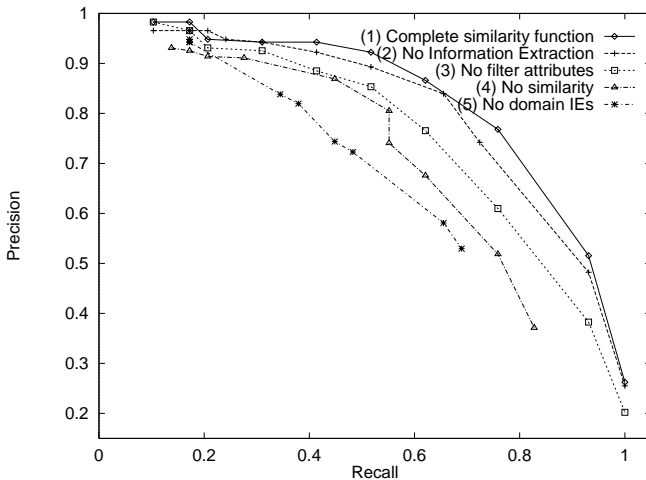


Fig. 5.2. Results of the CBR-ANSWERS ablation study

In Figure 5.2 some of the results obtained from evaluating the CBR-ANSWERS system with the SIMATIC domain (cf. Section 5.7.2) are shown. As the *precision–recall* curves show, the performance of the system degrades if less knowledge is integrated into the similarity measure. More precisely, the performance is degrading if more and more knowledge intensive components are eliminated:

- If no *Information Extraction* is performed, the system's performance degrades only slightly (curve 1 vs. curve 2).
- If the domain structure is not used to filter out similar documents from other areas⁴, the same *recall* values are achieved at a lower level of *precision* (curve 1 vs. curve 3).
- Another step can be observed if structural information contained in the documents is not considered (curve not shown here).
- If the similarity of index terms is not considered and instead only exact matches are taken into account, the performance further degrades (curve 3 vs. curve 4).
- Finally, a major loss in performance results if the domain-specific terms are removed from the set of IEs (curve 4 vs. curve 5).

In fact, the performance of the system is as expected. Another observation which is not shown in Figure 5.2 was that performance further degrades if structural information, such as attribute-value pairs describing release numbers of devices etc., is not represented explicitly but handled as ordinary text.

The CBR-ANSWERS system has been applied for several application projects which we will present briefly in Section 5.7. However, before this, we will discuss some related work in the area of Textual CBR.

5.6 Related Work

As mentioned in the introductory part of this chapter, some other projects addressed topics concerned with Textual CBR in recent years. In the following we discuss some of these.

5.6.1 FAQFINDER

The FAQFINDER project (Burke et al. 1997a; Burke et al. 1997b) also tries to apply CBR technology, in combination with other techniques, to document management. In particular, FAQFINDER's goal is to answer natural language questions by retrieving these from frequently asked questions (FAQ) files from USENET news groups FAQs.

FAQFINDER also uses a thesaurus (namely, WORDNET) to base its reasoning on a semantic knowledge base and assumes that documents are given in a semi-structured format (namely, as questions-answer (QA) pairs).

FAQFINDER differs from our projects in so far as it does not focus on a specific domain. Instead, it applies a two stage process:

⁴ Note that in the SIMATIC KNOWLEDGE MANAGER 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 (cf. Page 134).

- 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 decided on one of the presented news groups (i.e., after s/he selected a topic to focus on), a more sophisticated analysis of the related FAQ file starts to compare the contained QA pairs with the query entered by the user.

In some sense, this interactive scenario relates to the focus on a specific domain in that the user confirms the topic suggested by FAQFinder in the first stage. With the CRB-ANSWERS project, we have focussed still further on a specific domain in that each application has been designed specifically for a particular domain. For example, in technical areas, a lot of terms exist that would hardly be represented in general purpose thesauri, such as 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.

5.6.2 SPIRE

The SPIRE system introduced by Daniels and Rissland (1997) uses a completely different approach for dealing with textual cases. 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:

- Given a user's request, a HYPO-style CBR module is used to analyze this request semantically and select a small number of relevant cases representing text documents.
- The most relevant cases from the first stage are then used to pose a query to the INQUERY retrieval engine.
- After having retrieved relevant documents, cases are used again to extract the most relevant passages from the documents.

Compared to the projects described in this chapter, this is a completely different approach in which CBR is in fact used as an interface to IR. Thus, the *Paraphrase Problem* can be overcome by letting the CBR module suggest alternative phrases. Whether the *Ambiguity Problem* can be solved, too, is still an open question. Furthermore, domain knowledge is limited to the cases suggesting good indices for the IR system. It cannot be used for similarity assessment, etc.

5.6.3 Automatic Index Assignment

Brüninghaus and Ashley (1997) address the handling of textual documents in yet another way than described so far. Here, the objective is to have a set of

more abstract feature descriptions and utilize Machine Learning techniques in order to automatically assign these indices to textual cases.

In particular, the project builds upon the CATO project (Aleven and Ashley 1996), a CBR system in the domain of legal argument. More specifically, CATO uses so-called *factors*; a factor is a kind of fact pattern which describes a certain legal situation on a more abstract level.

Again, this is a completely different approach to the systems presented in this chapter. In particular, CATO assumes that a carefully selected set of factors (or even a factor hierarchy) is available. Nevertheless, ideas from this project might become useful if the functionality currently provided by the systems presented in Section 5.7 has to be extended in such a way that aspects of text classification become useful. For example, one could imagine that FAQs are classified according to whether they contain problem descriptions, requests for extended functionality, questions about available products, and so on. This information surely could provide further information and thus improve the performance of the systems.

5.7 Applications and Projects

Using CBR-ANSWERS as a common base, we performed several projects concerned with the application of CBR technology to document management tasks. Of course, each application had its own characteristics with respect to the particular application scenario, the application domain, available knowledge resources and so on. However, all applications followed more or less the hotline scenario of Section 5.2. In the following, we will sketch some of the specific properties of each application. Note, however, that most information used during application development has been confidential – so we are only able to give principle descriptions on how specific problems can be solved, i.e., without detailed examples.

5.7.1 The FALLQ Project

The FALLQ project has been performed in a cooperation between Humboldt University Berlin and LH Specifications (LHS), a market leader on the telecommunications software market (Lenz and Burkhard 1997a; Lenz and Burkhard 1997b). The primary objective of this project was to develop an information system that can be used for hotline support, e.g., for handling *Frequently Asked Questions*. As a result of the developed prototype, however, LHS realized that they may well utilize the same technology for maintaining various kinds of information within the company.

The Application Area. LHS is developing and selling a customer care and billing system for providers of telecommunications services. By spring 1997,

this system had more than thirty projects and installations world-wide. Because of the dynamics of the telecommunications market, the complexity of the application software as well as the number of installations are steadily growing. To cope with this development, there is an urgent need for an information system which helps:

- keeping track of customers' enquiries;
- finding answers to problems that have already been solved before, either for another customer or by different staff members;
- maintaining information useful for internal purposes, e.g., descriptions of defects which might be useful for the system developers.

Within the company, information and knowledge are stored in many different places using various types of documents, such as:

- descriptions of *defects* and problems during the development of modules;
- specifications of modules summarizing the functionality provided;
- *Frequently Asked Questions*-like documents;
- documentations delivered with the products, training material, etc.

Last but not least, a wealth of knowledge is kept by the experienced staff members which have been developing the software for several years. Because of the rapid growth of the market, however, more and more new staff are employed who lack this historical knowledge.

Current State. As the application directly fits the hotline scenario, a version of CBR-ANSWERS, namely the FALLQ system, could be installed at LHS. FALLQ is specific in so far as it is integrated into the environment at LHS, and as knowledge specific to the application domain has been added to the system. By spring 1998, FALLQ manages approximately 45,000 documents of various types and is accessed by about 20 users a day. Currently, an extension of the system, MERLIN++, is being developed which will support more advanced functionalities and will be made accessible to a wider group of customers.

5.7.2 The SIMATIC Project

CBR-ANSWERS also serves as the basic model for a project being performed in cooperation with Siemens AG, in particular with the customer support group of *Automation & Drives*. Again the objective is similar to the hotline scenario described in Section 5.2, i.e., the goal is to answer questions posed by customers by reusing knowledge contained in FAQs and other types of documents.

In fact, when facing some problem customers of Siemens may already browse these documents which are provided at a particular web site⁵. CBR-ANSWERS provides a means of intelligently searching and analyzing these documents rather than browsing numerous directories.

⁵ www.ad.siemens.de/support/html_00

The SIMATIC Knowledge Manager. An objective when developing the SIMATIC KNOWLEDGE MANAGER is that the hotline staff should be contacted only if a user's question can not be answered by showing them specific documents. This is related to the *call avoidance* strategy discussed in Section 4.6. More precisely, the process model of the SIMATIC KNOWLEDGE MANAGER (cf. Figure 5.3) is such that the customers use a specific WWW site to describe their problems in terms of a textual query, some information on the particular domain they are requesting help for, and possibly information on the devices involved. Given this, an available document collection is searched and the best matching documents are presented to the user, assuming that highly similar documents will very likely be helpful in answering the users' requests. Only if this does not yield a solution to the problem is the hotline staff contacted and asked for help. Finally, if the request is answered, this question-answer pair is added to the document collection and, thus, the SIMATIC KNOWLEDGE MANAGER automatically acquires new knowledge.

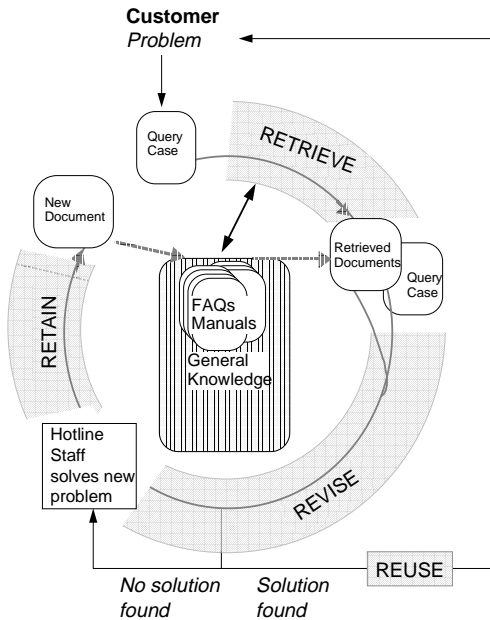


Fig. 5.3. Process model of the SIMATIC KNOWLEDGE MANAGER based on the R^4 model

The SIMATIC KNOWLEDGE MANAGER directly utilizes a number of document types which are available at Siemens. These include: the well-known FAQs, *Informal Notes* on recently observed problems as a simple textual description of these problems, and *User Information Notes*. While all three document types basically contain the same information, there is a difference in reliability. As the name indicates, *Informal Notes* are just draft notes which may give hints but are not trustworthy. On the other hand, FAQs express well-established knowledge resulting from frequent usage. *User Information Notes* lie somewhere in between these two extremes.

A key property distinguishing the SIMATIC KNOWLEDGE MANAGER from the FALLQ project is that several languages have to be supported by the system. In the first stage, a focus is set on German rather than English texts. This implies additional effort because German grammar is much more complicated and even more irregular than English. Later, English and other languages will be supported, too.

The approach of CBR-ANSWERS has the major advantage that textual documents are converted to cases represented by sets of IEs. These IEs are, in fact, a kind of *symbols* having a certain meaning in the real world. Consequently, it is possible to reuse exactly the same symbols including their similarity relationships for another language⁶. The only thing required is a description of how these symbols can be derived given the texts, that is a language-specific *parsing* method. Basically this should utilize a translation of the general purpose terms while specific terms, such as names of functions, devices, and modules, remain unchanged.

Current State. As a result of the project, a prototypical system was implemented which has been tested and evaluated by the SIMATIC hotline staff in a 3 weeks testing period. As with FALLQ, this system was intended to prove the applicability of the Textual CBR approach for the SIMATIC hotline. In particular, the system only worked with a subset of all available documents (about 500 only) and has not been integrated into the call tracking system running at the SIMATIC hotline. Nevertheless, with respect to the effort required to answer customer enquiries, savings of 10% to 30% have been estimated during the test period.

Figure 5.4 shows a snapshot of this application. Currently, the system is being extended in two directions:

- The SIMATIC KNOWLEDGE MANAGER scenario is being implemented for external customers. Besides the aspects of Textual CBR, issues of security etc. have to be considered for this.
- For *in-house* usage, the hotline application will be extended to cover the entire SIMATIC domain and will be integrated into the running call tracking system.

⁶ Remember that CBR-ANSWERS is operating in a very specific domain only and that a general world understanding is not necessary.

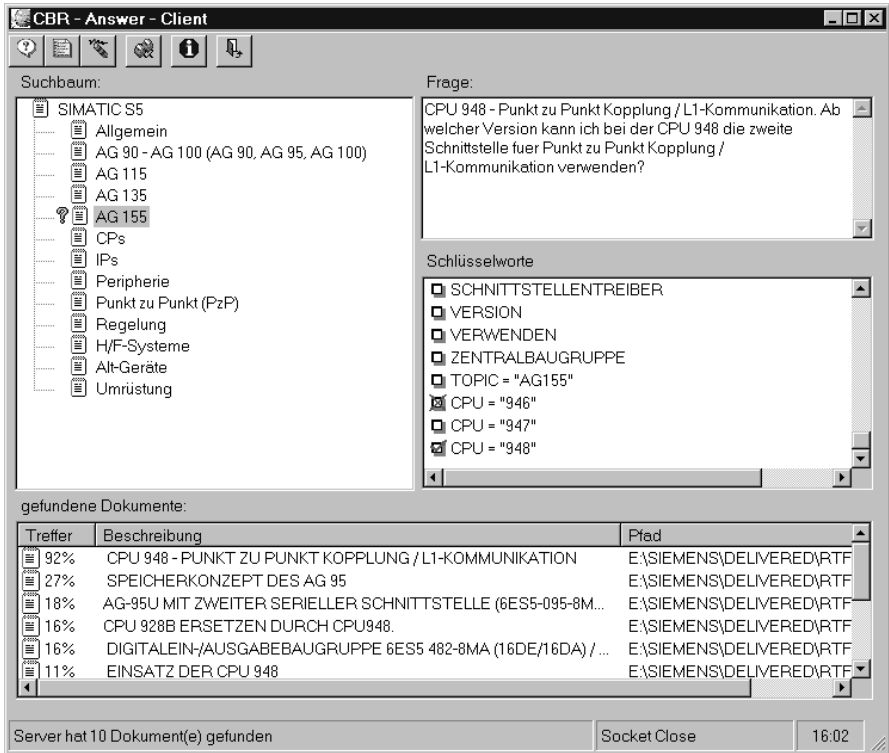


Fig. 5.4. Snapshot of the SIMATIC application CBR working with German documents

- For external usage, a CD-ROM will be produced which is then shipped regularly to customers. Using a search engine based on CBR-ANSWERS, customers can browse this CD-ROM when facing a problem. Thus, only when this problem cannot be solved in this way, they will have to contact the SIMATIC hotline.

5.7.3 The EXPERIENCEBOOK Project

The EXPERIENCEBOOK (Kunze 1997) project differs from the above described applications in so far as it does not reside directly in an industrial setting. Rather, the objective was to provide the system administrators of the Department of Computer Science of Humboldt University with a tool for knowledge management.

System administrators need very detailed knowledge on a wide area of hard- and software systems. This is partly due to the very heterogeneous equipment used in our department. Even worse, the systems are changing fast and thus knowledge is changing rapidly.

Another specific property of the EXPERIENCEBOOK project is that the experts themselves are the users of the system; i.e., they query the system when facing problems and they provide some part of the input to the system by inserting new documents. Thus, the EXPERIENCEBOOK serves as an *external memory* as discussed in Section 3.3. That is, it provides content-oriented search for specific documents. Furthermore, the EXPERIENCEBOOK allows for the exchange of information and knowledge between the system administrators and, thus, helps maintaining a *Corporate Know How*.

To equip the system with a certain basic amount of knowledge, external knowledge sources such as manuals, newsgroup postings, etc. have been used to build an initial case base. For details, the reader is referred to the work described by Kunze (1997).

To make the manually inserted cases more useful, additional information may be added such as, for example, the author of a case, the specific reason of a problem, or the steps required to solve a problem. These additional sections are not considered during retrieval but may provide uaseful insights when reading the retrieved cases.

For historical reasons, the case base of the EXPERIENCEBOOK mainly consists of English case documents completed by some German ones. The preferred language for queries and new cases is English.

5.8 Summary

In this chapter, we have presented major results from the area of Textual CBR, a particular branch of research which deals with handling of textual documents using CBR technology. We have discussed traditional methods of Information Retrieval and why CBR can be a more powerful technique under certain conditions. Finally, we have presented an overview of the CBR-ANSWERS project and sketched some of the implemented systems which are now running at industrial partners.