Surveying the Opportunities for Textual CBR: A Position Paper

Robin D. Burke
Intelligent Information Laboratory
University of Chicago
1100 E. 58th St., Chicago, IL 60637
burke@cs.uchicago.edu

May 15, 1998

Task Orientation

Riesbeck (1997) writes that the goal for AI and CBR in particular should be understood as "building components that reduce the stupidity of the systems in which they function." (pg. 377) This is an excellent description of the niche that textual case-based reasoning can and should occupy. This position paper explores some of directions in which this goal can be pursued, with special attention to the role of task orientation.

Progress in information retrieval and databases have brought us powerful, flexible tools for managing large amounts of textual data, yet from a user's point of view, these systems often seem stupid. They aim for a least-common-denominator interface, ensuring that every possible query can be posed, but do little to help satisfy user's most common goals. They treat data uniformly, putting the burden on the user to distinguish structural elements that are relevant for a particular task. As a result, users must master complex querying conventions to accomplish even simple tasks.

To this problem, case-based reasoning brings the legacy of planning and problem-solving research. Case-based reasoners, like human reasoners, seek retrieval because they have a problem to solve. CBR systems build their indexing and retrieval mechanisms within knowledge frameworks that identify important features and problem-solving strategies. An effective textual

CBR system therefore will be one that focuses carefully on a particular task and the knowledge requirements it imposes. Such a system will eschew flexibility and generality for precision and utility for a given group of users. In this paper, I will discuss a few examples of working textual CBR systems that achieve this task focus and present the beginnings of a taxonomy of different types of textual CBR systems that might be built.

Examples[1]

One way that we can focus a system on a particular task is to have the system present a task-focused interface. The interface can structure the user's interaction with a body of data in a way that supports problem-solving.

For example, consider the ENTREE system (Burke, Hammond, & Young, 1997), a case-based information access system for finding restaurants. The system presents a page similar to that found in any other web-enabled database: a HTML form where the user can state their desires. Two things distinguish it, however. With the exception of "Cuisine," the features from which users select are not the same as those found in restaurant descriptions. The restaurant descriptions are too long and complicated to present every selection to the user. Instead the interface generalizes over these details, using abstract features in a way that will be familiar to those acquainted with CBR. Secondly, the interface presents the option: "I would like to eat at a restaurant just like:" letting the user point the system at his or her favorite eatery as a model.

The interaction that follows an initial user query also has its roots in CBR: the user has a small set of "tweaks" that can be applied to the restaurants retrieved by the system. If the recalled restaurant is too expensive, the user can select the "Less $$" tweak. If the restaurant looks too staid, the user can choose "Livelier". There are seven tweaks available. The system encourages an exploratory interaction with the database, always grounded in specific examples to which the user is responding.

ENTREE does not provide full access to every record in its database. It does not allow the user to construct a convoluted query: "show me all the

[1] We have found the world-wide web to be an excellent test-bed and data source for textual CBR. Both of the example textual CBR systems mentioned here are accessible via the web: ENTREE at http://infolab.cs.uchicago.edu/entree/; FAQ FINDER at http://faqfinder.cs.uchicago.edu/

restaurants that have wheelchair access, valet parking, and serve Italian or Mexican food." In some cases, certain entries cannot be retrieved by any query: the system tries to maximize quality achieved at a given price, so restaurants that occupy uncompetitive positions in the quality/price tradeoff will never be recalled. ENTREE focuses the interaction on the typical goals of a user: finding a restaurant that is a good value, serves the right kind of food and has the right kind of atmosphere.

In building ENTREE, we had to encode knowledge about restaurants: relations between concrete features like "good for singles" and abstract features like "Lively" and similarity relations between features such as the connection between "Caribbean" and "Jamaican" cuisine. This knowledge is essential for the system to do its job, but it does not represent a major knowledge acquisition hurdle.

A textual CBR system can also achieve task focus by structuring its handling of textual data. In CBR, we understand that the best index to a solution is the problem that it solves. If our textual CBR systems are designed to assist problem-solving, we can pay special attention to those parts of a text that constitute the problem and solution, and treat the problem as the best index. A problem-solving textual CBR system or textual CBR adjunct to a human problem-solver will benefit from the structure of documents to the extent that it is possible to identify elements that are good cues to the problem solved by a case. FAQ FINDER is a natural language question-answering system that uses USENET frequently-asked question files (Burke, et al. 1997). FAQ files can be considered collections of problem/solution (question/answer) pairs, excellent candidates for a case-based approach.

FAQ FINDER uses a combination of matching techniques to compare user questions with the question/answer pairs in FAQ files. Following standard information retrieval practice, the system uses a statistical technique comparing term vectors derived from the input and each question/answer pair, but this measure is combined with a "semantic score." The semantic score is derived from a comparison between questions only, and uses the WordNet semantic network (Miller, 1995) to evaluate the semantic "distance" between the two questions. It would be computationally prohibitive and noise-prone to extend this analysis to the entire question/answer pair, but our research found that special attention to the question part of the QA pair yielded significant performance improvements over a uniform statistical approach over the entire pair.

Taxonomy of Textual CBR Systems

These examples indicate two directions that my colleagues and I have already pursued in the textual CBR. Included below is a partial taxonomy of the different types of systems that might be built. Each branch of the taxonomy suggests a different task focus, and would profit from a different kind of user interface and different text/case interpretation technique. It is intended to be suggestive, rather than exhaustive, and I welcome additions to the list:

Selection systems: Systems that help the user identify an entity of interest, where the entities are described largely in text and can be treated as cases. ENTREE and the other FINDME systems from the Intelligent Information Laboratory are simple examples of this category. A more complex example: The magazine *Consumer Reports* regularly generates text reviews describing and comparing the products in a given category: washing machines, for example. An textual CBR system could find the most recent reviews of the item of interest to the user and organize the information to assist decision making.

Threat detection systems: Systems that help the user identify a possible threat, based on textual descriptions of the activities of others. Example: Treating corporate press releases as cases, a textual CBR system might enable a user to identify possible companies and products likely to be a competitive threat in a particular market segment.

Pattern identification systems: Systems that help the user locate patterns in textual items, such as building clusters of related items. Example: In large software projects, the bug database can be very large and may be manage by a large geographically-distributed group of people. A textual CBR system helping to manage the database could cluster and organize bugs as they are added, helping users determine if a given bug has already been identified and perhaps fixed, and helping developers identify aspects of a program that are particularly vulnerable to bugs.

Document assembly systems: Systems that help the user assemble a custom document from standard parts, such as legal boilerplate. Example: a contract generator might have many textual elements of contracts labeled by the issues they address, such as liability, redress, etc. A user would describe the terms of a contract and the system would

retrieve and assemble the necessary pieces. (Such a system would also require a reasonably sophisticated text-case adaptation mechanism.)

Question answering systems: Systems that treat texts as answers, to be retrieved in response to specific questions. Example: The FAQ FINDER system mentioned above and also the Chicago Information Exchange (Kulyukin, in preparation).

Traps to avoid

As the above list shows, there is much territory to be explored under the banner of textual CBR, and many avenues to explore. In the spirit of McDermott's classic "Artificial Intelligence Meets Natural Stupidity" (McDermott, 1981), I would like to volunteer a few of the conceptual hazards that I have encountered in my experience in textual CBR in hopes of helping others avoid them:

Wishful natural language understanding: In building systems that deal with text, it is easy to end up in the situation where the design requires that the system understand the text it processes, in the sense of being able to build a deep representation of its contents. For example, a case-based system using newspaper stories might need indices that explain the events occurring in the stories. For some text corpuses and some tasks, it may be possible for an NLP-intensive approach to succeed, but we should understand that barring significant breakthroughs in NLP these will be the exceptional rather than the typical applications of textual CBR, and that such systems will by necessity be natural language systems first and case-based reasoners second.

Underestimating IR: Historically, information retrieval and artificial intelligence have been seen as making competing claims about what can be done with corpuses of text, with each disparaging the techniques of the other. This would be an unfortunate trap for textual CBR to fall into. A thorough grounding in IR methods and results should be considered fundamental training for anyone interested in textual CBR. For example, many of the obvious techniques for improving the term-level representation of a document have already been well-investigated in IR: detection of acronyms, proper names, phrase recognition, etc.

with results that are not always intuitive. It would be foolish for us to reinvent the wheel or, worse, the Edsel.

Overestimating IR: On the other hand, there is an equally-problematic trap to be found in ignoring the fundamental task and problem-solving orientation that CBR brings to the retrieval question. Information retrieval research finds little significance in targeted or "ad-hoc" systems that sacrifice generality in seeking to improve performance on a particular corpus, yet this is precisely where textual CBR's benefits are to be found. It is important therefore to understand the limits of the applicability of some IR results to research in textual CBR.

Evaluation methodology is one area in particular where CBR and IR are destined to clash. Information retrieval metrics, particularly the standard dyad of recall and precision, carry with them fundamental assumptions about the nature and purpose of retrieval. In the case of FAQ FINDER, a null result (no answer in the FAQ file) is potentially of great importance: the system may be able to route the question to an individual or to an appropriate newsgroup for answering. The standard metrics do nothing to help us assess this aspect of the system's performance because the IR metrics are based on the notion of relevance (finding related material), not the notion of problem-solving (answering a specific question).

Conclusion

The management and recall of text is an area where CBR techniques have already shown promise, and where the natural strengths of the CBR approach fit the problems encountered. I believe the problem-solving orientation of CBR will remain its biggest asset in building useful systems. To make full use of this asset we will need to develop our understanding of the kind of problem-solving processes that our systems might serve and what support these processes need from a textual case base. The taxonomy given above is a first step in this direction.

References

Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N., and Schoenberg, S. (1997) Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. *AI Magazine*, 18(2), pp. 57-66.

Burke, R., Hammond, K., and Young, B. (1997) The FindMe Approach to Assisted Browsing. *IEEE Expert*, 12(4), pp. 32-40.

Kulyukin, V. in preparation. Building Organization-Embedded Question-Answering Systems. PhD thesis. University of Chicago.

McDermott, D. (1981) Artificial Intelligence Meets Natural Stupidity. In J. Haugeland, ed. *Mind Design*, pp. 143-160.

Miller, G. A. (1995) WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11).

Riesbeck, C. (1997) What Next? The Future of CBR in Post-Modern AI. In D. Leake, ed. *Case-Based Reasoning: Experiences, Lessons and Future Directions*, pp. 371-388.