Textual CBR
Case Studies of Projects Performed

Mirjam Kunze and André Hübner*

Abstract

In this article, we present the Textual CBR approach in three different application areas: The EXPERIENCE-BOOK supports technical diagnosis in the field of system administration. In the FALLQ project we use our CBR system to support the hotline staff in an industrial setting. The third project, the SIMATIC KNOWLEDGE MANAGER, represents an automatic hotline at the *Automation & Drives* department of Siemens.

The objective of these systems is to manage knowledge stored in semi-structured documents. The internal case memory is implemented as a Case Retrieval Net. This allows the handling of large case bases with an efficient retrieval process. In order to provide multi user access we chose a client server model combined with a web interface.

Introduction

Within the field of CBR, a major area of research has been the development of systems providing knowledge in well structured domains. However, in many domains the problem descriptions are given as text documents in natural language. Usually, it is quite time-consuming to transform them into a structured case format, and some time it is even impossible to do so. It would be useful to have a system that can handle less structured source documents without lots of effort.

In this paper, we discuss three case-based systems. At first we give a short description of the application areas then expose the settings for the systems. The third section contains an explanation of the system architecture, the used case memory (Case Retrieval Nets) and its retrieval functionality. In the last section, we state some open issues.

Application Areas of the Projects

The first project (*ExperienceBook*) resides in the area of supporting system administration. System administrators need very detailed knowledge on a wide area. The knowledge is changing fast and new knowledge is added all the time. Their work is time crucial for a lot of people. Usually a system administrator manages different types of devices and software, which are located in different places within one building. Hence, our supporting tool has to meet the following requirements:

- it should be usable as a manual
- it should remember knowledge obtained by the system administrator
- it should be a black board to exchange knowledge between system administrators
- it should be accesible from all (or at least from most) computers
- it should provide multi-user access
- it should be easy to use

To build the case base for the system we can use manuals, provided by the producer of the soft- and hardware, installation instructions, newsgroup postings and frequently asked question documents. System administrators should be able to add new knowledge from their own experiences.

The second project (*FAllQ*) has an industrial setting. LHS is the leading developer of customer care and billing software for cellular telephone service providers. This software-bundle is very complex and exists in different versions. LHS has more than 30 customers worldwide, and the software will be adapted to fit the different requirements of each customer. The number of customers is growing considerably. Very important is a fast reacting support, which solves problems and bugs in a short time. Therefore, the support staff should be able to access the knowledge of the company easily. They have to find answers to problems that have already been solved before, either for another customer or by different staff members. Within the company, this information is stored in documents of various formats, e.g.:

- descriptions of bugs and problems together with the solutions are stored in a type of document called *defect*;
- other documents containing ***functional specifications*** of parts of the software;

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

- *Frequently Asked Question*-like documents;
- manuals and documentations that are delivered with the product.

LHS is a rapidly growing company, and new staff is hired all the time. New employees lack the experience of the senior ones. Their results depend on the knowledge obtained from documents. Currently, there are more than 40,000 *defect* documents, and thousands are added each year. So, a sytem is required which manages large document databases and efficiently finds relevant information.

The *Simatic Knowledge Manager* (SKM) is used at the customer support department at Siemens *Automation & Drives*.

In the business area of industrial automation systems Siemens sells a variety of products, e.g. hardware components like CPUs and software systems. Most of these products exist in different versions.

The customer support department already maintains a web server, where the customers can find information about the products. There are also around 2,000 FAQs which describe solutions of problems for different products. All documents exist in different languages, e.g. German, English, Italian, to support customers from different countries. Due to the vast amount of documents, most customers give up on looking through the documents and call the hotline. A tool is required that supports the customer to retrieve documents that are important for his problem. This tool should be integrated into the hotline process. It should also be able to support different languages.

Textual CBR in Practice

When applying Textual CBR in practice, the following issues seem to be the most important:

- The system must be easy to use.
- The result must be presented fast after asking the query.
- The system must be accessible from different kinds of computers. In the case of SKM it must even be accessible from all over the world.
- If the case base doesn't contain satisfying entries to a query, the user should be informed.
- The retrieval mechanism should be transparent for the user. For instance, he should be able to recognize what happens if he adds an attribute to a query.
- The users of the system have to be considered when deciding how many possibilities an user will have to change parameters and options of the retrieval.
- Once the system is working the manual effort for keeping the knowledge up-to-date must be very low.
- Most of the maintenance should be done automatically

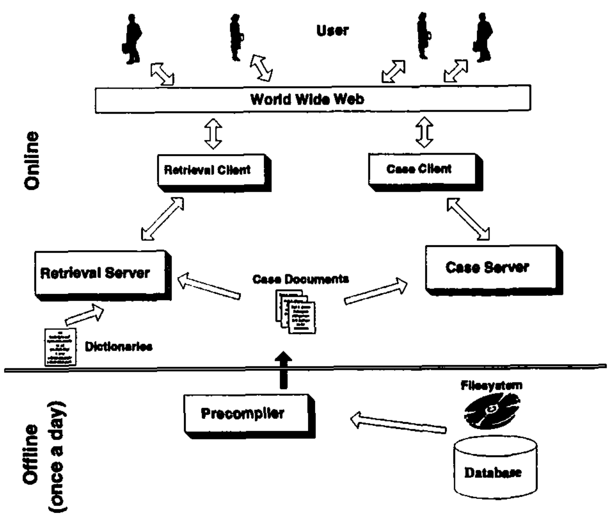

Figure 1: Overview of the system

- The system must be integrated into an existing environment (e.g. call centers).

Design Overview

We chose a client server architecture and implemented a graphical user interface with HTML pages that are displayed by a WWW browser to make the system accessible.

Figure 1 shows the major components of the implementation of the systems:

In an offline process the source documents are scanned and converted into a consistent case structure. The resulting case documents are semi-structured documents containing a unique case number, several passages in natural language and some attributes.

The *SKM*, for instance, uses the following case format:

Case = ''[CASE_NUMBER]'' '\n'
case_number '\n'.
''[TITLE]'' '\n'
text_of_title '\n'.
''[RETRIEVAL_ATTRIBUTES]'' '\n'
[AV_Pair { '\n' AV_Pair }] '\n'.
''[INFO_ATTRIBUTES]'' '\n'
[AV_Pair { '\n' AV_Pair }] '\n'.
''[DESCRIPTION]'' '\n'
text_of_symptoms '\n'.

AV_Pair = *attribute_name* '' ='' ' '' ' *attribute_value* ' '' '.

Source documents are obtained from a filesystem or a database and can occure in a variety of formats, e.g. html, plain text. The *preprocessor* also uses simple information extraction methods to determine attribute value pairs.

The *retrieval server* contains the internal case memory and the retrieval functionality.

The *case server* is able to receive case numbers and to return the text of the according cases. This server can be replaced by a simple database management system.

The *dictionaries* contain general and domain specific knowledge.

At the ExperienceBook there also exists a *retain server*, which helps to expand the case base. A user can send a new case, which is put into a file. Later it is manually prepared for a real update of the case base.

The clients (CGI-scripts) are needed to acces the servers via the world wide web. They are also used to build up the HTML-pages.

So the system is very easy to handle. The user only has to fill HTML forms and can navigate through the result pages with a few clicks.

The Retrieval Server

The heart of our implementation is the retrieval server. With the help of several dictionaries, it builds an internal case memory from the given case documents and then provides the retrieval process for incoming client requests.

For providing an efficient retrieval process we chose as internal case memory the model of *Case Retrieval Nets* (CRNs) (Lenz & Burkhard 1996).

In this model, cases are represented as a part of a graph. This graph is a net with the following nodes and weighted edges:

- nodes for the identification of cases (case descriptors);
- nodes for the information entities;
- edges to a case descriptor from all the IEs specifying this case (the weight of them is defined by the relevance of one IE for a case);
- edges between IE nodes expressing a similarity value by a weight.

So, the semi-structured case documents have to be represented by sets of *information entities* (IEs). The structure of the cases allows to distinguish different text sections and the attributes section. The relevant text sections and the attributes section have to be mapped to IEs.

For mapping plain text, a parsing algorithm uses one of the dictionaries containing the set of all known IEs:

index term: a concept representing a set of strings describing the same semantic content like grammatical forms, abbreviations or different kinds of spelling (e.g. "PostScript-Drucker" : "PS-Drucker", "Postscript printer").

index vocabulary: the set of all strings known within a case based system that can be mapped to index terms.

Every index term has an IE as representant. The set of index terms (resp. textual IEs) is divided into different categories, e.g. the category "domain specific term", "computer term" or "general term" .

category of an IE: a subset of the set of IEs containing all textual IEs semantically belonging to the same class of concepts.

index dictionary: contains the set of all index terms, the rules of mapping the index vocabulary unique to index terms and an unique mapping of each index term to an IE category.

For representing a text with a set of textual IEs the parsing algorithm

- reads the text word for word from left to right,
- looks for the first occuring word or sequence of words matching with a string of the index vocabulary (if a shorter and a longer one start with the same word, the longer one is preferred),
- maps this string to the according index term of the index dictionary and
- adds the according textual IE to the set of IEs representing the case.

Beyond this technique we use all attribute-value pairs occuring in any attributes section directly as IEs.

We apply a composite similarity measure: The more IEs are shared by two cases the more similar are these cases. A local similarity function is defined, which compares any two IEs. Doing so, two cases may also be similar if they are expressed in completly different words which, however, can be mapped to similar IEs. The values of the local similarity function is represented by a similaity edge of the CRN.

The retrieval is done by a spreading activation mechanism: The incoming query activates IEs. They propagate their activation along the similarity and relevance edges. The result is a preference ordering of cases based on the achieved activations.

Open Issues

Evaluation is not yet addressed sufficiently. However, measures like precision and recall are not directly applicable for our purposes (Lenz 1998). For example a satisfying result is already achieved if only one matching case is presented to the user.

A major topic of our future work is improving the data maintenance. The expansion of the set of IEs should become more easy. In both projects, most of the IEs have been gained manually by extracting them from the cases or from general sources like dictionaries or glossaries. In future, a suggestion should be generated automatically.

To improve the performance of the system we are currently testing further Information Extraction methods. With these methods we can extract attributes and IEs more effectivly. Also the maintenance of the local similarities should be more user-friendly. Perhaps some kind of similarity assistant might be useful.

Another topic is developing the similarity measure. We think about including shallow *Natural Language Processing* or using more *Information Retrieval* methods.

References

Gierl, L., and Lenz, M., eds. 1998. *6th German Workshop on CBR*, Rostock: University of Rostock.

Görz, G., and Hölldobler, S., eds. 1996. *KI-96: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, 1137. Springer Verlag.

Kunze, M. The ExperienceBook. http://informatik.hu-berlin.de/~kunze/ studienarbeit.ps.

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

Lenz, M., and Burkhard, H.-D. 1996. Case Retrieval Nets: Basic ideas and extensions. In Görz and Hölldobler (1996), 227–239.

Lenz, M. 1998. Textual CBR and Information Retrieval — A Comparison —. In Gierl and Lenz (1998), 59–66.