Heuristic Harvesting of Information for Case-Based Argument

Edwina L. Rissland, David B. Skalak, and M. Timur Friedman
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003
{rissland, skalak, friedman}@cs.umass.edu

Abstract

The BankXX system models the process of perusing and gathering information for argument as a heuristic best-first search for relevant cases, theories, and other domain-specific information. As BankXX searches its heterogeneous and highly interconnected network of domain knowledge, information is incrementally analyzed and amalgamated into a dozen desirable ingredients for argument (called *argument pieces*), such as citations to cases, applications of legal theories, and references to prototypical factual scenarios. At the conclusion of the search, BankXX outputs the set of argument pieces filled with harvested material relevant to the input problem situation.

This research explores the appropriateness of the search paradigm as a framework for harvesting and mining information needed to make legal arguments. We discuss how we tackled the problem of evaluation of BankXX from both the case-based reasoning (CBR) and task-performance perspectives. In particular, we discuss how various system parameters—start node, evaluation function, resource limit—affected BankXX from the CBR perspective and how well BankXX performs its assigned task of gathering information useful for legal argumentation by running BankXX on real legal cases and comparing its output with the published court opinions for those cases.

1. Introduction

In this paper we discuss heuristically-guided perusal, analysis, and harvesting of information for use in case-based argument. The specific task we study is the gathering and amalgamating of key pieces of information—called argument pieces—which serve as basic building blocks for argument; this is a prerequisite task to the actual generation of more familiar forms of argument, such as 3-ply arguments [Ashley, 1990] or structured memoranda [Rissland & Skalak, 1991; Branting, 1991; Rissland et al., 1993].

We are particularly interested in this task in realistic situations, such as performing scholarly research in a large well-stocked library with a vast body of potentially relevant materials. In such contexts, multiple retrievals and evaluations of relevance are necessary as one explores materials and builds up an analysis. One often pursues a two-step winnowing approach: gather up potentially useful information as one chases references and then analyze further only those materials with the potential to contribute significantly to key aspects of one's research. In this paper, we explore this view in our BankXX program, which models the process of gathering and analyzing information for the task of creating an argument with the classic paradigm of heuristic best-first search. BankXX's domain concerns an aspect of federal bankruptcy law for individual debtors, specifically the "good faith" requirement for Chapter 13 repayment plans.[1]

A major theme of this paper is that in the face of too much information and too few resources to examine it all, an intelligent information gatherer—person or program—must make choices about which leads to follow, which to ignore or postpone, which information to study closely, which to peruse superficially. A second theme is that the value of some information is not necessarily apparent on first perusal and that some information might not even be accessible on the initial foray into the knowledge base; that is, relevance and accessibility can change as more information is uncovered and examined. For instance, a seemingly unimportant case—one not most on-point or even highly similar—can gain in importance if it turns out that it is the only case addressing a key issue.

In this paper, we present our evaluation of how well BankXX performs on its assigned task of harvesting information useful for legal argumentation (on the "good faith" issue) by running BankXX on real legal cases and comparing its output with the published court opinions for those cases. We also evaluate how various internal parameters affect BankXX's performance as a CBR program. Finally we note that although the context of our research is legal argument, we believe that the use of such heuristic retrieval methods is applicable in other areas, such as diagnosis and design, where CBR methods have been classically applied.

Some of the points we will touch upon in this paper are:

This work was supported in part by the Air Force Office of Scientific Research under contract 90-0359.

[1] Section 1325(a)(3) of the federal bankruptcy code requires that debtors' plans be proposed in "good faith." This term is undefined in the statute and is interpreted in individual cases by application of case law.

• The overall process of gathering information for an argument can be usefully modeled as heuristic search.

• *Argument pieces* are used to represent argument and define an evaluation function.

• Retrieval of cases and other knowledge can fruitfully be done with a combination of knowledge-based indexing and heuristic search.

• Retrieval and relevancy assessment is carried out over many iterations.

This paper first discusses the BankXX system generally and exposits how the heuristic search model is embodied in BankXX. In particular, we describe the argument piece model of argument used in BankXX and how heuristic search and evaluation functions drive the program to instantiate it. Once all these pieces are in place, we describe a series of experiments to gauge the performance of the system. We conclude by discussing the contributions of this research.

2. Background

BankXX has all the standard ingredients of CBR—case memory, indexing, case selection metrics—and executes all the standard subtasks—case input and analysis, retrieval, selection, task performance—on the paradigm CBR performance task of creating precedent-based arguments ([Ashley, 1990; Branting, 1991]). BankXX's case memory is structured as a directed graph whose nodes represent legal objects such as cases and legal theories and whose arcs represent their interconnections. Case memory in BankXX is highly associative, heterogeneous, and has a large branching factor. The indices used by BankXX are arcs of the case memory network as well as more complex linkages computed from them. There are multiple paths to most items in memory [Kolodner, 1983; Turner, 1988]. Evaluation functions are used for case selection.

Kolodner [1993, p. 291] discusses retrieval techniques in terms of memory data structures and retrieval algorithms, such as flat memory with serial or parallel search, shared feature networks with breadth-first graph search, prioritized discrimination networks with depth-first graph search, and redundant discrimination networks with breath-first graph search. To this list we can add BankXX's highly-interconnected, heterogeneous semantic networks with heuristic best-first search.

BankXX's application of iterative, heuristic search to case retrieval distinguishes it from many other CBR programs. CBR is in general a heuristic process because its key mechanisms—indices, relevancy metrics, etc.—are heuristic; and CBR programs do implicitly search a case space. However, CBR programs are not usually designed in the classic search paradigm with its **iterative** control cycle of evaluating, selecting, and expanding possibilities on the problem-solving horizon. Most CBR programs do not repeatedly cycle through the retrieval and selection of cases. One exception is Owens's [1993] ANON, which iteratively integrates the search for cases and the extraction of discriminatory features. Direct Memory Access Parsing (DMAP, [Martin, 1990]) uses a semantic network of case frames that is searched via a marker-passing algorithm. Other CBR systems that use search include Branting's GREBE [1991] (A* search to match structured cases preparatory to making an argument), Alterman's PLEXUS [1988] (search of abstraction hierarchies for substitute plan steps for adaptation), and Kass and Leake's SWALE [1988] (local search of plan repair —"tweak"—hierarchy).

This research also develops a new methodology to evaluate precedent-based argument, and presents a new application of the precision-recall measures used in information retrieval to evaluate case retrieval. We describe an extensive series of experiments that examine BankXX's performance under various parameter settings, and compared to previous programs and actual legal opinions.

3. Argument Pieces: The Components of Argument

Our model of argument is based on the belief that one has a good general sense of the type of information needed to flesh out a good argument, even if one is not an expert in the particular area. This general knowledge includes:

1. what types of domain knowledge exist (e.g., cases, theories, prototypes) and how they are interconnected (e.g., intercase citation, case to theory pointers); and

2. what basic building blocks are needed to put together a good argument (e.g., a viable legal theory, favorable supporting cases, understanding of contrary cases).

Based on these observations, we have chosen a simple representation of an argument for purposes of this research as a collection of *argument pieces*. These building blocks of argument represent fragments of arguments or pieces of legal knowledge that an advocate would ideally like to have when making an argument. We recognize that this idealization of argument does not reflect the logical and rhetorical connections between the various pieces of an argument, or the complexity of argument in general. However, we feel that such information is a prerequisite for constructing more elaborate representations of argument. In BankXX argument pieces are used in two ways: (1) to represent argument, and (2) to define a heuristic evaluation function.

The 12 argument pieces currently used in BankXX are:

1. Supporting Cases - cases decided for the same "side" as the viewpoint assumed in the current problem situation.

2. Best Supporting Cases - the best cases decided for the current viewpoint.

3. Contrary Cases - cases decided for the opposing side.

4. Best Contrary Cases - defined similarly to 2.

5. Leading Cases - one of the five most frequently cited cases in our BankXX corpus. These were found by a frequency analysis of cases citations done on the full text of the opinions of the cases in the BankXX case base.

6. Supporting Citations - citations that: (i) are found in cases with the desired viewpoint, (ii) point to other cases with the same viewpoint, and (iii) use a "citation signal" indicating that the citing case agrees with the cited case (e.g., *accord*, *see*).

7. Factor Analysis - the set of domain factors ("dimensions") that are applicable to the current problem situation.

8. Overlapping Cases - cases sharing a large proportion of domain factors, where large in this paper means at least 75%.

9. Applicable Legal Theories - If each factor defining a theory is applicable to the problem situation, the theory is considered applicable.

10. Nearly Applicable Legal Theories - A theory is nearly applicable if a threshold percentage of defining factors apply. In this paper, the threshold is set at 50%.

11. Factual Prototype Story category - the category of story (e.g., student loan case, medical calamity) that the debtor's factual situation falls under. These categories were assigned by hand.

12. Family Resemblance Prototype - cases decided for the desired viewpoint having the highest family resemblance rating, with respect to a given family, to the instant case according to the Rosch measure of family resemblance [Rosch & Mervis, 1975].

Each of these argument pieces is defined computationally in BankXX. The definitions of "most on-point," "best," and domain "factor" are based directly on those used first in HYPO [Ashley, 1990] and then in CABARET, occasionally with some modification.[2] For each argument piece, there is a "functional predicate" that determines if a node can supply that useful piece of an argument and a data structure containing an object slot to store entities that satisfy its predicate. BankXX builds up their content incrementally (as its search proceeds) and the collection of filled argument pieces is output to the user at the conclusion of BankXX's processing.

4. The Heuristic Search Model in BankXX

BankXX builds up the content of the argument pieces by performing heuristic best-first search in a network of domain knowledge. BankXX always begins its processing by analyzing the problem situation for applicable domain factors and computing a claim lattice, which partially orders the cases that have some of the same factors at work as the current problem. The best and most on-point cases are identified. These provide potential new nodes to be explored and are always the first nodes to be placed on the open list.

[2]For instance, the "best" cases in BankXX are most similar to the problem situation whether or not they are most on-point. This is different from the definition in HYPO.

BankXX continues by performing the standard cycle of iterative, best-first search. Neighbors of the current node are generated using BankXX's neighbor methods. The "best" node on the open list—one with the maximum value under one of BankXX's evaluation functions—is identified and is then examined by each of the argument pieces in turn in order to determine if it can contribute to that component of the argument. Information that can be harvested by the argument pieces is appended to their data structures. This cycle continues until the search exceeds a user-specified time or space bound (e.g., 30 nodes closed), or until the open list is empty. At the conclusion of the search, the argument is output in a template containing the argument pieces, which have been incrementally filled during the search.

4.1 The Case-Domain Graph

State-space search is defined by a triple: *initial state, set of operators on states, set of goal states.* In best-first search, an *evaluation function* is used to guide the exploration of the state-space [Barr et al., 1981]. We begin by describing the search space.

The Search Space. In BankXX it consists of a semantic network whose nodes represent cases and legal theories from the application domain, the "good faith" issue for personal Chapter 13 bankruptcy plans. Labeled links represent their interconnections. We refer to this network as the *case-domain-graph.* There are six types of case-domain-graph nodes: five represent legal cases in various perspectives proven useful to human legal reasoners and one represents legal theories. In this area of the law, appeals courts often articulate approaches—"theories"—for dealing with the good faith question; these are typically described in terms of domain factors. The five ways legal cases are represented are (1) as factual situations, (2) as bundles of citations, (3) as stereotypical stories or scripts, (4) as a collection of legal factors, or (5) by the measure of their prototypicality. Cases of like type can be grouped into *spaces*: (e.g., Case Citation Space, Legal Theory Space). Each space captures a particular type of knowledge and its natural interconnections. For instance, intercase citations are captured in the Citation Space and legal theories and relationships between them (e.g., refinement) in the Legal Theory Space. All the spaces are interconnected. For instance, cases point to the legal theories that are applied in them and the story prototype they fall under. Thus the case graph is highly interconnected. For more details see [Rissland, Skalak & Friedman, 1993].

The Start Node. In BankXX the default for the initial state is the user-supplied problem situation, which is represented using the same set of hierarchical frames used to represent a case as a collection of facts. Alternatively, the user can input the problem case but specify another node as the start node, for instance, a favorite or well-known case, in order to concentrate the search initially in a particular region of the space.

The Operators. The set of operators used in BankXX are called *neighbor methods*. These use links in the case-

domain-graph to generate the "successor" nodes to be opened in search. Some follow in-space or cross-space pointers in a straightforward way. For instance, *case-theory-neighbors* generates all the cases that have applied a particular theory. Others, similar to macro-operators, follow a fixed sequence of links. For instance *theory-case-theory-neighbors* finds all the theories applied by any of the cases that use the theory used in the current node. BankXX has 12 neighbor methods. In general, they are more complex than the simple following of outward arcs from a given node.

Goal Nodes. We do not provide goal states to BankXX because of the difficulties inherent in defining an "argument goal" in a way that is consistent with our understanding of how humans develop and evaluate legal arguments. It is hard to say in general that an argument does or does not meet some plausible persuasive or rhetorical goal, or even that one has completed the supporting research.

4.2 Evaluation Functions

We have experimented thus far with three different types of evaluation functions. They differ in the level of abstraction that they use to evaluate nodes in the case-domain graph. All of the evaluation functions are simple linear functions. They form a progression of increasingly more informed evaluation methods, whose considerations range from (1) only the type of information encoded in a node to (2) the contribution of the node to the standard argument pieces and (3) the incremental impact of a node on the overall state of the evolving argument.

In this paper we concentrate on experiments using only the first two. Briefly, they are:

(1) Node-type evaluation function. Its form is:

$$w_1\ type\text{-}pred_1(c) + w_2\ type\text{-}pred_2(c) + \ldots + w_n\ type\text{-}pred_n(c)$$

where *type-pred* checks the *type* of the current node c. This function assesses the potential according to pre-assigned estimations of how useful various types of nodes are. It causes node-types to be examined in the order defined by the weights w_i. In these experiments legal theories have some preference but there is not much difference among the others.[3]

(2) Argument piece evaluation function. The form of this function is:

$$w_1\ arg\text{-}piece\text{-}pred_1(c,a) + w_2\ arg\text{-}piece\text{-}pred_2(c,a) + \ldots + w_n\ arg\text{-}piece\text{-}pred_n(c,a)$$

where c is the current node and a is the current state of the argument. Each *arg-piece-pred* computes whether a particular argument piece is fillable by the current node and if that argument piece has not already been completely filled: if so, it returns 1; else, 0. This evaluation function prevents BankXX from wasting computing resources by unnecessarily bolstering parts of the argument that are already well-established.[4]

5. The BankXX Experiments

In this paper we report on two types of empirical evaluations:

1. comparing the performance of BankXX with itself as a CBR program, by varying parameter settings; and

2. comparing the performance of BankXX with hand-coded arguments found in opinions of actual court cases.

In other experiments, we further explore BankXX's performance.

5.1 Methodology

The methodology for the first set of experiments is straightforward: run BankXX on each of the 54 cases in its case base in a *de novo* manner—that is, excise the case and all its linkages from BankXX's case-domain-graph—and count the number of items filling each of 10 argument pieces.[5] To compare BankXX with written case opinions, we encoded the 54 opinions into "answer" keys comparable in form to those generated by BankXX and applied standard precision and recall measures.

Precision is the ratio of what was mentioned by both the decision and BankXX to that mentioned just by BankXX. **Recall** is the ratio of what was mentioned by both the decision and BankXX to that mentioned just by the decision. We hasten to add that given the small numbers used in these experiments, these measures are very sensitive to small changes. For instance, for a given argument piece, if BankXX retrieves one item that is one of only two items mentioned in the opinion, its precision is 100% and recall is 50%. Should BankXX retrieve an "extra" item not mentioned in the opinion, its precision will drop to 50%; two extra items drop precision to 33%. Its recall will not increase. Since BankXX diligently harvests as much information as it can, it is likely to mention more items than the opinion and be penalized for it in precision and not get credit for it in recall. Thus, one should be careful in reading too much into these traditional metrics. Nonetheless, given their widespread use, we do use them here.

Creating the "answers" needed for precision-recall comparisons was done by reading the court's opinion and

[3]The weights are 8 (theories), 6 (cases), 5 (citations), 4 (domain factors), and 3 (factual prototypes).

[4]The weights and the limits on the number of items considered to fill each argument piece (given in brackets) are: 2 [3] (supporting cases), 7 [5] (best supporting cases), 1 [3] (contrary cases), 5 [3] (best contrary cases), 6 [5] (leading cases), 1 [5] (supporting citations), 1 [5] (overlapping cases), 8 [6] (applicable legal theories), 6 [3] (nearly applicable theories), and 6 [1] (factual prototype stories).

[5]There are 10 terms whereas there are 12 argument pieces because the factor analysis argument piece is filled during system initialization, and we do not use the family resemblance prototype argument piece in these experiments.

encoding each case and theory actually cited in the opinion. One problem inherent in encoding written opinions with the set of original argument pieces is how to identify elements fitting each argument piece, since some have technical BankXX meanings (e.g., best case) or make fine distinctions hard for human readers to discern (e.g., applicable versus nearly applicable legal theory, best versus merely supporting cases). In BankXX, these distinctions are made in a principled way with computational definitions. To compensate for such difficulties, the argument pieces were aggregated into four larger-grained argument pieces that were easy to apply.[6] These were then used in hand-coding court opinions and as the basis of BankXX versus actual court performance comparisons. The four simplified argument pieces are: (1) **Cited-Supporting-Cases**,[7] (2) **Cited-Contrary-Cases**,[8] (3) **Cited-Leading-Cases**, and (4) **Cited-Legal-Theories**.[9]

With these aggregated argument pieces, hand-coding was straightforward and involved little subjective judgment. Any case cited in the opinion is listed as a **cited-supporting** case or a **cited-contrary** case depending on how its outcome compares with decision in the opinion.[10] If a cited case is also one that is frequently cited by written opinions in general,[11] it is also listed as a **cited-leading** case. If an opinion explicitly articulates a theory of its own, reiterates or applies the theory of another case, or appeals to a general domain theory (e.g., a "totality of the facts and circumstances" theory of good faith), then that theory is encoded as a **cited-legal-theory**.

Output from these BankXX-court comparison runs can be viewed in various ways. **Figure 1** displays graphically the finest-grained analysis. It shows results for retrieval of objects for the aggregated **cited-leading-cases** argument piece for each of the 54 cases. Each bar compares performance of BankXX with the court opinion on one case.

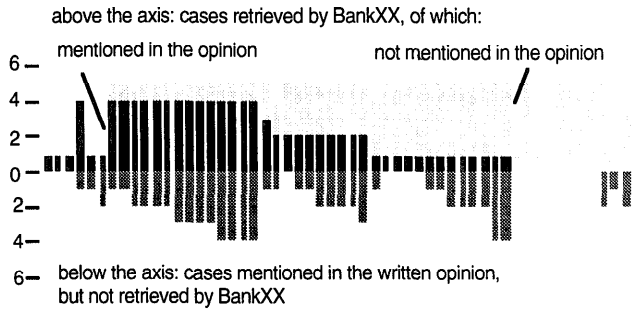

Figure 1: Comparison of retrieved *cited-leading-cases* using the argument piece evaluation function. Performance on each of the cases, in order from highest to lowest precision.

The vertical axis indicates the number of items retrieved. Everything above the zero represents items retrieved by BankXX with the black part of a bar representing those retrieved by BankXX and mentioned in the written opinion and the lightly shaded part of the bar representing items retrieved by BankXX that were not mentioned in the opinion. The darkly shaded part of the bar extending below zero represents items mentioned in the opinion that were not retrieved by BankXX. Graphically, **precision** is the proportion of black out of the total bar above the zero; **recall** is the proportion of black out of the combined black and darkly shaded parts of the bar.

In summary, we ran BankXX on each of the 54 cases in its case base in *de novo* fashion with each of two evaluation functions, and compared retrieval on each argument piece: approximately 1500 data points.[12]

5.2 BankXX as a CBR program

This section describes three experiments we performed to answer questions about BankXX as a case-based retrieval system:

1. How important is the initial query in determining the eventual outcome of retrieval?

2. How much knowledge must the case retrieval function have in order to be effective?

3. When can search terminate and the retrieval result be satisfactory?

As a baseline, BankXX was run with the *Estus* case, 695 F.2d. 311 (8th Cir. 1982), as start node, the argument piece evaluation function, and search limited to closing 30 nodes. We addressed the three questions above in search terms by examining the effects of:

1. varying the start node,

2. changing the evaluation function, and

3. trying different limits on the number of nodes that could be closed.

---

[6]Note five argument pieces are not used in the aggregated argument pieces: supporting-citations, factor-analysis, overlapping-cases, factual-prototype-category, family-resemblance-prototype.

[7]Defined for BankXX as the union of supporting-cases and best-supporting-cases.

[8]Defined for BankXX as the union of contrary-cases and best-contrary-cases.

[9]Defined for BankXX as the union of applicable-legal-theories and nearly-applicable-legal-theories.

[10]Complications, such as the fact that a same side case may have been cited (with a so-called *But see* citation signal) in order to differ with its rationale while still agreeing with its outcome, are overlooked.

[11]A frequency analysis was done on a corpus of cases of approximately 800 cases gathered with a WestLaw retrieval. We then checked the citation frequency of each of BankXX's cases in this larger corpus. The five most frequently cited cases were used to define cited-leading-case category applied to written opinions. By contrast, for BankXX leading-cases is defined with respect to frequency of citation within BankXX's own corpus.

[12]Given 10 argument pieces used in the general CBR experiments and 4 in the BankXX-Court comparisons, there are (2x10 + 2x4) x 54 data points.

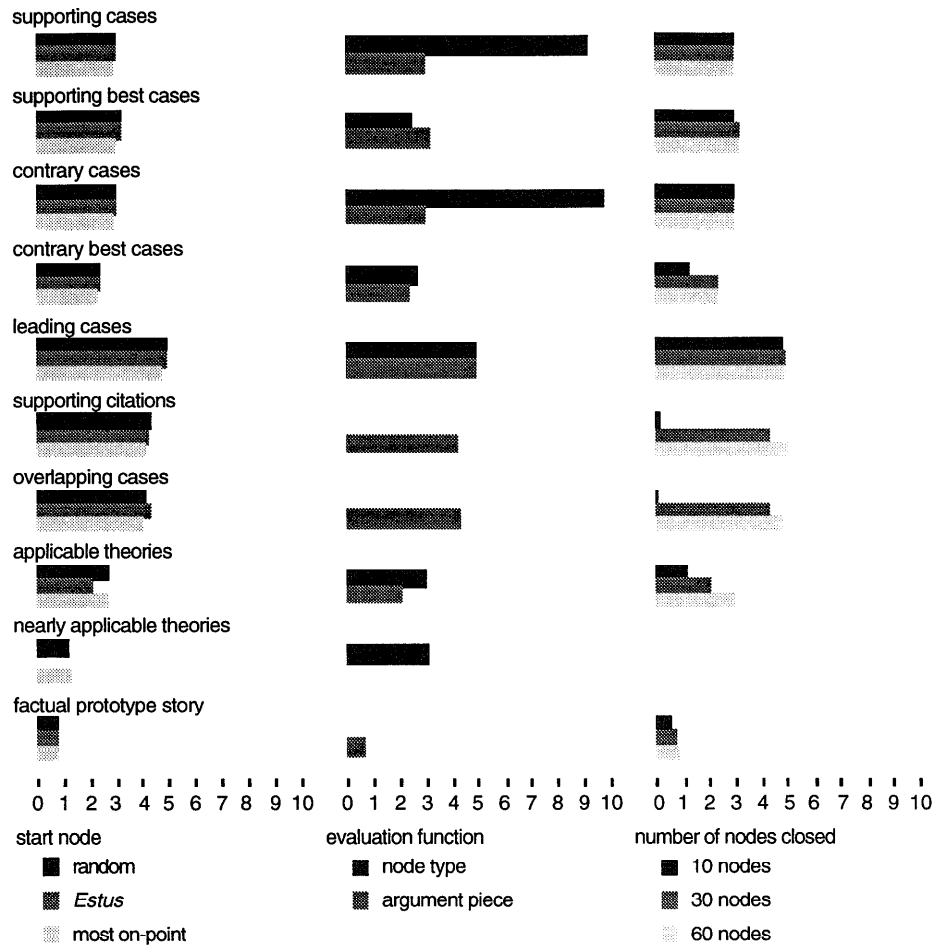

Figure 2: Average number of objects filling each argument piece as the start node is varied (left), the evaluation function is varied (middle), and the number of nodes closed is varied (right).

We ran BankXX *de novo* on all 54 cases in the case base to obtain averages for the number of objects filling each argument piece.[13]

5.2.1 Initial Query Formulation. Using the argument piece evaluation function and stopping search after closing 30 nodes, three different nodes were used as start nodes: a random case, the *Estus* case, and a most on-point case. The random case provides a base line. *Estus* is well known in this area of bankruptcy law—almost any research materials consulted by an attorney will soon lead to it—and therefore it may be considered a realistic and useful starting point. A most on-point case is another starting point likely to be relevant.

The results showed that the choice of start node, which is the initial query to the case base, made little difference to retrieval. As the left hand side of **Figure 2** shows, the average number of objects found for each argument piece is about the same for each of the three start nodes. We examined search paths through the case-domain graph to understand why. It turns out that no matter where search starts in this case-domain graph of 150 nodes, it soon leads to a highly interconnected region which contains many useful cases and theories. For example *Estus* and *Flygare* (another well known case) and the theories promulgated by these cases are part of this area of the graph. Informally speaking, it doesn't matter where search starts because in this domain all roads lead to *Estus*.

We conclude that in browsing a case base where there is a sense of location and a sufficiently rich indexing fabric, the initial probe to case-memory may not matter in a multiple-probe situation.

5.2.2 Case Retrieval Function. Next, we compared the effects of varying the evaluation function while keeping the 30 closed node limit and always starting at the *Estus* node. The node-type evaluation function finds more contrary cases and same side cases, but does so at the expense of failing to fill other argument pieces. See the middle of

[13]N.B., numbers of nodes closed, opened, and filling an argument piece are not the same. In general, many more nodes are opened than closed, and the total number of items filling the set of argument pieces exceeds the number of closed nodes (see Figure 2).

Figure 2. The node-type function uses only the type for each node and does not limit the number of objects retrieved for any argument piece. Considering its lack of knowledge, it does surprisingly well.

To understand how a knowledge-poor function can produce satisfactory results, one can consider search as just the first of a two-stage retrieval process for filling the argument pieces. The second stage applies the argument piece predicates to the retrieved objects to determine if they fulfill the requirements of the argument piece.

We conclude that in a two-phase retrieval, a knowledge-poor function to generate candidates in the first phase may be sufficient, as long as the performance criteria in the second phase are sufficiently rich. The efficacy of the classic generate-and-test or "many-are-called/few-are-chosen" (MAC/FAC) approach has been observed in other research as well [Gentner & Forbus, 1991].

5.2.3 Termination of Search of Case Memory. There is no objective standard for when one has completed research or completed an argument. Thus BankXX has two termination parameters that may be set by the user: limiting the time ("billable seconds") used and the number of nodes closed. In these experiments BankXX was set to terminate after it had closed 10, 30, and 60 nodes.

With the argument piece evaluation function and *Estus* as the start node, 10 nodes was too few to fill up many of the argument pieces. As a rough guide, 30 nodes seemed an appropriate compromise between more exhaustive search and too scanty an examination of the domain-graph. Incremental benefits of more search decreased after about 30 nodes. See the right hand side of Figure 2.

From the CBR perspective, we conclude that the decreased marginal utility of finding more cases causes there to be a point at which additional search of the case base is not effective. This conclusion echoes the results of Veloso and Carbonell [1991] as to the optimal amount of time to search a case base in a hybrid planner.

5.3 BankXX as an Argument Program

Using standard precision and recall measures, we compared the performance of BankXX with written judicial opinions. All 54 cases were run *de novo* with *Estus* as start node and a limit of 30 closed nodes. Results were averaged over the 54 cases.

5.3.1 Precision/Recall Performance and the Evaluation Functions. We were somewhat surprised to find that in general the knowledge-poor node-level evaluation function usually exhibited higher recall and precision than the knowledge-richer argument piece function. For instance, all the average recall values for the node-type function lie above the corresponding values for the argument piece function. Averaged over the four simplified argument pieces, the node-type evaluation function gave higher recall (0.55), than the argument piece evaluation function (0.44). We conclude that BankXX's overall **recall** performance seems to depend more on the choice of evaluation function than the choice of argument piece. The node-type evaluation function may give higher recall simply because it retrieves more items than the argument piece function. See the middle of Figure 2. The argument piece evaluation function is more selective but pays a price for that in recall.

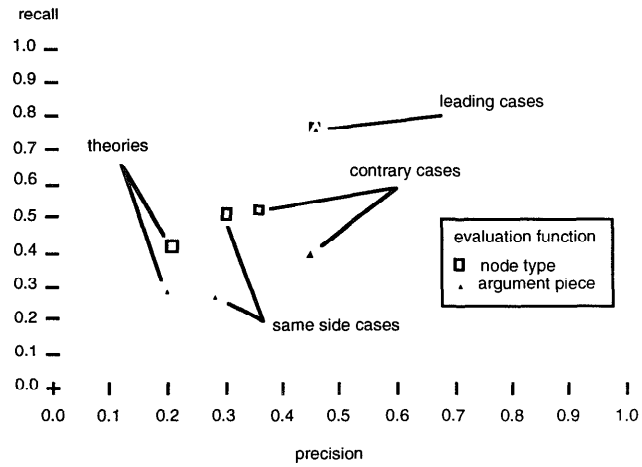

Figure 3: Average precision and recall (over all 54 cases) for the four aggregated argument pieces.

On the other hand, there seems not to be much difference in overall **precision** performance between the two evaluation functions. Each argument piece performs at about the same precision for each function. As we did in Section 5.2.2, we ascribe this to BankXX's two-stage approach: the lack of precision inherent in the node-type function is ameliorated by the precise filling of the argument pieces. Finally, we note that we did not observe the classical trade-offs between precision and recall. This might be because BankXX is not be functioning at a frontier where such phenomena occur or we need to vary other parameters to see them. In these studies, we only varied two, the evaluation function and the argument piece.

5.3.2 Recall/Precision and Argument Pieces. We observed differences in retrieval precision for the different argument pieces (see **Figure 3**). For both evaluation functions, highest precision was found for **cited-leading-cases** (0.46), followed by **cited-contrary-cases, cited-supporting-cases,** then **cited-legal-theories** (0.21). The results for recall were similar for the argument piece function. For the node-type function there was a flattening of performance differences among recall for the three argument pieces involving cases; all three did well.

We interpret the better precision on **cited-leading-cases** as follows. Since the same small group of leading cases are cited repeatedly in the opinions (that's what makes them leading cases), the probability that a given leading case is mentioned is higher than that for an arbitrary contrary or supporting case or legal theory. Thus if BankXX mentions a leading case it is likely to be in the opinion as well and hence BankXX's good precision marks on this argument piece.

For the other argument pieces, there is a wide range in the amount of information mentioned in the opinions. Thus if BankXX retrieves information not found in the opinions—which is likely to happen given BankXX's diligence in going after information—this lowers BankXX's precision. In particular, BankXX's low precision and recall scores on **cited-legal-theories** may be due to the high number of legal theories (18) relative to the number of cases (54), and the similarity of many theories. The program receives no credit for retrieving a useful but uncited theory in the absence of a metric to measure the similarity of the retrieved theory to the one actually applied by a court.

5.3.3 Precision-Recall Measures - Limitations. Again, let us note that the answers derived from actual opinions are not necessarily the best possible nor the only answers. Each opinion is the product of an individual judge and clerks. Some will cite many cases in support of their argument. Others will cite few. Some will mention only the legal theory of their particular judicial circuit. Others will look to other circuits as well. We found that earlier decisions, those written when the good faith issue was first being addressed under the new law, tended to look further afield and compared more different approaches. Once a number of appeals courts had set standards for analyzing good faith, opinions tended to look more exclusively to appeals cases in their own circuit for guidance.

Further, the way we have applied precision-recall measures—using the court's opinion as the "right" answer—is but one way to examine performance. Another would involve comparing BankXX with other programs. Without such comparisons, it is hard to judge BankXX's performance.

Lastly, these measures are problematic for a program like BankXX which seeks to harvest as much information as its resource limits allow. If BankXX retrieves information not found in the opinions—which is likely to happen given its biases—this lowers BankXX's precision and does not help its recall, even though BankXX might be doing a superb job of legal analysis. Benchmarks better measuring retrieval *accuracy*[14] are needed in our experiments—and CBR or AI and Law, in general.

6. Conclusions

The general conclusion that we draw from BankXX is that the process of gathering information for an argument can be usefully modeled as heuristic search. In particular, the retrieval of cases and other knowledge can fruitfully be done with a combination of knowledge-based indexing and heuristic search. Using heuristic search as the mechanism to traverse memory permits relevancy assessment and case retrieval to be repeated iteratively in order to locate the nodes in the case graph that provide the underpinnings of an argument.

[14]In engineering, accuracy is different from precision, which only notes to what decimal point one measures.

7. References

Alterman, R. (1988). Adaptive Planning. *Cognitive Science*, 12, 393-422.

Ashley, K. D. (1990). *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. Cambridge, Massachusetts: M.I.T. Press.

Barr, A., Feigenbaum, E. A. & Cohen, P. (1981). *The Handbook of Artificial Intelligence*. Reading, Massachusetts: Addison-Wesley.

Branting, L. K. (1991). Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis. Ph.D. Thesis, Technical Report AI90-146, AI Laboratory, University of Texas, Austin, Texas.

Gentner, D. & Forbus, K. D. (1991). MAC/FAC: A Model of Similarity-based Retrieval. *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 504-509. Chicago, IL. Lawrence Erlbaum, Hillsdale, NJ.

Kass, A. M. & Leake, D. B. (1988). Case-Based Reasoning Applied to Constructing Explanations. *Proceedings, Case-Based Reasoning Workshop 1988*, 190-208. Clearwater Beach, FL. Morgan Kaufmann.

Kolodner, J. L. (1983). Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7(4), 243-280.

Kolodner, J. L. (1993). *Case-Based Reasoning*. San Mateo, California: Morgan Kaufmann.

Martin, C. E. (1990). Direct Memory Access Parsing. Ph.D. Thesis, Yale University, New Haven, CT.

Owens, C. (1993). Integrating Feature Abstraction and Memory Search. *Machine Learning*, 10(3), 311-340.

Rissland, E. L., Daniels, J. J., Rubinstein, Z. B. & Skalak, D. B. (1993). Case-Based Diagnostic Analysis in a Blackboard Architecture. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 66-72. Washington, DC. AAAI Press/MIT Press.

Rissland, E. L. & Skalak, D. B. (1991). CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 34, 839-887.

Rissland, E.L., Skalak, D.B. & Friedman, M. T. (1993). Case Retrieval through Multiple Indexing and Heuristic Search. *Proceedings, 13th International Joint Conference on AI*, 902-908. San Mateo, CA: Morgan Kaufmann.

Rosch, E. & Mervis, C. B. (1975). Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology*, 7, 573-605.

Turner, R. (1988). Organizing and Using Schematic Knowledge for Medical Diagnosis. *Proceedings, Case-Based Reasoning Workshop 1988*, 435-446. Clearwater Beach, FL. Morgan Kaufmann.

Veloso, M. M. & Carbonell, J. G. (1991). Variable-Precision Case Retrieval in Analogical Problem Solving. *Proceedings, Third Case-Based Reasoning Workshop*, May 1991, 93-106. Washington, D.C. Morgan Kaufmann, San Mateo, CA.