Applying Textual Case-based Reasoning and Information Extraction in Lessons Learned Systems

Kevin D. Ashley

Graduate Program In Intelligent Systems
University of Pittsburgh Learning Research and Development Center
3939 O'Hara Street
Pittsburgh, Pennsylvania 15260
ashley+@pitt.edu

Abstract

Textual Case-Based Reasoning and Information Extraction may assist in constructing Lessons Learned Systems where the lessons are texts. For a particular lesson domain, developers first should identify the kinds of information needed to compare lessons. Information Extraction techniques may then be applied in at least three ways to help extract such information automatically from lesson texts.

Introduction

A Lessons Learned System (LLS) makes available an organization's previously learned lessons so that decision makers can use them to guide the analysis of new situations. It should assist users to: (1) retrieve relevant previous lessons, (2) assess whether a lesson can be applied in the current circumstances, and (3) adapt a lesson to the current problem's differences.

The belief that lessons learned are a valuable organizational resource is inherently case-based. Case-Based Reasoning (CBR) means comparing a problem to prior cases in order to draw conclusions about the problem and guide decision-making (Ashley 1992; Kolodner 1995). Treating lessons learned as the cases, CBR may be helpful if it is reasonable to assume that previously learned lessons can guide decision-making in similar circumstances.

Determining similarity is crucial. A decision-maker must assess whether the new circumstances are relevantly similar. She must be able to compare the factual contexts of the past case and current problem to decide whether the lesson is feasible and appropriate in the new circumstances and how to adapt it to account for any important differences.

In this respect, the record of a past lesson must include enough information for a decision-maker to assess relevance, application and adaptability. Normally, this includes a description of the past situation's facts, the problem presented, the successful solution to the problem, any failures encountered in finding or implementing the solution, and an explanation of the successes or failures.

While an accurate description of a past situation may include any kind of data, text is often the most natural medium for describing the facts, problem, solution, and explanation. Similarly, current scenarios are most likely to be described in textual terms.

Automating Lessons Learned Systems

The simplest kind of Lessons Learned "System" is a notebook of textual lessons. The lessons may be indexed by abstract characterizations of the lesson or its factual scenario. A human reader must select candidate lessons from the index. Applying the lesson requires a decision-maker to read the lesson, compare it to the current problem, assess whether it really is relevant and adapt its lesson appropriately.

Web-based and Information Retrieval (IR) techniques may automate the retrieval process. As lessons increase in number and complexity, they may be stored in an on-line database accessible via a website. The subject-matter index may become an indexed list of hypertext links to individual lessons. In a full-text IR system, the lesson texts could be indexed by every important word in an inverted index. Texts could be retrieved through key word searches or natural language queries, searched and highlighted for key search terms. Document relevance would be assessed in terms of standard IR measures: satisfaction of boolean queries, vector distance or probabilities (Turtle 1995, 17).

Role of Textual CBR

Depending on the size of the lesson database, the complexity of the lessons and how decision-makers will use them, an IR approach to automating the retrieval task in an LLS may not suffice. IR relevance measures may not be powerful enough to find relevant cases. An IR approach cannot assess whether

a case applies or how to adapt it.

An AI/CBR technique may automate retrieval with a more knowledge-intensive relevance measure, one which directly relates to the problem-solving tasks users need cases to perform. CBR relevance criteria relate to the domain-specific features with which to compare and adapt cases to problem situations. Using these criteria, the system could help assess applicability and assist with adaptation.

To apply CBR, one must develop a framework for representing the case features that are important for comparing and adapting cases. For convenience, I will refer to this as the Case Representation Framework (CRF). The CRF enables a computer program to compare cases for purposes of assessing relevance, applicability, and adaptation.

Factors, a device for representing important features of legal cases (Aleven 1997; Ashley and Aleven 1997), may have some utility in capturing a lesson learned. Factors represent stereotypical patterns of facts in a case that strengthen or weaken an argument in favor of its conclusion. Factors may correspond to the relative strengths and weaknesses, benefits and costs, or pluses and minuses of following one plan versus an alternative. Where, as often happens, a case involves competing factors, a successful decision in the case is some evidence that the strengths outweigh the weaknesses in similar contexts. That may be the "lesson" of the case. Assessing whether it applies in a current problem involves determining whether the same set of conflicting factors is present, whether some factors are missing or new factors are present and their significance. One must also consider background changes since the previous lesson.

The traditional CBR approach is problematic, however. To apply the relevance criteria, cases must be represented in the Case Representation Framework so that the program can perform comparisons. This means that the textual lessons must be translated or mapped into the CRF. Lacking general AI techniques for understanding natural language, case entry usually is a time-consuming, manual process and a serious knowledge acquisition bottleneck. As a result, the case databases of traditional CBR programs are tiny compared to IR databases. Similarly, one needs to represent new problem scenarios in the CRF.

Research in a new sub-field, Textual CBR (TCBR), has addressed the challenges of a subset of CBR domains where cases are texts (i.e., textual cases) (Ashley and Lenz 1998). Programs are under development automatically to: (1) assign indices to textual cases, (2) retrieve relevant cases, and (3) extract or highlight relevant passages in textual cases.

TCBR may assist LLS developers to: (1) assign indices to lessons automatically, (2) retrieve lessons with relevance criteria directly related to how a lesson may solve a new problem, and (3) highlight parts of a lesson that are most relevant for assessing whether it applies and how to adapt it.

Significantly, TCBR techniques do not obviate the need for a Case Representation Framework. While some TCBR retrieval techniques do not require a highly elaborated CRF (e.g., Burke, et al. 1997[1]; Lenz and Burkhard 1997), systems that learn automatically to index or highlight cases still do require a CRF, both to process texts and reason with cases.

An important initial question for LLS designers, then, is what kind of Case Representation Framework their particular domain/task requires. How do/should decision-makers compare the current problem with past lesson scenarios to decide whether a lesson applies and how to adapt it? On what kinds of features do the comparisons turn? Answering these questions will help in designing appropriate CRFs and should be a major focus of the Workshop.

Role of Information Extraction

Once an LLS designer has settled upon a Case Representation Framework, the kind of information to be extracted from the textual lessons becomes clear. For purposes of discussion, let us assume that factors are part of the CRF.

Recent work on Information Extraction (IE) may be useful. IE comprises knowledge-based techniques for extracting information from a corpus of texts within a particular domain of discourse (Riloff 1996; 1993). Typically, IE programs apply a dictionary of concept nodes, specially-generated from the corpus, to extract information from new texts. Each concept node is a frame of information. Triggered by the appearance of a particular word in a new text, the concept node tells the IE program which additional features to look for in the text. For instance, a concept

[1] FAQ FINDER (Burke, et al. 1997) is a natural language question-answering system based on USENET files of frequently-asked questions. Given a query, it retrieves the most similar Q/A pair from the most appropriate FAQ file. Since lessons learned and FAQs may be similar, FAQ FINDER may be a useful prototype for LLSs.

node named "target-subject-passive-verb-kidnapped" is triggered when the IE program encounters the term "kidnapped" in a passive voice construction, as in "A U.S. diplomat was kidnapped by FMLN guerillas today." (Riloff 1993, 812). The concept node frame tells the program to look for a sentence subject of the class human, the person who was kidnapped, and a prepositional phrase referring to the kidnappers.

Recently, the all-important concept dictionaries can be produced automatically, or semi-automatically. For example, AutoSlog-TS can identify an ordered list of short phrases that are characteristic of a corpus of relevant documents from which a human can select the best phrases for adding to the dictionary (Riloff 1996).

The question is whether IE can extract from a textual lesson data to fill out a Case Representation Framework. Can it extract factors? Factors are sentence-length concepts more complex than those that IE has succeeded in extracting. The lesson texts, although not as complex as, say, legal opinions, probably are more complex than newspaper articles, the typical texts to which IE has been applied.

In work on the SMILE program (Smart Index LEarner) we apply AutoSlog-TS to help the program learn to extract information about particular factors from textual cases. SMILE learns to assign factors to legal opinions involving trade secret law. It employs a machine learning algorithm, ID3, to learn decision trees for making the assignments.

The SPIRE program (Daniels and Rissland 1997) takes a different approach to a similar task, highlighting factor-like descriptions in texts.

In SMILE, the training instances come from factual case summaries in which factors have been identified manually. Sentences from which it may be directly inferred that a factor applies are positive instances. All other sentences are negative instances. An example of one of the factors is F15, Unique-Product. A plaintiff's claim for trade secret misappropriation is strengthened to the extent that the "Plaintiff was the only manufacturer making the product." Figure 1 shows four positive instances of Factor F15.

ID3 learns a decision tree for a factor by recursively partitioning the training set to best discriminate the positive instances like those in Figure 1 and the negative instances.

For this factor, SMILE induced the decision tree in Figure 2. The decision tree summarizes a set of rules for classifying sentences: If the term "unique " is present then F15 applies; else if "known" and "general" are present then F15 applies; else if "capable " is present....

1. [Plaintiff] introduced evidence that [plaintiff's product] was a unique product in the industry. (from *Innovative v. Bowen*)
2. It appears that one could not order [plaintiff's product] in any establishment other than that of the plaintiff. (from *Mason v. Jack Daniel Distillery*)
3. The information in the diagram is not generally known to the public nor to any of [the plaintiff's] competitors. (from *Tri-Tron v. Velto*)
4. Several features of the process were entirely unique in [product-type] manufacturing. (from *Sperry Rand v. Rothlein*)

Figure 1: Instances of Factor F15, Unique-Product

We have confirmed empirically that SMILE's decision trees do a better job of assigning factors than two reasonably intelligent baseline procedures (Brüninghaus and Ashley 1999). Nevertheless, there is room for improvement. For instance, SMILE does not yet take negation into account. A product is unique if it is *not* generally known; the decision tree in Figure 2 does not make that distinction.

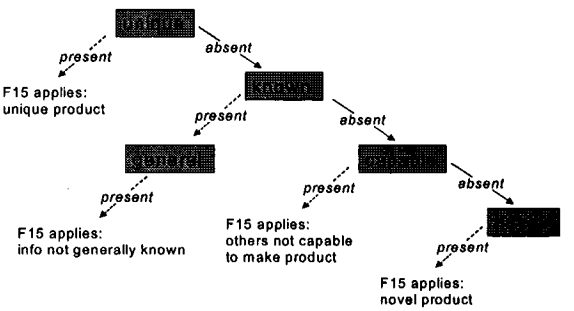

Figure 2: SMILE's Decision Tree for Factor F15

To improve the process of inducing decision trees, we plan to use Information Extraction in three ways.

First, we attempt to select the short phrases that may be the most effective concepts for ID3 to employ in discriminating positive and negative instances of the factors. We apply AutoSlog-TS to two collections: trade secret case texts where the relevant factors are discussed and other, irrelevant cases. AutoSlog-TS identifies the phrases most characteristic of the relevant documents.

Second, we employ IE to extract from the cases certain data associated with typical fact patterns. This information can improve decision trees and help

control the complexity of learning. In a trade secret case, the plaintiff usually has developed some confidential product information and sues the defendant for using that information to develop a competing product. It is feasible to extract certain role-playing information from such patterns, including the parties' names, their roles in the lawsuit (i.e., plaintiff or defendant), the names of their competing products, and possibly their product types.

In the training instances, one can then substitute the more general role information for the specific names of parties, products, and product-types. This substitution makes the training examples more abstract and useful. By referring to role-playing concepts like "plaintiff", "defendant", "plaintiff's product", and "product-type", the decision trees may better discriminate positive and negative instances. Instead of referring to specific names, unlikely to appear in more than one case, the instances refer to generic role-playing concepts, likely to appear over a much wider range of cases. For example, in each of the instances of factor F15, Unique-Product, shown in Figure 1, I manually substituted role-playing information for specific names. Originally, sentence 1 from the case of *Innovative v. Bowen* stated, "Innovative introduced evidence that Panl Brick was a unique product in the industry." Decision trees learned from more general examples like these will better discriminate positive from negative instances. The tree can relate the role information "plaintiff's product" with "unique" to better capture the pattern of concepts associated with the factor.

Third, the extracted information may focus automatic text-processing on the most relevant parts of a text. Parsing texts yields certain linguistic information (e.g., about negation and phrases) that would help SMILE discriminate positive and negative instances of a factor. On the other hand, parsing generates too many features and renders the learning task too complex. It would help if the program could select the most productive sentences to parse. When a textual case refers to the parties and products by name, it is more likely that the court is discussing the case's specific facts. These parts of the text are more likely to identify information from which it can be inferred whether factors apply.

Conclusion

Textual Case-Based Reasoning programs like SMILE apply Information Extraction techniques to extract complex information like factors, which may be useful for comparing lessons. The IE techniques can help in selecting representative phrases, extracting information about typical fact patterns, and directing more intense language processing to the most fruitful parts of a text. These techniques may help construct Lessons Learned Systems where the lessons are texts. Developers, however, must first identify the kinds of information needed to compare lessons.

References

Aleven, V. (1997) *Teaching Case-Based Argumentation through a Model and Examples*, Ph.D. Dis. Intelligent Systems Prog., U. Pittsburgh. www.cs.cmu.edu/~aleven

Ashley, K. D. (1992) Case-Based Reasoning and its Implications for Legal Expert Systems". In *Artificial Intelligence and Law.* Vol 1, No. 2, pp. 113-208. Kluwer. Dordrecht, Netherlands.

Ashley, K. and V. Aleven. (1997) Reasoning Symbolically About Partially Matched Cases. *Int'l Joint Conf. on AI*, IJCAI-97. 335-341. Morgan Kaufmann: San Francisco.

Ashley, K.D. and Lenz, M. (eds.) (1998) *Textual Case-Based Reasoning, Papers from the AAAI-98 Workshop*, AAAI Tech. Rep. WS-98-12 AAAI Press, Menlo Park.

Brüninghaus, S. and K.D. Ashley (1999a) Bootstrapping Case Base Development with Annotated Case Summaries. In *CBR Research and Development: Proc. 3d Int'l Conf. on CBR.* 59-73. Lecture Notes in AI 1650. Springer: Berlin. www.pitt.edu/~steffi/papers/iccbr99.ps

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 57-66.

Daniels, J.J. and Rissland, E.L (1997) What you saw is what you want: Using cases to seed information retrieval, *Proc. 2d Int'l Conf. on CBR CBR Research and Development* ICCBR-97. 325-336. Lecture Notes in AI Series No. 1266. Springer: Berlin.

Lenz, M. and Burkhard, H.-D. (1997) CBR for Document Retrieval: The FAllQ Project In *CBR Res. and Devel.: Proc. 2d Int'l Conf. on CBR,*. ICCBR-97. 84-93. Lecture Notes in AI Series No. 1266. Springer: Berlin.

Kolodner, J. (1995) *Case-Based Reasoning* Morgan Kaufmann: San Mateo, CA.

Riloff, E. (1996) Automatically Generating Extraction Patterns from Untagged Text, *Proc. 13th Nat'l Conf. on AI*, 1044-1049, AAAI Press/MIT Press. Menlo Park, CA.

Riloff, E. (1993) Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proc. 11th Nat'l Conf. on AI*, 811-816. AAAI /MIT Press. Menlo Park.

Turtle, H.R. (1995) "Text Retrieval in the Legal World" in *Artificial Intelligence and Law*, 3: 5-54. Kluwer: Dordrect.