



Interactive CBR for Precise Information Retrieval

Dr. Ashwin Ram, CEO, Enkia Corporation

Dr. Mark Devaney, CTO, Enkia Corporation

To appear in SK Pal, DW Aha, & KM Gupta (eds.),
Case-Based Reasoning in Knowledge Discovery and Data Mining
John Wiley Publishers, New York

Enkia Corporation • 75 Fifth Street NW, Suite 354 • Atlanta, Georgia 30308 • www.enkia.com

Introduction

The knowledge explosion has continued to outpace technological innovation in search engines and knowledge management systems. It is increasingly difficult to find relevant information, not just on the World Wide Web at large but even in domain-specific medium-sized knowledge bases—online helpdesks, maintenance records, technical repositories, travel databases, e-commerce sites, and many others. Despite advances in search and database technology, the average user still spends inordinate amounts of time looking for specific information needed for a given task.

This chapter describes an adaptive system for the precise, rapid retrieval and synthesis of information from medium-sized knowledge bases in response to problem-solving queries from a diverse user population. We advocate a shift in perspective from “search” to “answers”—instead of returning dozens or hundreds of hits to a user, the system should attempt to find answers that may or may not match the query directly but are relevant to the user’s problem or task.

This problem has been largely overlooked as research has tended to concentrate on techniques for broad searches of large databases over the Internet (as exemplified by Google) and structured queries of well-defined databases (as exemplified by SQL). However, the problem discussed in this chapter is sufficiently different from these extremes to both present a novel set of challenges as well as provide a unique opportunity to apply techniques not traditionally found in the information retrieval literature. Specifically, we discuss an innovative combination of techniques—case-based reasoning coupled with text analytics—to solve the problem in a practical, real-world context.

We are interested in applications in which users must quickly retrieve answers to specific questions or problems from a complex information database with a minimum of effort and interaction. Examples include internal helpdesk support, web-based self-help for consumer products, decision-aiding systems for support personnel, and repositories for specialized documents such as patents, technical documents, or scientific literature. These applications are characterized by the fact that a diverse user population accesses highly focused knowledge bases in order to find precise answers to specific questions or problems. Despite the growing popularity of on-line service and support facilities for internal use by employees and for external use for customers, most such sites rely on traditional search engine technologies and are not very effective in reducing the time, expertise, and complexity required on the user’s part.

The ideal solution is one that returns single (or small number of) answers to a user’s question along with a confidence rating and justification for each answer. Depending on the domain, the “answer” may be a technical support document, a suggested resolution, or a prototypical similar problem description from the database. For example, consider a technician in the control center of an electricity plant who is faced with an unexpected shutdown of the gas turbine that powers the plant. Or consider an average user who logs in to the web site of a computer manufacturer to find out why her color printer is printing bands across the paper. Ideally, the user

should be able to obtain an answer to the problem (such as “exhaust thermocouple overload” or “printer driver obsolete”, respectively) along with a few examples that serve as evidence and provide additional details.

This problem differs from traditional search paradigms in several ways. Unlike traditional web search, the problem requires precise search over medium-sized knowledge bases; it is not acceptable to return hundreds or thousands of results matching a few keywords even if one or two of the top ten are relevant. Unlike traditional information retrieval, the problem requires synthesis of information; it is not acceptable to return a laundry list of results for the user to wade through individually but instead the system must analyze the results collectively and create a solution for the user to consider. And unlike traditional database search, the users are both experts who know how to ask the appropriate questions and non-experts who have more difficulty in knowing the exact question to ask or the exact database query to pose. For this reason, existing approaches, such as employing natural language to SQL converters as front ends to relational databases, are not sufficient.

Research Problem

As increasing amounts of information are stored electronically, the need for simple yet powerful techniques for using these knowledge stores becomes increasingly crucial. Organizations and businesses are adopting new business processes that include these knowledge systems in the decision-making loop. From customer support representatives to high-level executives, people need intuitive, easy, and precise access to information to aid in effective performance of their tasks. Whereas in the consumer world people have gotten used to wading through large amounts of irrelevant information to find a news article, a book, or a cheap air fare, in the workplace it is critical that information be filtered, digested, and presented in a manner that minimizes the amount of information the user must examine manually to make timely and effective decisions.

The specific problem addressed here is that of *technical support*: helping a user diagnose a problem with a complex piece of equipment. When you’re looking for a vacation spot in Costa Rica, you want information about several destinations and you’d like to see several different articles, reviews, and so forth. When you’re looking for a reason why your new CD-ROM is not recognized by the computer’s motherboard, however, you don’t want a wide range of results; you want a precise answer even though you may not know exactly how to formulate the query.

Informally, the problem lies on the continuum between web search and relational database search. More formally, the problem has the following characteristics:

- *Diverse users*: The user may be an end-customer (such as the average user with the motherboard problem) or a highly-trained technician (such as the power plant operator).
- *Specialized knowledge bases* that are medium-sized, focused, unstructured: Knowledge bases are not as large or as diverse as the entire World-Wide

Web, yet they are unstructured, may contain free text documents, and may not share semantics or ontologies between them.

- *Precise search*: Search queries may be longer than the one or two word queries typical for web search, but they are unlikely to contain all the right keywords. Yet it is not acceptable to return dozens or hundreds of irrelevant results, even if the right answer is amongst them. In information retrieval terms, “precision” is important and may not be traded off for “recall”.
- *Knowledge synthesis*: The user expects the system to provide an “answer” and not simply a list of documents to read in which the answer may be buried. The system needs to integrate and correlate information from multiple documents, from multiple data sources, and/or from multiple reasoning strategies so as to develop a specific recommendation for the user. The system may need to provide an explanation for the recommendation in terms of supporting cases or documents that were retrieved from the knowledge bases.
- *Interactivity*: The session may take the form of a brief dialog, in which the query is successively refined based on user’s navigational behavior, preferably with little or no explicit effort on the part of the user.
- *Workflow integration*: Interaction with the system should be easy and should fit naturally into the normal workflow of the user. The actual workflow will depend on the application or task in which the user is engaged, but a simple user interface, while not part of the research *per se*, is an important element of the success of the project.

Example

Consider the following scenario: software developers seeking help with technical problems. The knowledge bases might be one or more databases containing technical documents, such as msdn.microsoft.com, hp.com/support, or search.intel.com/support. More broadly, one might include technical repositories, such as trouble tickets and maintenance records.

These databases contain technical question-and-answer documents that are written as unstructured, free-form FAQs (“frequently asked questions”). For example, support.microsoft.com/default.aspx?scid=kb;en-us;307145 explains a problem with the ASUS P2B motherboard that causes it to stop responding if Windows XP is installed, and docs.info.apple.com/article.html?artnum=61858 explains what to do if Keynote quits unexpectedly during a slide show. These documents are written in freeform English with little or no structure, and any structure that might exist is not shared across different knowledge bases even from the same manufacturer.

Why is it difficult to search these knowledge bases? Consider the following example, based on an actual problem that one of our colleagues had recently. He had just

bought a Dell computer with a USB 802.11 card, and was attempting to connect to the Internet via an existing AirPort-based WiFi home network. While his Macs had no trouble recognizing the AirPort Base Station, the Dell was unable to do so. He went to AppleCare (apple.com/support) and typed “my windows machine does not recognize my airport” into the search box. It returned zero results.

He then typed “windows airport” and got about 100 results, each being a detailed technical document. Results included:

- titles that were clearly irrelevant (e.g., “Using Simple Finder”)
- titles that appeared relevant but were not (e.g., “Base Stations Don’t Appear In Scan List”—this applies to Windows AAU 3.4 only)
- titles that might have been relevant but he had no way to tell (e.g., “Windows Clients Cannot Connect to Encrypted Network (WEP)” —what is WEP, and would it preclude the computer from connecting to the network or from recognizing it in the first place?)

Reading a few dozen multi-page technical documents was difficult enough, but what was more frustrating was that this person had no idea what to do next. How should he refine his query so as to find fewer but more relevant results?

The above example is taken from a lay user’s perspective, but even expert software developers are familiar with the frustrating experience of trying to resolve an issue using MSDN or other technical repositories. Many are turning to user groups for help; witness the popularity of Google Groups or www.orafaq.com for this purpose. The issue is not one of inadequate search technology (dozens of companies offer specialty search engines for every application one might imagine) but of leveraging search technology within a broader troubleshooting framework. Returning to the “windows airport” example, the retrieved documents are relevant to the query (they contain the specified keywords, after all) but not to the *intent* of the query (which is to resolve a problem whose symptoms can be described in multiple ways).

Solution

Our approach to the above problem is to formulate it as a problem of *decision making* instead of one of *information retrieval*. The solution is based on an artificial intelligence approach known as *case-based reasoning (CBR)*, in which past cases are used to solve new problems (Kolodner, 1993; Leake, 1996). The key lies not in running a smarter search engine against a set of documents, but in understanding which documents contain appropriate answers to different kinds of queries. While driven by information retrieval techniques, the decision-making process goes beyond matching queries against *documents* to matching queries against *cases*. Cases are stored in a *case library* and represent the system’s experience or historical record of previous queries and responses.

We implemented the solution using Enkia’s Sentinel product. Sentinel is an artificial intelligence product for equipment maintenance. It can be used in two ways:

- Real-time monitoring of operational equipment for failure prediction and root cause diagnosis
- Interactive analysis of faults for problem resolution and repair recommendation

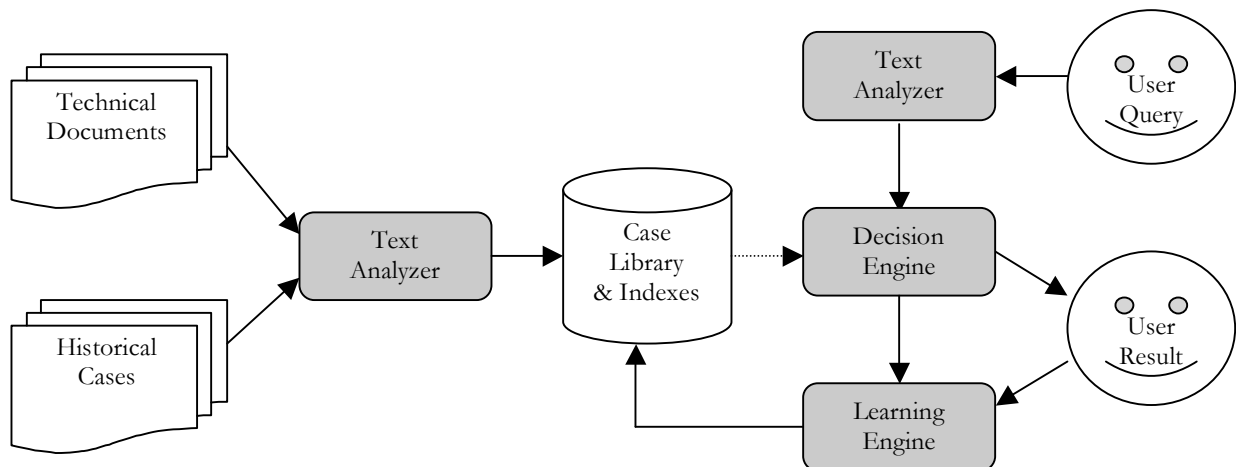
The latter configuration was used for the results reported in this chapter. Sentinel is built on advanced technologies that merge real-time and historical data with advanced case-based decision-making algorithms to analyze situations, predict problems, and recommend solutions. The system combines three kinds of artificial intelligence (AI) technologies into a robust, commercial-quality AI architecture with a professionally-designed user interface. The technologies are:

- *Text Analytics*: Methods for parsing, analyzing, indexing, retrieving, correlating, and clustering text, including user queries, knowledge documents, and historical cases.
- *Decision Making*: Methods for making decisions based on retrieved data or “cases” in the knowledge base.
- *Machine Learning*: Methods for using relevance feedback from users to associate queries to relevant documents.

These AI technologies are brought together via two additional components:

- *Architecture*: Blackboard architecture for scalable integration of hybrid technologies.
- *User Interaction*: XML-based user interface for easy customization.

The system is conceptualized in the following (simplified) diagram:



Using a product approach enabled us to build a solution that can be used and tested by potential users in addition to the researchers. The technologies embedded in Sentinel’s AI modules (*Text Analyzer*, *Decision Engine*, and *Learning Engine*) were evaluated through experimentation with a complete, testable system, as discussed later in this chapter.

Related Work

Our approach leverages a key element of human expertise that is largely lacking in computer systems: *experience*. Humans develop expertise in a domain through experience. They remember their experiences and apply them in new situations. They develop a repertoire of *cases*, situations they have seen, problems they have solved, solutions they have adopted, lessons they have learned. This approach has been algorithmized into a family of techniques known as *case-based reasoning (CBR)* (Kolodner, 1993; Leake, 1996).

CBR systems have been applied to a number of problem solving domains, such as planning (Hammond, 1989), mediation (Simpson, 1985), and design (Goel et al, 1997; Sycara et al, 1992). In our own work at Georgia Tech, we have focused on enhancements that allow CBR to be scaled and applied to large, real-time domains. For example, the system may proceed with a "best guess" case while continuing to search asynchronously for a better case (Francis & Ram, 1997). The system may reason with more than one case or with pieces of cases (Ram & Francis, 1996; Redmond, 1990). The system may interleave reasoning and learning in a real-time or "anytime" manner (Ram & Santamaria, 1997).

In industry, CBR has been successfully applied to problems as diverse as autoclave layout (Lockheed-Martin; Barletta & Mark, 1988; Hinkle & Toomey, 1995), power forecasting (Niagara Mohawk Power Corp; Jabbour et al, 1988), printer troubleshooting (Compaq; Nguyen et al, 1993), software quality control (NEC; Kitano et al, 1992), aircraft fault diagnosis (British Airways; Magaldi, 1994), and turbine fault diagnosis (GE Power Systems; Zaremba & Ram, 2003). Some successful applications are listed at www.ai-cbr.org/applied.html.

While these successes are encouraging, there has been less work in academia or industry in applying the CBR approach to problems involving unstructured text. A few researchers have applied CBR methods to natural language analysis (Cox & Ram, 1999; Ram, 1994), but this research is not yet mature and with few exceptions has not been scaled to industry-strength problems. Nevertheless, it is instructive to compare these approaches to coupling CBR and information retrieval (IR) with the approach proposed here.

Rissland et al (Ashley & Rissland, 2003; Rissland & Daniels, 1995) describe a hybrid CBR-IR system in which CBR is used to drive IR. Their system uses a standard frame-based representation of a problem and matches it against frame-based representations of cases using a previously developed CBR system called HYPO. Documents stored in those cases are then used to construct keyword queries, which are run against a large document repository using the INQUERY search engine. This approach relies on frame-based symbolic AI representations, unlike our approach, which runs CBR directly on vector space representations of text documents. In addition, their approach returns a potentially large set of documents for the user to interpret, whereas we use a decision making engine that supports the user in making decisions based on retrieved documents.

Roth–Berghofer and Iglezakis (2000) describe a CBR-based helpdesk application in which a text analyzer is used to extract concepts from queries and fill in attribute slots in a pre-specified frame-based representation known as an object model. Like Rissland et al and unlike our approach, their approach requires domain analysis to develop frame-based representations, which are then used as the basis for the CBR engine. Our approach, in contrast, was to develop a CBR engine capable of using text representations directly as the basis for case representation, indexing, retrieval, application, and learning.

In this respect, our approach is more similar to that of Lenz et al (1998), whose Textual CBR method applies CBR directly to text representations. However, their system requires domain knowledge and stops at information retrieval, whereas ours is domain independent and adds a decision-making component. Wilson and Bradshaw (1999) also use text representations in the form of vector space models, coupled with cosine similarity metrics for matching. Their approach is perhaps the closest to ours, with the difference that they too rely on domain knowledge in what they call a “weakly textual CBR system.”

Gupta and Aha (2004) survey several methods for indexing text cases, and conclude that these approaches require manual effort to assign indexes to cases as opposed to developing indexes automatically. They propose an approach that uses a generative semantic lexicon and a logical form representation for extracted features. In contrast, our approach uses vector space representations to minimize the need for domain-specific lexicons and other specialized knowledge. Other relevant approaches include those of Bruninghaus and Ashley (2003), Burke and Hammond’s FAQ Finder (Burke et al, 1997), Marinilli et al (1999), and others summarized in the *AAAI Workshop on Textual CBR* (Lenz & Ashley, 1998); see, e.g., Kunze & Hubner (1998).

While the above list is not comprehensive, it serves to illustrate some of the similarities and differences of our approach and other approaches in the literature. A key difference is that most existing approaches treat *documents* as cases, reducing the underlying problem to one retrieving the best cases, analogous to the information retrieval problem of finding the best matching documents. In contrast, we propose a case representation that captures historical problem solving experiences of the system, which in this domain are comprised of past queries, answers, confidence metrics, and justifications. In this sense, our approach is closer to Kolodner’s (1993) description of case-based reasoning as an approach to solving problems using actual *experiences* as opposed to *knowledge*. This being said, it should be noted that our approach has much in common with the other approaches mentioned above, and we have built on the existing body of knowledge as much as possible.

CBR Architecture

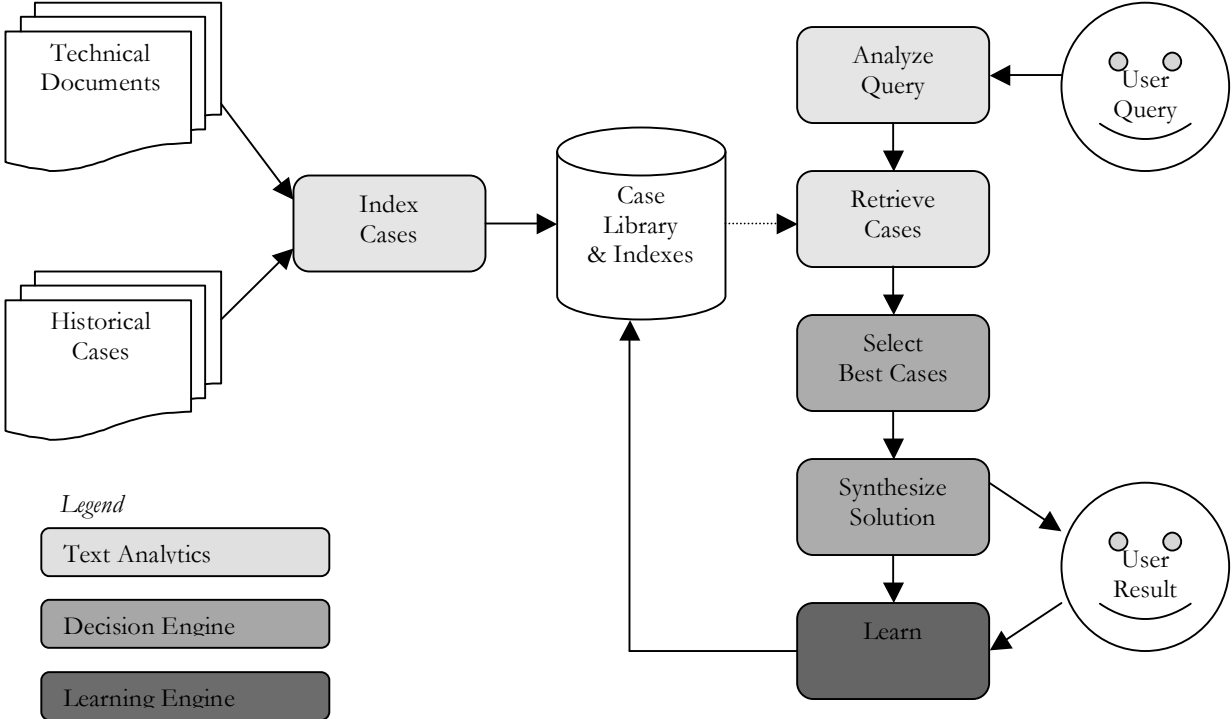
A typical case-based reasoning (CBR) system works as follows. Given a new problem: (i) *retrieve* a few past cases from a “case library” that are close to the new problem in a suitable representation space; (ii) for each retrieved case, calculate a similarity metric between the case and the new problem and *select* the best match; (iii)

apply the solution in the selected case to the new problem; (iv) *learn* by modifying the proposed solution based on feedback from execution and storing it back in the case library.

In our project, we used Enkia’s Sentinel product to address the technical support problem. Sentinel uses the familiar CBR cycle (*retrieve, select, apply, learn*) but with the following differences:

- *retrieve* and *select* require text analytics, since the problem description and cases are unstructured text instead of traditional AI representations
- *apply* requires synthesis of information from multiple retrieved cases rather than straight application of a single best match
- *learn* requires user modeling with human-in-the-loop relevance feedback, and requires modification of existing indexes in addition to storing new cases in the case library

Using this framework, we can refine the previous conceptual diagram into the following cognitive architecture:



Instead of matching queries against keywords in technical documents, the system develops a case library of past problem-solving sessions containing previous queries the system has seen and corresponding solutions the system has proposed.

The key research question, then, is how a system can perform case-based reasoning with textual information—how it indexes, retrieves, selects, synthesizes; how it learns by interacting with the user; and how these algorithms can be integrated into a robust, interactive system. More specifically, this approach raises the following research issues:

- *Knowledge Representation*: What information does a case contain? How is this information represented? How is the case library initialized and what happens if there are no past cases for a given query?
- *Indexing & Retrieval*: How are cases organized to enable relevant cases to be found later? How are cases retrieved in response to a user's query? How is the relevance of a case determined?
- *Decision Making*: How are multiple cases combined to produce the final answer(s)? How is the level of confidence in an answer determined?
- *Learning*: How are new cases learned? How are indexes and cases updated through experience?

Answering these questions requires developing an algorithmic solution to the problem of performing case-based reasoning with unstructured text. Details of our solution are presented next.

Knowledge Representation (Case Library)

The key issue in a case-based reasoning approach is the nature of a case. The problem at hand is characterized by the following workflow:

- a) User has a technical problem
- b) User poses a text query
- c) System suggests one or more answers
- d) System justifies its answers by showing relevant documents

Between steps b and c, the system must:

- i.) Find previous cases that are relevant to the new problem
- ii.) Retrieve answers from those cases, both good and bad
- iii.) Determine one or more answers to the new problem
- iv.) Develop a justification structure to present to the user

In the traditional information retrieval paradigm, queries are matched against documents in the corpus or database using a “search engine” to match keywords and rank results based on keyword similarity between query and documents. There has been a large amount of academic and commercial research into improving and optimizing this process.

While this approach is suitable for search engine applications, it does not solve the problem discussed at the beginning of this chapter. Every document with one or more of the specified keywords is returned in the result set. As illustrated by the “my windows machine does not recognize my airport” example mentioned earlier, either many irrelevant documents are returned (e.g., any document with the word “airport” in it) or relevant documents may be missed because of an overly specific keyword (e.g., the word “recognize” may rule out relevant documents).

Thus, it is not effective to match the query directly against the technical document database. Instead, Sentinel matches queries not against *documents* but against previous *cases* or problem-solving episodes: specifically, against *queries* that the system has previously answered (primary index) and against *supporting documents* that the system has used to justify those previous answers (secondary index).

Sentinel’s representational structures are based on CBR methodology. The representation distinguishes between *content features*, which are all the features in the feature vector representation of a document’s content, and *index features*, which are relevant features used to index a case. Continuing the above example, “windows” and “airport” might be good index features whereas “machine” and “recognize” might not, although all might appear in the content feature vector. The question, then, is what a case consists of and which parts of it constitute an effective index.

Sentinel’s knowledge base consists of a case library in which each case contains the following components:

<i>Structure of a case</i>
<ul style="list-style-type: none"> • Query • Answers • Confidence metrics (relevance, ranking) of the answers • Justifications (supporting documents) for the answers • Correctness of the answers (learned from user feedback)

Since queries, answers, and documents are, ultimately, natural language text, they are represented using *feature vectors* developed using information retrieval and text analytics techniques as discussed below. Exploring detailed alternatives for case representations is a key issue for future research.

Sentinel is initialized with technical documents in the knowledge base. The title of these documents is typically a one-sentence question or summary that constitutes an idealized query (e.g., the title of `support.microsoft.com/default.aspx?scid=kb;en-us;307145` is “Computer That Has ASUS P2B Motherboard Stops Responding After Windows XP Installation”). This is represented as a feature vector and stored in the “query” part of the case with the full document as the single “answer” with 100% confidence.

Over time, Sentinel develops a library of cases, each with the structure described above, based on actual usage and user feedback. In effect, the system learns a *many-many mapping between queries and answers* through an interactive case-based reasoning process.

The innovative aspect of the proposed representation is the capturing of the system’s experiences, i.e., past query-answering episodes, in case representations as opposed to representing knowledge documents directly as cases.

Algorithms (Case-Based Reasoning with Text Analytics)

Indexing & Retrieval

Indexing and retrieval correspond to the following modules in the above architectural diagram: Index Cases, Analyze Query, and Retrieve Cases. The key technology used in these modules is Text Analytics: a family of algorithms responsible for parsing, analyzing, indexing, retrieving, and matching of text information. Using these algorithms, Sentinel analyzes the documents in the database to construct a case library (*index cases*), analyzes the user's query (*analyze query*), and finds and assesses relevant cases from the case library in response to the user's query (*retrieve cases*).

The basic text analysis cycle works as follows. Given a piece of text, the system computes a "feature vector" comprising of a set of pairs, each pair being a "feature" coupled with a "weight" or importance of that feature in that document. This is known as the *vector space model* (VSM) in the literature (Salton & McGill, 1986). A feature may be an individual word, a pair of words or bigram, a phrase, or a concept inferred from the text. Weights are determined using heuristics such as the number of occurrences of the feature, the location of those occurrences within the document, the number of documents in the entire corpus containing that feature, and other metrics.

This text analysis cycle is used to compute feature vector representations of queries (primary index) and justification documents (secondary index). These are stored as part of the {query, answers, confidence metrics, justification documents, correctness} case representation tuple. When a new query is received, the same text analysis cycle is used to compute a feature vector representation of the query, which is then matched against the primary and secondary indexes to retrieve relevant cases. Matching consists of computing similarity or distance in feature space using an appropriate similarity metric. The primary index helps in retrieving similar past queries, whereas the secondary index helps in retrieving a broader range of documents if similar queries have not been encountered in the past. This allows the system to improve precision without sacrificing recall.

Decision Making

Decision-making corresponds to the following modules in the above architectural diagram: Select Best Cases and Synthesize Solution. Sentinel selects one or more cases retrieved in response to a user query (*select best cases*) and develops a proposed solution to the user's query (*synthesize solution*) using the CBR cycle described earlier. By matching the query against the index structure of the case library, a base set is retrieved as a starting point. This set is placed into the blackboard for further processing by individual modules. Each set is ranked with respect to the user's query, using an appropriate distance metric in feature space. This metric provides a weight or relevance metric for each case.

For the results presented in this chapter, a weighted voting approach was used to compute the final answer. Each case suggests one or more technical documents (*answers*) each being weighted by its relevance to that query (*confidence metrics*). Each case, in turn, is weighted by its relevance to the new query. All the answers suggested by the retrieved cases are candidate answers for the new query, and the likely relevance or confidence metric for each answer is dependent on the relevance of the case to the query and the relevance of the answer to the case.

In the chosen domain, the “answer” consists of a set of technical documents that are suggested to the user. In other applications, the “answer” might be a specific resolution or recommended action that is encoded in the case. The system may show all answers and all documents, or all answers with documents shown on demand, or the best answer with other answers shown on demand. The system may also show other relevant documents automatically. These and other UI issues are dependent on the workflow that is appropriate to the application at hand, and outside the research scope of this project.

The innovative aspects of this module are the use of CBR algorithms with unstructured text as opposed to traditional symbolic AI/problem-solving representations, and the recommendation of answers instead of document retrieval alone.

Self-Learning

A key aspect of our approach is self-learning which allows the system to improve its performance over time. Sentinel achieves this by obtaining user feedback, also known as “relevance feedback,” and improving the case library by adding new cases, purging obsolete cases, and reorganizing the index structure to improve future retrieval. There are several metrics that can be used for self-learning, including popularity (most frequently-viewed documents are ranked higher), collaborative filtering (co-occurrence frequencies are used to bias search results), user modeling (short-term or long-term), and relevance feedback (user feedback is used to learn better mapping of queries to documents).

For this chapter, our primary research focus was on the latter. The system uses relevance feedback from the user to learn the weights stored in cases as *confidence metrics* for individual answers. While relevance feedback is typically used in IR systems for query augmentation (e.g., Salton & Buckley, 1990), it can also be used to modify confidence metrics so as to make it more or less likely that an answer will be suggested again based on whether it is considered a good or bad answer, respectively. These weights are adjusted based on explicit feedback from the user using a logarithmic weighting scheme (Robertson & Sparck Jones, 1988).

In future research, we plan to incorporate methods for relevance feedback based on implicit feedback—for example, if the user views an answer and then goes on to view another answer, the system may assume that the former did not answer the user’s question. We will also investigate more complex weight adjustment schemes using adaptive clustering which has been shown to be effective for content-based retrieval systems (Kim & Chung, 2003).

Popularity and collaborative filtering are less likely to be valuable for the problem at hand. We want the right answers, not the most viewed answers. User modeling may enhance the effectiveness of the system by improving focus on cases that are more relevant to a given user. Enkia has previously developed proprietary user modeling algorithms. These algorithms allow the system to track a user’s behavior and modify the system’s retrieval behavior based on knowledge of the individual user and group population. Reinforcement learning techniques may also be used to learning mappings of problem queries to answer documents via feedback that occurs several steps away from the initial document set; this is referred to as *delayed reward* (Kaelbling, Littman, & Moore, 1996). These are important issues for future research.

Evaluation

In order to evaluate the approach, Enkia’s Sentinel product was integrated with an existing medium-sized database of approximately 16,000 technical support documents. 100 of these documents were randomly selected as a set of “targets” – the desired solutions to 100 unique problems. For each of the documents, 10 unique queries were generated, simulating the various ways human users might formulate a query for each particular problem. The queries were generated by randomly permuting the title of each of the target documents by removing terms, adding common words, or adding words from other portions of the document.

Original Title	Sample Query
INFO: Changing Window Colors with Control Panel	Window Colors the with article Serial
	INFO: this the 3.1 Control in Panel
Dates in Unexpected Order in AutoFilter Drop-down List	Windows Unexpected Excel Order Excel 3.1 information Can't
	The 7.0a Microsoft the in Open Excel Drop-down 95 Dialog
CE_OVERRUN Errors with Serial Communications	Printer in article Serial Windows Prefix
	Message article with information Serial Communications

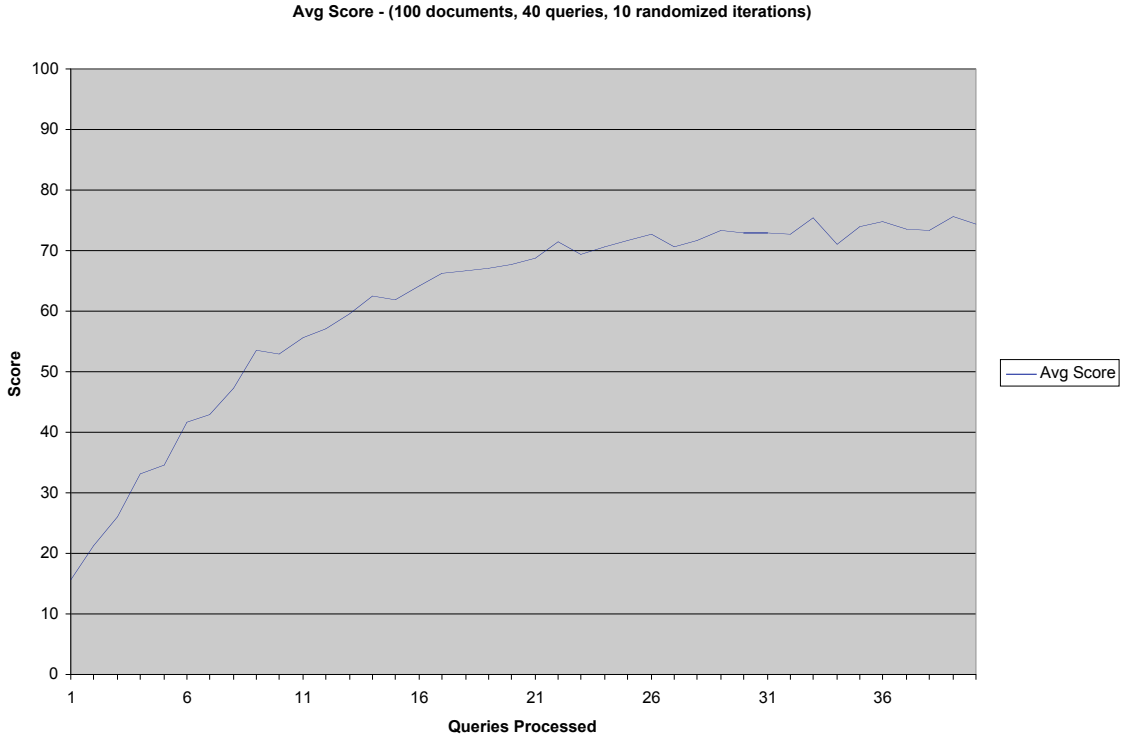
Each of the 10 unique queries was repeated 4 times, providing 40 queries for each of the 100 target documents. These queries were randomly ordered and issued to the information retrieval system which returned 100 documents in response to each query. The system’s performance for each such episode was scored as the position of the target document in the returned results. Appearing first in the list resulted in

the highest possible score (100) and not appearing in the list resulted in the lowest possible score (0).

For each of the returned documents, user feedback was simulated by marking each non-target document as “not relevant” and the target document, if present, as “relevant”. Thus, each of the 16,000 episodes consisted of a query presented to the system, a score computation based on the position of the target document for that query, and issuance of feedback to the system.

Results

The use of feedback to bias information retrieval resulted in substantial improvement in the score as more experiences were accumulated. The following figure depicts the average score over all target documents as a function of the amount of feedback received.



As can be seen from this figure, Sentinel learns to associate the “right” answers to user queries, performing significantly better than the initial condition where it is essentially matching query keywords to documents since it does not have the experiential knowledge to do anything different. The final accuracy score levels off at 75% to 95% depending on the dataset; a representative run at the lower end of this range is shown in the figure above.

Conclusions

Interactive text-based CBR is a new and effective method for addressing the problem of precise information retrieval. We summarize the answers to the research issues raised earlier as follows:

Knowledge Representation:

- *What information does a case contain?* Past problem-solving episodes, including user queries, proposed solutions, correctness of proposed solutions, justifications (technical documents) for proposed solutions.
- *How is this information represented?* In a schema that contains feature vector representations of underlying textual data, produced using text analytics and information retrieval algorithms.
- *How is the case library initialized?* Using queries or summaries embedded in knowledge documents (e.g., in their titles) as the initial set of cases, and bootstrapping the system from that point.

Indexing & Retrieval:

- *How are cases organized to enable relevant cases to be found later?* Via indexes containing relevant features identified through an information gain metric.
- *How are cases retrieved in response to a user's query?* By matching query vectors against case indexes (primary) and justification documents (secondary).
- *How is the relevance of a case determined?* Using a distance metric in feature space.

Decision Making:

- *How are multiple cases combined to produce the final answer(s)?* By correlating past solutions stored in the retrieved cases to find highest-likelihood solutions to the new problem, using a weighted voting algorithm and a conceptual clustering algorithm.
- *How is the level of confidence in an answer determined?* By combining the relevance of a case to the query with the confidence metrics of the answers provided by that case.

Learning:

- *How are new cases learned?* By storing new queries and associated answers into the case library.

- *How are indexes and cases updated through experience?* Via user-provided feedback that is analyzed by a relevance feedback algorithm and a reinforcement learning algorithm (future).

Sentinel can be used for a range of self-service problems: in product companies such as who need better web support for their customers, in technology companies who need tools for their customer support representatives and engineers who are on call to help their customers, in large financial institutions who need better internal tools for their technical support staff. These companies face a common problem: the necessary information is available in their and their partners' repositories and databases, yet their customers and staff are unable to find what they need in an efficient and reliable manner. Typically, these companies have already tried search engine technology and are looking for a solution from the AI community. We believe our perspective, that of *decision support* instead of *document retrieval*, is innovative and can be used fruitfully in other applications that have similar characteristics.

References

- K.D. Ashley & E.L. Rissland (2003). Law, learning and representation. *Artificial Intelligence*, 150(1-2):17-58.
- R. Barletta & W. Mark (1988). Explanation-based indexing of cases. In J.K. Kolodner (ed.), *Proceedings of the DARPA Case-Based Reasoning Workshop*, Morgan Kaufmann.
- S. Bruninghaus & K.D. Ashley (2003). Combining model-based and case-based reasoning for predicting the outcomes of legal cases. In D. Bridge & K.D. Ashley (eds.), *Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning*, Springer Verlag.
- F.-C. Cheong (1996). *Internet Agents: Spiders, Wanderers, Brokers and Bots*. New Riders.
- M.T. Cox & A. Ram (1999). Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence*, 112:1-55.
- A.G. Francis & A. Ram (1997). Can your architecture do this? A proposal for impasse-driven asynchronous memory retrieval and integration. In *Proceedings of the AAAI-97 Workshop on Robots, Softbots, Immobiles: Theories of Action, Planning, and Control*. AAAI Press.
- A. Goel, S. Bhatta, & E. Stroulia (1997). Kritik: An early case-based design system. In M. Maher & P. Pu (eds.), *Issues and Applications of Case-Based Reasoning in Design*, Erlbaum.
- K.M. Gupta & D.W. Aha (2004). Towards acquiring case indexing taxonomies from text. In *Proceedings of the Sixteenth International Conference of the Florida Artificial Intelligence Research Society*, AAAI Press.

- L. Kaelbling, M. Littman, & W. Moore (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4.
- K. Hammond (1989). *Case-Based Planning*. Academic Press.
- B. Hayes-Roth (1985). A blackboard architecture for control, *Artificial Intelligence*, 26(3):251-321.
- D. Hinkle & C. Toomey (1995). Applying case-based reasoning to manufacturing. *AI Magazine*, 16(1):65-73.
- K. Jabbour, J.F.V. Riveros, D. Landsbergen, & W. Meyer (1988). ALFA: Automated load forecasting assistant. *IEEE Transactions on Power Systems*, 3(3):908-914.
- D.-H. Kim & C.-W. Chung (2003). Qcluster: Relevance feedback Using adaptive clustering for content-based image retrieval. *ACM SIGMOD*, San Diego, CA.
- H. Kitano, A. Shibata, H. Shimazu, J. Kajihara, & A. Sato (1992). Case-Method: A methodology for building large-scale case-based systems. In *Proceedings of the Tenth AAAI Conference*, AAAI Press.
- J.L. Kolodner (1993). *Case-Based Reasoning*. Morgan Kaufmann.
- M. Kunze & A. Hubner (1998). Textual CBR – Case studies of projects performed. In *Papers from the AAAI-98 Workshop on Textual Case-Based Reasoning*, AAAI Press.
- D.B. Leake (1996). *Case-Based Reasoning: Experiences, Lessons and Future Directions*. AAAI Press/MIT Press.
- M. Lenz, A. Hubner, & M. Kunze (1998). Question answering with textual CBR. In T. Andreasen, H. Christiansen, & H.L. Larsen (eds.), *Flexible Query Answering Systems*, Springer-Verlag.
- M. Lenz & K.D. Ashley (eds.) (1998). *Proceedings of the AAAI-98 Workshop on Textual Case-Based Reasoning*. AAAI Press.
- M. Marinilli, A. Micarelli, & F. Sciarone (1999). A case-based approach to adaptive information filtering for the WWW. In *Proceedings of the Second Workshop on Adaptive Systems and User Modeling on the World Wide Web, Sixth International Conference on User Modeling*.
- R.V. Magaldi (1994). Maintaining aeroplanes in time-constrained operational situations using case-based reasoning. In J.-M. Haton, M. Keane & M. Manago (eds.), *Advances in Case-Based Reasoning*, Springer-Verlag.
- S. Minton (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42.
- T. Nguyen, M. Czerwinski, & D. Lee (1993). COMPAQ QuickSource: Providing the consumer with the power of artificial intelligence. In *Proceedings of the Fifth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press.
- A. Ram (1994). AQUA: Questions that drive the explanation process. In R.C. Schank, A. Kass & C.K. Riesbeck (eds.), *Inside Case-Based Explanation*, Lawrence Erlbaum.

- A. Ram & A.G. Francis (1996). Multi-plan retrieval and adaptation in an experience-based agent. In D.B. Leake (ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press.
- A. Ram & J.C. Santamaria (1997). Continuous case-based reasoning. *Artificial Intelligence*, 90(1-2):25-77.
- M. Redmond (1990). *Learning by Observing and Explaining Expert Problem Solving*. PhD dissertation, College of Computing, Georgia Institute of Technology, Atlanta, GA.
- S.E. Robertson & K. Sparck Jones (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129-46.
- E.L. Rissland & J.J. Daniels (1995). Using CBR to drive IR. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal.
- T. Roth-Berghofer & I. Iglezakis (2000). Developing an integrated multilevel help-desk support system. In *Proceedings of the Eighth German Workshop on Case-Based Reasoning*.
- G. Salton & M.J. McGill (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill.
- G. Salton & C. Buckley (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288-297.
- R.L. Simpson (1985). *A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation*. PhD dissertation, Georgia Institute of Technology, Atlanta, GA.
- K. Sycara, R. Guttal, J. Koning, S. Narasimhan, & D. Navinchandra (1992). CADET: A case-based synthesis tool for engineering design. *International Journal of Expert Systems*, 4(2).
- D.C. Wilson & S. Bradshaw (1999). CBR textuality. In *Proceedings of the Fourth UK Case-Based Reasoning Workshop*, University of Salford.
- S. Zaremba & A. Ram (2003). CBR Engine for Diagnostics of Gas Turbine Trips. Paper presented at the Industry Day of *the Fifth International Conference on Case-Based Reasoning*, Trondheim, Norway.

Acknowledgements

The authors wish to thank Evan Kerns for his assistance with this project.