Evaluating a Legal Argument Program: The BankXX Experiments

Edwina L. Rissland
David B. Skalak
M. Timur Friedman

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

rissland@cs.umass.edu

Abstract

In this article we evaluate the BankXX program from several perspectives. BankXX is a case-based legal argument program that retrieves cases and other legal knowledge pertinent to a legal argument through a combination of heuristic search and knowledge-based indexing. The program is described in detail in a companion article in (this issue of) the *Journal of Artificial Intelligence and Law*. Three perspectives are used to evaluate BankXX: (1) classical information retrieval measures of precision and recall applied against a hand-coded baseline; (2) knowledge-representation and case-based reasoning perspectives, where the baseline is provided by the functionality of a well-known case-based argument program, HYPO [Ashley, 1990]; and (3) search perspective, in which the performance of BankXX run with various parameter settings, for instance, resource limits, is compared. In this article we report on an extensive series of experiments performed to evaluate the program. We also describe two brief experiments on ancillary questions regarding the program's search behavior and knowledge representation. Finally we offer some general conclusions that might be drawn from these particular experiments.

This research supported in part by grant No. 90-0359 from the Air Force Office of Sponsored Research and NSF grant No. EEC-9209623 State/University/Industry Cooperative Research on Intelligent Information Retrieval.

Contents

Part I: Introduction

1. Introduction: The Problem of Evaluating Arguments

There is no doubt that more research needs be done on the problem of evaluating arguments, whether they are complete arguments produced by humans or vestiges of arguments created by computers. Perelman and Olbrechts-Tyteca [1969, p. 462] have noted, in fact,

> In view of the complexity of the factors to be taken into consideration even just to judge whether an argument has any strength at all, it is curious that the writers of treatises on rhetoric should so glibly state, almost incidentally, that the strength of arguments is common knowledge and that they should base their advice regarding the order of discourse and the sequence of replies, on the degree of conviction that the arguments must have produced, which it is not hard for us to know, because we know what ordinarily brings this about.
>
> [Footnote to *Rhetorica ad Herennium*, I, 10].

These respected authors also observe [p. 461] that it is difficult to characterize even just the positive aspects of an argument: “Thus the strength of an argument shows itself as much by the difficulty there is in refuting it as by its inherent qualities.”

Fox and Clarke [1991] have suggested a collection of heuristic rules to gauge the relative persuasiveness of arguments:

> (R1) A larger set of positive arguments is more persuasive than a smaller set;
> (R2) A smaller set of negative arguments is less dissuasive than a larger set;
> (R3) Arguments that make unverified assumptions (e.g., defaults) are less persuasive than those which are based on grounded arguments (e.g., observations);
> (R4) Arguments that explain more observations are more persuasive than those that explain less;
> (R5) Any pair of arguments that is strictly independent is more persuasive than a pair in which [each] argument depends upon the other.

Ashley also provided a list of nine argument-evaluation criteria. Three criteria speak to the comparison of the whole argument proposed by a side, rather than the comparison of constituent precedents or points:

> 7. Improving an argument: The more nontrumped points there are, the better for a side’s argument.
> 8. Comparing opposing sides in one argument: If all of the nontrumped points favor Side 1 and there are no nontrumped points for Side 2, Side 1’s argument is stronger.
> 9. Comparing same side in two arguments: If there are more nontrumped points favoring Side 1 in argument [b] than in argument [a], argument [b] is stronger for Side 1 than argument [a].
>
> [Ashley, 1990, p. 279]

In general, there has been too little work that shows how to evaluate an argument. The difficulty stems from at least two sources. First, the subjective nature—or rhetoric—of argument is difficult to define and quantify, since it depends on how persuasive the argument is to its audience, jury, court, etc. Second, there is a fundamental lack of knowledge on what an argument is comprised of—its epistemology or structure. While there have been several fundamental contributions on this front—Olbrechts-Tyteca, Toulmin, Ashley, Branting, etc.—much remains to be done. We have barely scratched the surface for certain kinds of argument, such as appellate oral argument (e.g., [Rissland, 1990]), and have yet to address others, such as legal briefs. Our work here with argument pieces (elaborated in the first article of this two article pair) as well as past work on the effect of high-level purposes on the content of argument [Rissland et al., 1993], statutory argument strategies and tactics [Skalak & Rissland, 1992], precedent-based arguments [Rissland et al., 1984; Ashley, 1990], etc. has addressed certain structural aspects of argument. Of course, although we emphasize the non-rhetorical aspects of argument, what we have encoded via such mechanisms as argument factors, argument pieces, dimensions, etc. does reflect some of our intuitions about the persuasive aspects of argument.

With respect to arguments produced by computers, previous researchers have relied on a variety of techniques to validate their programs against realistic baselines. For HYPO, Ashley made comparisons between the output of HYPO program and actual opinions in the trade-secrets misappropriation area [Ashley, 1990]. Branting [1991] performed a series of experiments that enlisted the services of a domain expert in the Texas worker's compensation area to grade the output of the GREBE program against memoranda written by four advanced law students. Gardner [1987] overcame the lack of a right answer to most legal questions by comparing the output of her program with the answers given to five short offer-and-acceptance questions in *Gilbert Law Summaries: Contracts,* a study aid for law students [Eisenberg, 1982].

In this work, we address the evaluation question in three types of experiments:

1. **External benchmarking comparisons** of BankXX by running BankXX on real legal cases and comparing its output with hand-coded representations of the opinions in the cases;
2. **External system comparisons** of BankXX with a re-implemented version of the HYPO program called μHYPO; and
3. **Internal system comparisons** of the performance of BankXX with itself, by varying parameter settings.

No single one of these evaluation approaches is perfect. For instance, none captures all the aspects we wish to evaluate, and some introduce aspects (e.g., constraints on case citation stemming from jurisdictional considerations) that we do not consider in BankXX. Nonetheless, the composite does give us a fairly detailed look at BankXX's performance. Also, note that only the internal comparisons are completely "fair" in the sense that what are being compared (BankXX against BankXX) are perfectly commensurable. On the other hand, the external comparisons are more interesting. Evaluating argument is truly a "two-handed" exercise—on one hand, on the other hand.

Another way of understanding our approach to evaluation is in terms of the following three substantive perspectives addressed in the BankXX project:

1. **Information retrieval:** How well does BankXX retrieve the "right" information. To benchmark BankXX's performance from this perspective, we measure its performance against hand-coded versions of real cases using traditional IR measures of precision and recall.
2. **Knowledge representation and case-based reasoning, particularly case retrieval**: How well do BankXX's representation and CBR mechanisms allow it to perform? In this perspective, we compare retrieval performance of a re-implemented subset of the seminal case-based reasoning program, HYPO [Ashley, 1990] with that of BankXX. BankXX borrows a number of case representation and analysis techniques from HYPO, but HYPO does not contain some types of knowledge that are represented in BankXX, such as legal theories and factual prototypes.
3. **Heuristic search**. How do the computational details of heuristic search—evaluation function, start node, resource limit—affect performance? In this set of experiments we compare the efficiency and efficacy of BankXX under different (internal) parameter settings that define its operation.

These three perspectives correspond with the three approaches to evaluation. They are the "semantics" underlying the "syntax" of those experimental approaches. The driving motivation for all of this work is our fundamental concern with legal argument. It permeates all our experiments and all aspects of the BankXX system.

In general, our goal in this paper is to explore the use of heuristic search, guided by evaluation functions capturing various types of information needs—as reified in BankXX—to retrieve information, such as cases and legal theories, for use in legal argument from a "library" of highly interconnected sources, as is typical in law. In particular, we investigate several hypotheses:

• BankXX endowed with a knowledge-richer evaluation function performs better than BankXX with a knowledge-poorer one, that is, BankXX/AP and BankXX/AF outperform BankXX/NT.

• BankXX/AP produces more balanced arguments than BankXX/NT.

• BankXX exhibits performance improvement over HYPO on the task of retrieval.

• BankXX's performance improves, and then levels off, with increased resources.

In our experiments, we measure performance—and thus define "better"—with traditional precision and recall statistics and with counts of numbers of items retrieved. To use precision-recall measures, we must define test problems and answer keys; for these, we use real cases—the 55 represented in BankXX's case base—and their court opinions. Since we are concerned with the use of such measures, which may not be totally appropriate in our domain, we also briefly explore an alternative approach using modified precision-recall measures.

The remainder of this paper is divided into six main sections. First we describe general aspects of the experiments carried out upon BankXX and further discuss the three evaluation perspectives and types of evaluations. Each is explored in depth in one of the next three sections (Sections 3, 4, 5) of the paper. The sixth, and penultimate section of the paper, describes a final set of experiments that respond to questions that arose in the course of these experiments and an alternative approach to precision and recall. The article closes with a short discussion of alternative design decisions and with general points that might be drawn from these experiments with BankXX and the particular knowledge we have given it.

2. Evaluation of BankXX: General Experimental Design

2.1 The Experimental Space

Setting up a series of empirical evaluations involves making a series of choices. Here we briefly lay out the choices we explore in our BankXX experiments. Obviously there are others as well.

One can think of the top level choice as the choice of experimental approach. There are the three major choices in approach, as laid out in the introduction:

1. External benchmarking (against court opinions)—explored in Section 3.
2. External cross-system comparison (with HYPO)—explored in Section 4.
3. Internal comparison (with different parameter choices)—explored in Section 5.

Next, there are several parameters that can be varied within BankXX. These stem primarily from BankXX's computational architecture as a heuristic search program. The configuration choices for BankXX are:

1. the evaluation function:
   a. *node-type* (NT) evaluation function (denoted ***BankXX/NT***)
   b. *argument-piece* (AP) evaluation function (***BankXX/AP***)
   c. *argument-factor* (AF) evaluation function (***BankXX/AF***)
2. the start node:
   a. a standard well-known case (*Estus*)
   b. the problem case
   c. a random node
3. resource limits:
   a. number of nodes closed (e.g., 30, 60, 90);
   b. number of "billable seconds" (e.g., 1, 2, 10)[1]

The terms and weights for the three evaluation functions are given in the Technical Appendix. Note that BankXX/AP in addition to using the fill limits on argument pieces in the argument-piece evaluation function, also uses the same fill limits when deciding whether or not to harvest a node it has closed. Thus, for instance, if a case is both a leading case and a same-side case and BankXX has already reached its fill on same-side but not leading cases, the case will only be harvested, and listed in BankXX's output, as a leading case.[2] Regardless of configuration, BankXX always harvests information for the same standard dozen argument pieces.[3]

[1] "Billable seconds" is the time BankXX takes from the moment it is invoked until it returns its output. It includes any time spent on garbage collection and output to the screen during intermediate processing.

[2] See the companion paper, for instance, Section 3.5 concerning the argument-piece evaluation function and Section 4.3 concerning an extended example of BankXX/AP processing the *Estus* case as a problem case.

[3] N.B., in these experiments the *family-resemblance-prototype* argument piece is usually empty since we usually have disabled the aspects of BankXX needed to fill it.

We vary each of these three configuration choices in the internal comparisons, discussed in Section 5. In both external comparisons, the start node and resource limits are not varied. **All external comparison experiments use the *Estus* case as start node and resource limits of 30 closed nodes and 1000 billable seconds.** These particular parameter settings were found to be a good choices after examination of the results of the internal experiments. (See Section 5.) In particular, the time limit is more than enough to allow BankXX to run until its list of items to explore (i.e., its OPEN list) is empty.

In addition, there is a fourth configuration choice as to whether BankXX filters its output for items—cases decided or theories promulgated—dated after the problem case and when it does so. Date-filtering is needed in both types of external benchmarking experiments in order to exclude post-dated cases and theories that are extraneous to the problem-case. The choices for date-filtering are:

a. *no filtering*
b. *post-processing filtering*—after BankXX has concluded processing a problem case.
c. *during-processing filtering*—while BankXX is processing a problem case, specifically, at the time a node is expanded. Thus, post-dated items never get put on the OPEN list.

All external experiments—discussed in Sections 3 and 4—filter for dates during processing. The internal experiments do not filter for dates at all. By comparison, the extended example of the companion article used post-processing date filtering.

Finally, there are choices of metric, comparison points, and test cases to use in the external benchmarking experiments comparing BankXX's output with real court opinions:

1. Evaluation metric:
   a. classic precision-recall
   b. modified precision-recall
2. A set of comparison points:
   a. the full set of BankXX's standard 12 argument pieces
   b. a set of 4 simplified *aggregated* argument pieces
3. A set of test cases (to use in averages):
   a. the entire 55 case corpus (called *All*)
   b. only the top 20 cases in the leading-cited cases ranking (*Top 20*)
   c. only the 14 appellate cases from the corpus (*Appellate*)
   d. only the 14 so-called *meaty* cases (*Meaty*)[4]

For the most part, we only use the classic precision-recall measures in these experiments although we briefly explore a modified version of them (in Section 3.6) to emphasize that the choice of metric is indeed a choice. There is no a priori mandate to use, or use exclusively, precision-recall measures in their traditional form. And, that in the field of AI and Law, there are good reasons not to and that we need alternative approaches in fields, such as ours, where

[4]A meaty case is a case having numbers of cited cases above a certain threshold. This class was defined because so many of the cases in this area of law cite very few precedents. A few cite none at all. See Section 3.5, especially Table 3, for data on this point. The definition of *meaty* case is given below in Section 3.6.

there are no unassailable “correct” answers to use as evaluative standards. We explore use of a modified precision-recall to encourage, by example, development of other measures.

To suit the circumstances of what is being compared—BankXX-Court opinions, BankXX-HYPO, BankXX-BankXX—we use different sets of comparison points to define what is being compared in the various experiments. The external experiments use a set of four simplified aggregated argument pieces. The internal experiments use the standard dozen. Finally to analyze performance on different types of cases, we use four different partitions of the BankXX case corpus.

In summary, the experimental space is quite large: 3 overall experimental approaches, 3 configurations of BankXX, 4 partitions of the case base, etc. In these experiments we try to explore portions of this space in the classical way by varying one aspect while holding others constant. For instance, within the internal comparison experiments, we vary the start node while holding the evaluation function and resource limits constant. We do not use n-fold validation because there is no training phase in BankXX; all cases can be used as test cases. We use all 55 cases in BankXX’s case base as problem cases in a leave-one-out *de novo* manner. Of course, this can be considered an extreme case of cross-validation.

Even with the very rich set of variations engendered by the above choices, we are still only exploring a small subset of the possibilities. For instance, throughout our experiments we have not tinkered with the terms and weights in our evaluation functions. In fact, each of the three evaluation functions is but one particular function from a family of functions spanned by the terms; others can be generated by varying the weights. Other such closely related evaluation functions beg to be explored. In particular, if one is interested in fielding a system such as BankXX, tuning the weights and terms is a must.[5]

We have performed a large number of experiments—and this is a long paper—and some results require a close technical explanation that appeals to the design and implementation of the program. For convenience, for each experiment, we summarize the parameters that are varied and held constant, and we offer a very broad summary of the results. These are presented in boxed areas of text. However, we caution the reader against wholesale reliance on these summary characterizations and extraction of the summaries from the experimental context in which they are made. A rather blanket caveat is that the results we present are limited to the particular set of cases and theories we have incorporated into the program.

[5]The data in these experiments suggest some changes to term weights and thresholds, for instance, increasing the fill limits on *supporting-cases* and *contrary-cases* argument pieces in the argument-piece evaluation function.

2.2 The *de novo* Treatment of a Problem Case

In these experiments when a case from the case base is treated as a problem situation *de novo*, it is run as if never seen before by BankXX. The point of view (debtor or creditor) given to BankXX is that of the side that prevailed in the actual court case. Each of the 55 cases in the BankXX case base is treated in this way.

In the de novo approach, we excise the problem case from the case-domain graph by excising the nodes that represent it (e.g., fact situation node, citation nodes) and disabling the pointers that connect it to other cases and items of knowledge (e.g., legal theories). In particular, we decouple the de novo case from any legal theories it actually put forth or applied. For instance, when the *Estus* case is run in a de novo fashion, the *Estus* theory, which was promulgated by the *Estus* case, is treated as if it came from some other case (not included in BankXX's case base) since it is left in the knowledge base but its links to the *Estus* case are removed.

2.3 Additional Comments on Date Filtering

We use the *during-processing* date filtering option in the external benchmark experiments since post-processing filtering would be disadvantageous to BankXX. The question of dates is not an issue in the internal experiments. Of course, in the intended use of BankXX on new problem cases, date-filtering is not needed since everything known to BankXX should be available.

With the post-processing date-filtering option, all cases and theories remaining in the case-domain graph after de novo preparations are available to BankXX, regardless of their dates relative to the problem case. For instance, a case decided after the problem case is still fair game for consideration. The effect of post-processing date filtering can be significant since many items are often deleted, especially for earlier cases.

There are two reasons not to use post-processing date filtering. First, post-dated cases and theories—cases decided or theories promulgated after the date of the problem case—are ignored in the external comparisons, so they count for naught. Second, they represent wasted resources. Especially with BankXX/AP, where there is a limit on how many items can be harvested to fill any argument piece, a harvested post-dated item represents a lost opportunity: a pre-dated item actually used in the actual court opinion might have been harvested instead. For instance, there are limits of three cases each for supporting and contrary cases; if these argument pieces are filled by extraneous cases, pertinent cases must be passed up.[6] In addition, consideration of extraneous items uses up other limited resources like allowed computation time.

[6]See the treatment of [GOEB] and [IACOVONI] in the extended example of the companion paper (Section 4.3).

Part II: The BankXX Experiments

3. External Evaluation from an Information Retrieval Perspective: Comparing BankXX with Hand-Coded Court Opinions

In this section, we present our primary external evaluation of BankXX in which we compare the performance of the three configurations of BankXX against hand-coded opinions of the cases in the BankXX case base. In Section 3.1, we present our methodology, including the definitions of the four aggregated argument pieces used in the comparisons and a discussion of precision and recall measures. In Section 3.2, we give an overall assessment of BankXX's performance. In Sections 3.3, 3.4, and 3.5 we examine the results in much greater detail. In particular, in Section 3.4 we examine BankXX performance on various subsets of cases and in Section 3.5 we examine the effects of sparseness. In Section 3.6 we explore an alternative definition of precision and recall based on case similarity.

In these BankXX-court experiments we examine how well BankXX retrieves the cases and theories that are actually cited in the court opinions of real legal cases. The evaluation methodology was to run BankXX in the *de novo* manner (see Section 2.2) on each of the 55 cases in the BankXX case base three times—once in each configuration of BankXX with other parameters held constant—and then to compare the items harvested by BankXX with those from the hand-coding. For this comparison we represent both the opinions and BankXX output in terms of four simplified, aggregated argument pieces. For each of the four aggregated argument pieces, we computed precision and recall scores. This produced 3 x 4 x 55 = 660 raw data points and 1320 individual precision and recall scores, which we examine in various ways.

Parameters Varied:
    BankXX configurations: BankXX/NT, BankXX/AP, BankXX/AF
Parameters Held Constant:
    Start node: *Estus*
    Closed node limit: 30
    Time limit: 1000 billable seconds
Date-filtering option:
    During-processing

3.1 Methodology

3.1.1 The aggregated argument pieces

In order to carry out BankXX-court opinion comparisons—external benchmarking with an IR perspective—we needed to create a corpus of "correct" answers. To do this we encoded each case in the BankXX corpus by hand by entering each case and theory actually cited in the published opinion into its appropriate argument piece[s]. We then checked these against those contained in BankXX's case-domain-graph and only kept those that are; for comparison purposes, it would not make sense to check if BankXX harvested something that it could not know about. For example, if an opinion cited the important *Estus* and *Deans* cases, which are present in BankXX, we would list them in appropriate argument pieces, such as *leading-cited-cases*.

In doing this encoding, we initially tried to use all twelve categories of cases and theories defined by the standard dozen BankXX argument pieces. (See Section 3.3.1 of the companion paper). However, we found that some of the technical distinctions made easily by BankXX were difficult for us to make. For example, it is difficult to say whether a theory belongs to the *applicable-theories* or the *nearly-applicable-theories* argument piece. This is not a problem for BankXX since BankXX uses a well-defined test based on the domain factors to decide whether a theory is applicable or nearly applicable. In order to avoid forcing hard-to-make distinctions, we defined a set of ***aggregated argument pieces***. They aggregate case or theory sub-categories and are simpler than those normally used by BankXX.

To encode court opinions, we used three broad categories aggregating pairs of standard BankXX argument pieces:

- *applicable* and *nearly applicable* theories,
- *supporting-best* and ordinary *supporting* cases,
- *contrary-best* and ordinary *contrary* cases.

We also used the *leading-cited-case* category without alteration since there were no difficulties involved in applying it: one merely checks whether a cited case is on the 'hit parade' of the top five most cited cases in the BankXX corpus.[7] See Figure 1.

1. ***theories***	any theory mentioned in the opinion
2. ***pro (same-side) cases***	any same-side case mentioned in the opinion
3. ***con (contrary) cases***	any contrary case mentioned in the opinion
4. ***leading cited cases***	any of the top 5 most cited cases in the BankXX corpus

Figure 1. The four categories used to encode court opinions in the BankXX-court comparisons. The pro viewpoint in the encoding is that of the side that prevailed in the actual court case.

Given a simplified encoding for the court opinions, we needed to use the parallel set of aggregated argument pieces for BankXX output: the three new aggregated argument pieces plus the original *leading-cited-cases* argument piece. See Figure 2.

1. ***aggregated-theories***	union of *applicable-theories* and *nearly-applicable-theories*
2. ***aggregated-pro-cases***	union of *supporting-cases* and *supporting-best-cases*
3. ***aggregated-con-cases***	union of *contrary-cases* and *contrary-best-cases*
4. ***leading-cited-cases***	*leading-cited-cases*

Figure 2. The four aggregated argument pieces used for BankXX output in the BankXX-court comparisons. The pro viewpoint is that of the prevailing side in the actual court case.

These four argument pieces—involving seven of the original dozen argument pieces—are the basis of all the BankXX-court comparisons. We ignored the other five argument pieces (*supporting-citations, factor analysis, overlapping-cases, factual-prototype-story, family-*

[7]The top five most cited cases—the *leading-cited-cases*—are: 1. *Rimgale*, 2. *Estus*, 3. *Goeb*, 4. *Deans*, 5. *Iacovoni*.

resemblance-prototype) because they were typically too computational in nature to use in the hand-encoding of court opinions.[8] While the aggregated categories do wash out some of the distinctions that we feel are indeed important in legal argument and they ignore much of what BankXX accomplishes, they allowed us to minimize subjectivity or ambiguity in the hand-encoding.[9]

In summary, if an opinion cited a case, we simply encoded it as *pro*, *con,* or *leading*, and then filtered it against those known to BankXX (in the case-domain-graph). We did not try to distinguish a set of best cases, work out other nuances captured in the BankXX argument pieces, or hand-simulate their computational definitions (functional predicates).[10] A cited theory was listed, regardless of whether the court said that it did, nearly did, or should apply, etc.

3.1.2 Comparing BankXX output with the hand-coded case opinions

At this new coarser-grained level of representation, the arguments output by the program and those encoded by hand from actual court opinions were compared automatically. For each aggregated argument piece a comparison module listed the items that were:

(a) ***overlap***—included in both BankXX's output and the hand-coded argument;

(b) ***missed***—included in the hand-coded argument but not output by BankXX;

(c) ***additional***—output by BankXX but not included in the hand-coded argument.

This breakdown provides a qualitative sense of how well the program has performed in analyzing a given case. See Figure 3 for an example.

[8]For instance, *overlapping-cases* must have 75% or more of their domain factors in common with the problem case. *Factor-analysis* requires strict (computational) application of domain factors (dimensions), particularly, their prerequisites; in addition, it is not that illuminating in the context of this experiment. *Supporting-citations* requires a careful check of cases cited by supporting citations (e.g., *accord*, *see*). *Family-resemblance-prototype* requires application of the Rosch prototypicality metric. The *factual-prototype-story-category* was also left out because (if retrieved by BankXX) it would always match on both sides of the comparison since it was a category tag assigned by us.

[9]There is no doubt that judges and lawyers do have a sense of "best" or "better" case. The problem is that it is sometimes hard to make such distinctions consistently in a well-defined manner. Since we wanted the same intuitive sense to be used throughout the set of hand-coded answers, we decided to sacrifice descriptiveness for consistency in the external BankXX-court comparison experiments.

[10] If we had used a hand-simulation sense for encoding the opinions, we would have defined the best BankXX could have done on arguments represented in BankXX's terms. This would have been another interesting set of experiments to have done. Since it would have involved a great deal of effort and we were interested in a true external comparison, we did not perform it.

```
AGGREGATED-THEORIES:
OVERLAP              MISSED                         ADDITIONAL
                     ALL-THE-FACTS-AND-CIRCUMSTANCES   OLD-BANKRUPTCY-ACT-GOOD-
                                                            FAITH-DEFINITION

LEADING-CITED-CASES:
OVERLAP              MISSED                         ADDITIONAL
ESTUS                                               IACOVONI
DEANS
GOEB
RIMGALE

AGGREGATED-PRO-CASES:
OVERLAP              MISSED                         ADDITIONAL
ESTUS                KULL                           BURRELL
HEARD                                               CHURA
                                                    IACOVONI

AGGREGATED-CONTRARY-CASES:
OVERLAP              MISSED                         ADDITIONAL
DEANS                BELLGRAPH                      GOEB
RIMGALE              BARNES
```

Figure 3. Comparison of the argument generated by BankXX/AP with the hand-coded version of the *Kitchens* case using aggregated argument pieces. BankXX filtered for dates during-processing. The point of view taken by BankXX ("pro") was that of the creditor who won in the actual case.

To get a quantitative measure of comparison, we use traditional *precision* and *recall* measures used in information retrieval (IR) [Salton, 1989] with the hand-coded argument serving as "the answer" or "benchmark" against which BankXX is measured.[11] In other words, an item is deemed "correct" if it occurs in the hand-coded answer. Precision and recall are calculated separately for each of the four aggregated argument pieces used in the external comparisons.

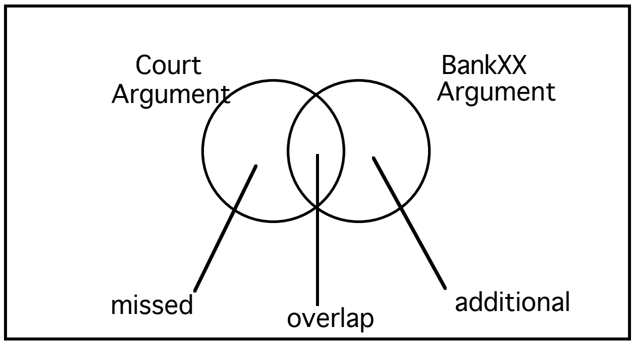

Figure 4. Regions of interest in BankXX-court experiments: cases and theories found by both BankXX and the opinions (*overlap*), those found in the opinion but not by BankXX (*missed*), and those found by BankXX but not in the opinion (*additional*).

In terms of overlap, missed, and additional cases, the standard definitions of precision and recall are:

[11]Note, if one reverses this convention and uses BankXX as the benchmark, precision and recall values are interchanged.

1. **recall** is the fraction of correct items that were retrieved by BankXX:
   $|overlap| / |missed \cup overlap| = |overlap| / |hand\text{-}coded\ answer|$
2. **precision** is the fraction of items retrieved by BankXX that are correct:
   $|overlap| / |additional \cup overlap| = |overlap| / |BankXX\ output|$

See Figure 4.

3.1.3 Precision-recall scores and empty answers

In the case of ratios of 0/0, these numbers are undefined. These occur when either the hand-coded answer is empty or BankXX's answer is empty.[12] The matrix in Figure 5 sums up the possibilities.

	BankXX's answer	
Coded Answer	**empty**	**non-empty**
empty	(A) p=0/0 r=0/0 (N=21)	(B) p=0 r=0/0 (N=90)
non-empty	(C) p=0/0 r=0 (N=3)	(D) as defined (N=546)

Figure 5. The possible combinations of empty/non-empty court and BankXX answers. ***N*** is the number of occurrences out of the 660 possible data points from all three BankXX configurations.

For instance, a **recall** value cannot be computed when there are no items listed in the hand-encoding of the court opinion (cells ***(A)*** and ***(B)*** of the matrix). The numerator is 0 since the intersection of those items retrieved by BankXX and those listed in the hand-encoded answer—the null set—is the null set, and the denominator is 0 since BankXX has not missed any items. Thus for a case with an empty answer, the recall ratio is 0/0, an undefined value. The problem of empty answers with concomitant undefined recall values occurred many times in our experiments (N=111 or about 17% of the possible data points). Twenty of our cases had at least one argument piece that was empty.[13] The problem of undefined recall scores does not arise with non-empty answers.

[12]When we speak of a BankXX *answer*, we mean the answer for a particular aggregated argument piece with respect to a particular configuration of BankXX. Since there are 4 aggregated argument pieces, 3 configurations of BankXX, and 55 cases, there are 660 BankXX answers produced in this set of experiments. Of course, there are only 220 (4x55) hand-coded answers since these are the same regardless of BankXX configuration.

[13]In the BankXX corpus of 55 cases, there were 37 aggregated-argument pieces having empty hand-coded answers: 12 cases had no *aggregated-theories,* 8 had no *aggregated-same-side-cases*, 13 had no *aggregated-contrary-cases*, 4 had no *aggregated-theories*. Thus, there are 111 (3x37) instances of argument pieces with empty hand-coded answers in our BankXX-court comparison experiments over the three configurations. Many cases had more than one empty aggregated argument piece.

For a case with an empty hand-coded answer, if BankXX retrieves no items, the **precision** ratio will also be 0/0 (cell ***(A)*** of the matrix). This occurred very infrequently (N=21 or about 3%) and only with cases decided in 1980 or 1981.[14]

However note that with an empty hand-coded answer, precision will be 0% if BankXX retrieves any items whatsoever—they'll all be "additional"—cell ***(B)*** of the matrix. Since a BankXX answer is almost never empty, this phenomenon of p=0 on empty hand-coded answers occurred many times (N=90 or about 14% of all the answers). Approximately two-thirds of these situations occurred with early cases (i.e., case decided in 1980 through 1983), when there were few precedents available to cite.[15]

For a non-empty hand-coded answer, the precision ratio can also be undefined when BankXX's answer is empty since in this situation, both the overlap and additional item sets will be empty (cell ***(C)*** of the matrix). (*N.B.*, in this case recall will 0%.) This problem almost never happens, since BankXX is designed to retrieve as much information as its resource limits allow. It occurred extremely infrequently (N=3 or less than .5%)—and only with the earliest (1980) cases.[16] What caused BankXX to produce its few empty answers was the lack of date-appropriate items. If few items can pass through the date filter—which is the situation with the earliest cases—few can be opened, let alone harvested (*if few can be called, fewer can be chosen*).

If BankXX produces a non-empty answer with no overlap with a non-empty hand-coded answer, both precision and recall will be 0 (as they should be). Of the (546) instances where both precision and recall values are defined (cell ***(D)*** of the matrix), this problem of *skew* or *double-0* answers happened quite a bit (107 times). It is very much related to the sparseness of some cases.[17] It occurred the most with *aggregated-theories* where 50% of all such skew answers occurred. Double-0 answers occurred the least on BankXX/NT and the most on BankXX/AF.[18]

---

[14]All of these 21 instances occurred in 9 (of the 10) 1980 and 1981 cases.

[15]Of the 90 scores in the cell (B) category, 61 involved early (1980 through 1983) cases. With BankXX/NT 71% of such cell (B) scores occurred with early cases, 64% with BankXX/AP, and 67% with BankXX/AF. With respect to argument pieces: 75% (9 of 12) of such *aggregated-theory* scores, and 67% (52 of 78) of such scores on the three case-related argument pieces occurred with early cases. There is quite a strong tie between 0%-precision-on-empty-cases and the early cases. In fact, earliness and sparseness are not unrelated but tend to co-occur. Over 50% of the early cases are empty or "sparse" (i.e., just have 1 or 2 items in hand-coded argument-piece categories) and 50% of empty or sparse cases are early. This is not surprising.

[16]It occurred on only 3 of the 660 possible instances, and only with two 1980 cases (*Heard*, *Terry*). For instance, *Terry* had 1 *leading-cited-case* and BankXX missed it and found no other alternatives.

[17]With BankXX/NT, 18 of 24 (75%) of the double-0 answers involved sparse cases; with BankXX/AP, 29 of 36 (81%); and with BankXX/AF, 36 of 47 (77%). See the discussion of sparseness in Section 3.5.

[18]Of 220 possible instances in each configuration, it occurred on 11% (N=24) of the instances with BankXX/NT; 16% (N=36) with BankXX/AP and 21% (N=47) with BankXX/AF.

In summary, if the hand-coded answer is empty (regardless of whether BankXX's answer is empty or not), recall is undefined. If BankXX's answer is empty (regardless of whether the hand-coded answer is empty or not), precision is undefined. The matrix in Figure 5 sums up the possibilities.

3.1.4 Handling undefined precision and recall ratios

There are two approaches to dealing with undefined values:

1. call them *undefined*; and not use them (e.g., in any averaging computations)
2. give BankXX 100% credit, that is, set them to 1.

There are arguments for both options. For instance, if no cases are cited by the case opinion and none are found by BankXX—resulting in a 0/0 for both precision and recall—BankXX has done exactly what is called for according to the answer: BankXX retrieved all the cases there were and no additional ones. That is, it retrieved exactly what was in the answer and should get 100% for both precision and recall. Throwing out the datum is somewhat "unfair" to BankXX since this in effect penalizes BankXX for the lack of a contentful answer in the court's opinion (that overlaps with BankXX's knowledge-base).

We have used both approaches and label results accordingly. We indicate the number of 0/0 situations included/excluded in any averages we compute. In fact, there is not much difference between the two approaches in the context of these experiments except that the second approach leads to slightly higher averages, which is to be expected.

3.1.5 Propriety of precision and recall as performance measures

In our context of legal argument, the traditional IR assumption of an unequivocal master answer key is a weighty one. Jurisprudentially, it is problematic to elevate the cases and theories that are mentioned in a legal opinion as the "correct" or "best" ones. It is not even clear from a jurisprudential standpoint what it might mean to consider case citations as the best or the correct ones. Given the workload of most courts in the U.S., it is doubtful that judges and their clerks have the time or the facilities to seek out the very best case. They may rely for their citations and theories upon briefs written by attorneys who are equally overburdened and, of course, interested in analyzing the case from the viewpoint of their clients. While there may be a presumption of the appropriateness and worth of citations found in U.S. Bankruptcy Court decisions, the matter is far from proved to the extent necessary to rely on those citations as providing an answer key. There is no assurance that the court is putting forth the best argument that could be made. It is easy to conceive that a better argument may be made than the than the court's, which clearly undermines the utility of treating the opinion as an answer key.

Furthermore, each opinion is the product of an individual judge and clerks. Some cite many cases in support of their argument; others, few. Some mention only the legal theory of their particular judicial circuit; others 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. This can create mismatches between the criteria, such as jurisdiction, used by a court in selecting its citations and those used by BankXX.

In our domain, many of the court opinions are quite skimpy, which means very small numbers are involved in the precision-recall calculations. (See the Table 3 giving sparseness data in Section 3.5.) This makes such scores ‘unstable.’ For instance, a zero level of recall might reflect the failure of BankXX to find the one or two cases cited by a court; finding those one or two cases could boost recall from 0% to 100%. A low level of precision might reflect the fact that there are very few cases in the hand-coded answer and thus very few cases found by BankXX count as the “correct” cases to have found; any extras drive down the precision. For example, if there is one case in the overlap and no extras, precision is 100%; one extra case found by BankXX halves this to 50%, one more cuts it to 33%. Thus, large differences in recall and precision can be engendered by the small numbers of items involved.

In general there is a tendency to place too much faith in numerical scores in all evaluation work. This is especially risky in the case of our precision-recall scores since the numbers involved are small. For instance, an opinion might mention only 2 or 3 cases. A recall of .66 can sound quite seductive whereas 2 of 3 might not and a precision of 100%, smashing, whereas 2 of 2, maybe not. There is a danger in thinking of such numbers as more meaningful than they are.

In addition, 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 and the sparseness of many opinions—this lowers its precision and may not help its recall, even though it might be doing a good job of legal analysis.

Nevertheless, the theories and cases cited in an opinion do provide a useful benchmark and the rest of this section on external evaluation uses hand-coded opinions as the benchmark against which to measure BankXX’s performance. However one must bear in the mind the jurisprudential and measurement difficulties we have touched on.

3.2 Comparing Evaluation Functions

3.2.1 Qualitative observations

BankXX/NT

Since in this set of experiments, the start node was always taken to be the well-known *Estus* case, all the answers produced by BankXX using the node-type evaluation function (BankXX/NT) look very similar. BankXX/NT explored the case-domain graph in roughly the same way in each problem case since BankXX/NT in essence prioritizes its search according to the *types* of nodes and the types of nodes connected to *Estus* (via the neighbor methods) do not change.

What accounted for the small changes among BankXX/NT's raw output on problem cases were: (1) the year of the problem case and hence the cases that could pass through the date filter and be opened; (2) the cases that qualified as most on-point and hence could be used to initialize the open list; and (3) the cases that qualified as best cases and could be harvested as *aggregated-pro-cases* and *aggregated-con-cases*, which include best cases. There is more variation in BankXX/NT's search in early cases than in late ones (where hardly anything is passed over due to date-filtering).

In summary, BankXX/NT produced quite rote problem-solving. There were two typical strings of harvested cases, one pro-debtor and one pro-creditor. BankXX/NT did surprisingly well on recall with these. Thus, BankXX provides a good baseline, even if it was neither particularly discriminating nor problem-sensitive.

Concerning the two baseline strings of cases, it is important to note that we would not have known about them if we had not run BankXX/NT. This raises the interesting question of how to use such an initial result—produced by not particularly clever but still useful means—as a "learning experience" for the system to be used later in more clever problem-solving. For instance, a human legal researcher would eventually learn a standard set of cases to use as an initial guess as to which cases to cite in a problem. But, of course, these would only serve as an initial "first-order approximation" to an answer since one cannot get away with a totally rote dumping of the same string cites for each case.

BankXX/AP and BankXX/AF

The other two configurations of BankXX produced quite different searches for different problem cases. In particular, BankXX with the argument-piece evaluation function (BankXX/AP) produced very varied results since its evaluation function is quite responsive to the problem-solving context. For instance, the argument-piece evaluation function (see the Technical Appendix) breaks cases down into 6 sub-categories that depend on the problem case. In contrast, the node-type evaluation function uses only 5 broad categories that are independent of the problem case.

BankXX/AP and BankXX/AF were much less profligate in their output than BankXX/NT (see Figure 6). In the case of BankXX/AP, fill limits on argument pieces kept numbers of harvested items quite low, especially for pro and con cases.[19] In fact, the sum total of fill limits from the 7 argument pieces that contribute to the 4 aggregated argument pieces that comprise an answer is 28, less than the limit of 30 closed nodes.[20] This guarantees small answers for BankXX/AP. Output from BankXX/AF was also comparatively small, not because of any limits on items harvested, but because of resource limits since computing with argument factors is time-consuming.

[19]E.g., fill limits relevant to pro and con cases are: 5 *best-supporting-cases*, 3 *supporting-cases*, 3 *best-contrary-cases*, 3 *contrary-cases*.

[20]Since an individual item can be harvested by more than one argument piece, BankXX/AP is likely to throw away some of the information it finds. For instance a best case can be used by two (if it's a leading case, three) argument pieces.

Note these observations—larger, fairly rote, answers under BankXX/NT and smaller, problem-sensitive answers under BankXX/AP and BankXX/AF—suggest a classic trade-off due to differences in how knowledge-rich the evaluation functions are: larger but lesser quality answers with BankXX/NT and smaller but higher quality answers with BankXX/AP or BankXX/AF. Analogous observations arise in the precision-recall data, explored in the detailed in following sections.

BankXX/NT:

AGGREGATED PRO CASES: *Estus, Heard, Kull, Terry, Strong, Sotter, Sheets, Sellers, Sanders, Iacovoni, Burrell, Chura*

AGGREGATED CON CASES: *Deans, Barnes, Rimgale, Valentine, Flygare, Goeb, Ali*

LEADING-CASES: *Estus, Deans, Goeb, Rimgale, Iacovoni*

THEORIES: Old-Bankruptcy-Act, Kitchens-Kull Theory

BankXX/AP:

AGGREGATED PRO CASES: *Estus, Heard, Burrell, Chura, Iacovoni*

AGGREGATED CON CASES: *Deans, Rimgale, Goeb*

LEADING-CASES: *Estus, Deans, Goeb, Rimgale, Iacovoni*

THEORIES: Old-Bankruptcy-Act

BankXX/AF:

AGGREGATED PRO CASES: *Estus, Burrell, Chura*

AGGREGATED CON CASES: *Deans, Barnes, Goeb, Ali*

LEADING-CASES: *Estus, Deans, Goeb*

THEORIES: Kitchens-Kull

Figure 6. Sample BankXX Output: the *Kitchens* (1983) case.

Qualitative observations on system performance can be shown in histograms that record the number of objects in the OVERLAP, MISSED and ADDITIONAL categories. These give a rough idea of how BankXX output compares with the hand-coded arguments. Figure 7 shows histograms for each of the four aggregated argument pieces for BankXX/AP. The two other BankXX configurations produced similar sets of histograms.

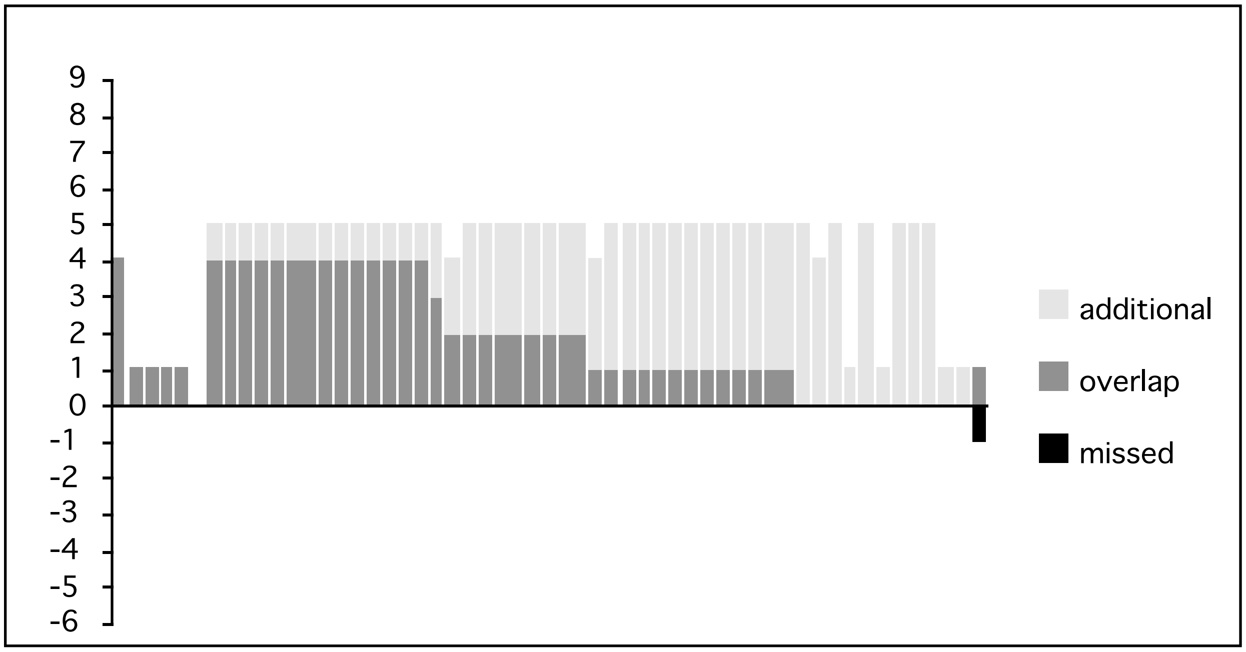

Figure 7a. Histogram showing for each case the number of items that were in the overlap, missed and additional categories for the *leading-cited-cases* argument piece for BankXX/AP.

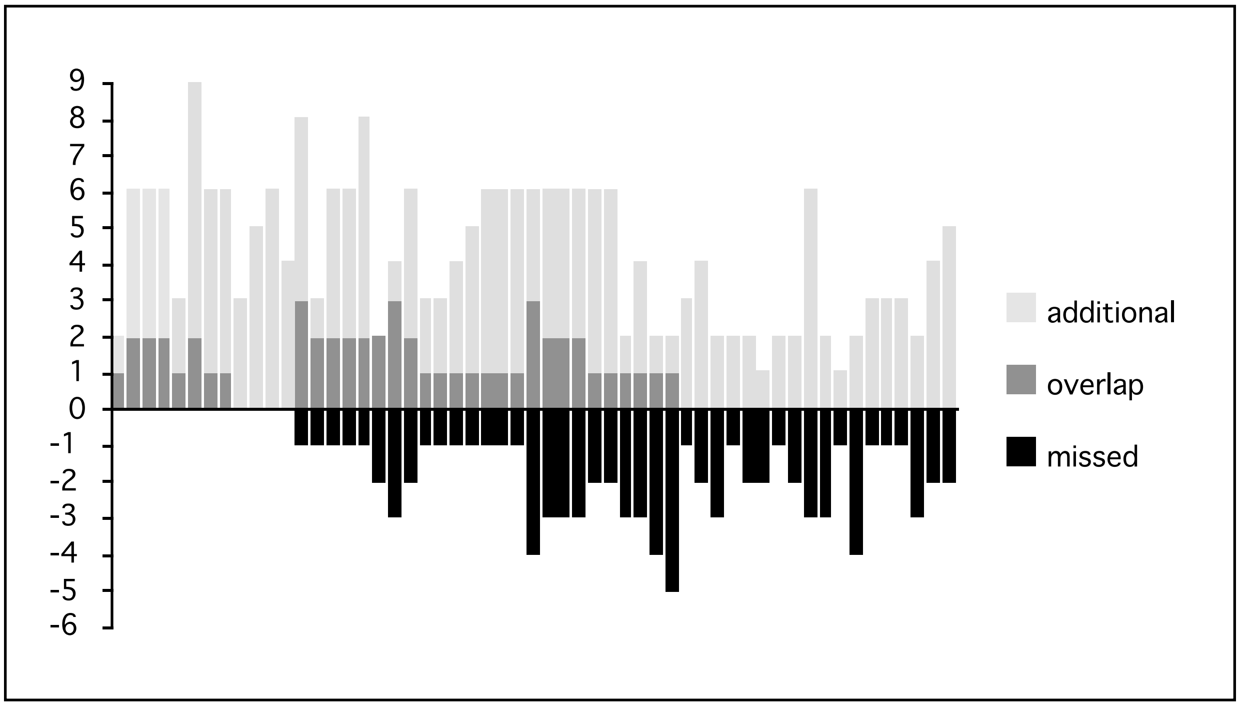

Figure 7b. Histogram showing for each case the number of items that were in the overlap, missed and additional categories for the *aggregated-theories* argument piece for BankXX/AP.

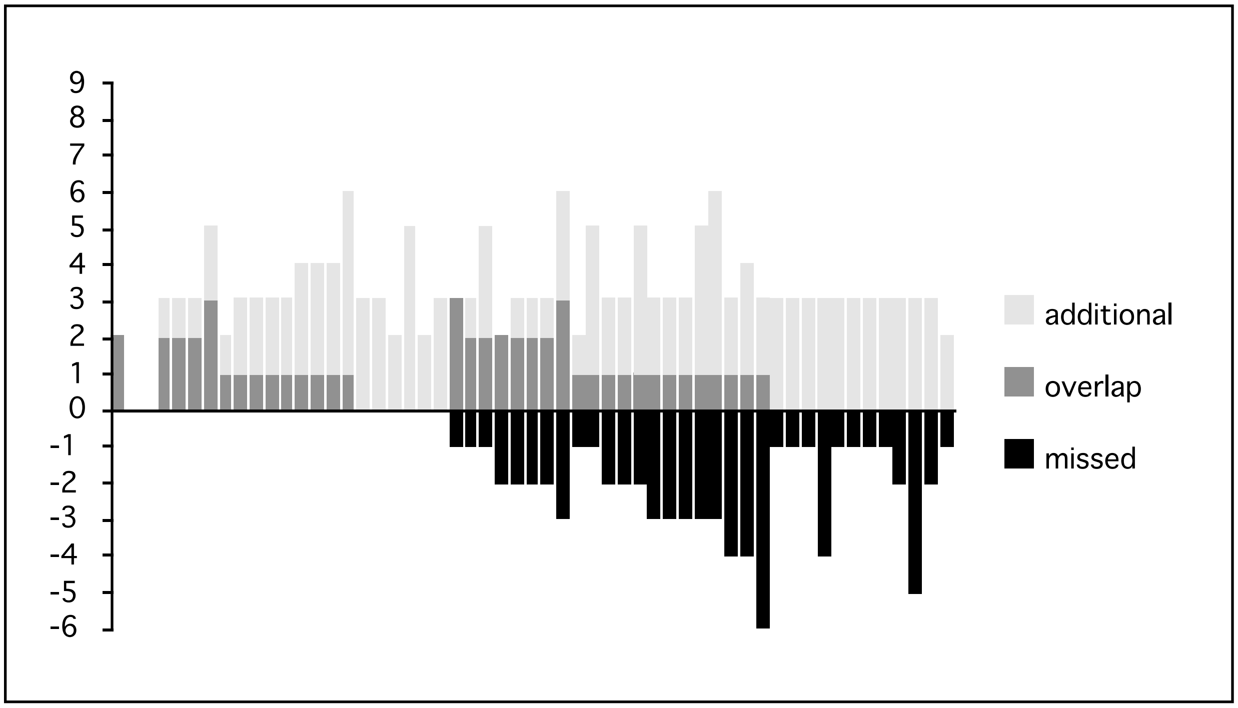

Figure 7c. Histogram showing for each case the number of items that were in the overlap, missed and additional categories for the *aggregated-pro-cases* argument piece for BankXX/AP.

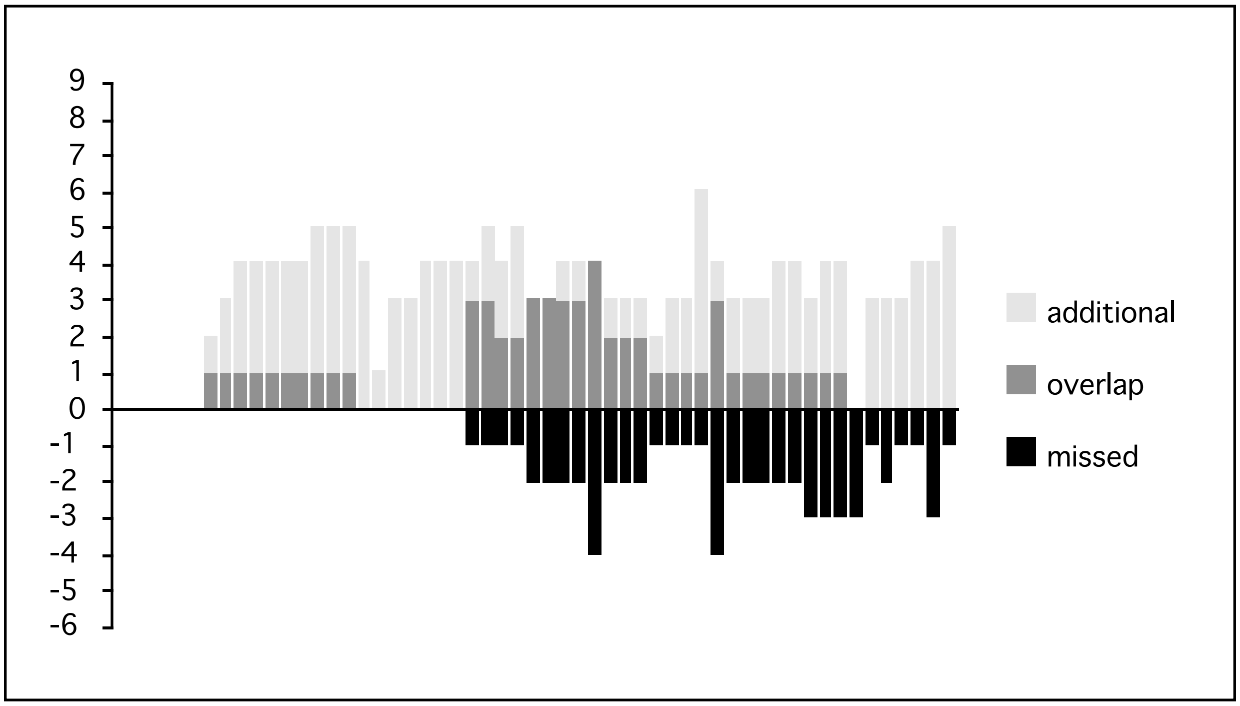

Figure 7d. Histogram showing for each case the number of items that were in the overlap, missed and additional categories for the *aggregated-con-cases* argument piece for BankXX/AP.

Each summarizes BankXX/AP's performance on all 55 cases treated as *de novo* problems (case names are omitted). The vertical axis indicates the number of items retrieved. Everything above the zero line represents items retrieved by BankXX/AP, with the dark gray part of a bar representing those retrieved by BankXX/AP and mentioned in the opinion. The lightly shaded portion represents items retrieved by BankXX/AP that were not mentioned in the opinion. The black part of the bar below the zero line represents items mentioned in the opinion that were not retrieved by BankXX/AP. Graphically, *precision* is the proportion of gray out of the total bar above the zero; *recall* is the proportion of gray out of the combined gray and black parts of the bar.

Bars are organized from left to right from highest to lowest recall and within a given recall band from highest to lowest precision, and within that from highest to lowest absolute number of objects in the overlap between BankXX's output and the hand-coded answer. This ordering means that cases do not appear in the same order in the histograms. In addition, some bars toward the far left may be empty due to an empty hand-coded answer and/or empty BankXX output since we are using the convention here that the undefined value of 0/0 is set to 1.

Notice that many cases display low levels of *overlap*, which can translate into low values for recall and precision. This often occurred because many of the opinions cite very small numbers of cases or theories. (See Section 3.5.) This sparseness makes it hard for there to be much shared between BankXX and the hand-coded answers.

3.2.2 Quantitative precision-recall analysis

Another way to examine the data is to compute precision-recall statistics. Averaging precision and recall over the case base smoothes out the vagaries of any individual scores. Averaged precision and recall results for the BankXX-court comparisons are shown in Table 1 and shown on a precision-recall plot in Figure 8.

Averaged precision & recall percentages across all cases

	AGG'D-PRO-CASES	AGG'D-CON-CASES	LEADING-CASES	AGG'D-THEORIES
BankXX/NT	p=24 (23_{54}) r=76 (72_{47})	p=23 r=84 (79_{42})	p=46 (45_{54}) r= 99 (99_{43})	p=23 r=53 (49_{51})
BankXX/AP	p=32 (30_{53}) r=55 (48_{47})	p=42 (33_{48}) r=64 (53_{42})	p=46 (45_{54}) r=99 (99_{43})	p=21 r=43 (38_{51})
BankXX/AF	p=30 (26_{52}) r=56 (48_{47})	p=33 (26_{50}) r=63 (52_{42})	p=55 (52_{51}) r=78 (72_{43})	p=26 r=38 (33_{51})

Table 1. Averaged precision and recall values with undefined ratios set to 1. Averages taken only when a value is defined are given in parentheses; the number used in these average is given subscripts.

Note, when undefined precision/recall ratios are set to 1, a few individual scores arguably can be said to be "artificially" high. Thus, to give a more conservative measure of performance—one that probably undervalues BankXX performance on empty cases—the table also gives averages when undefined values are excluded from the averages.[21]

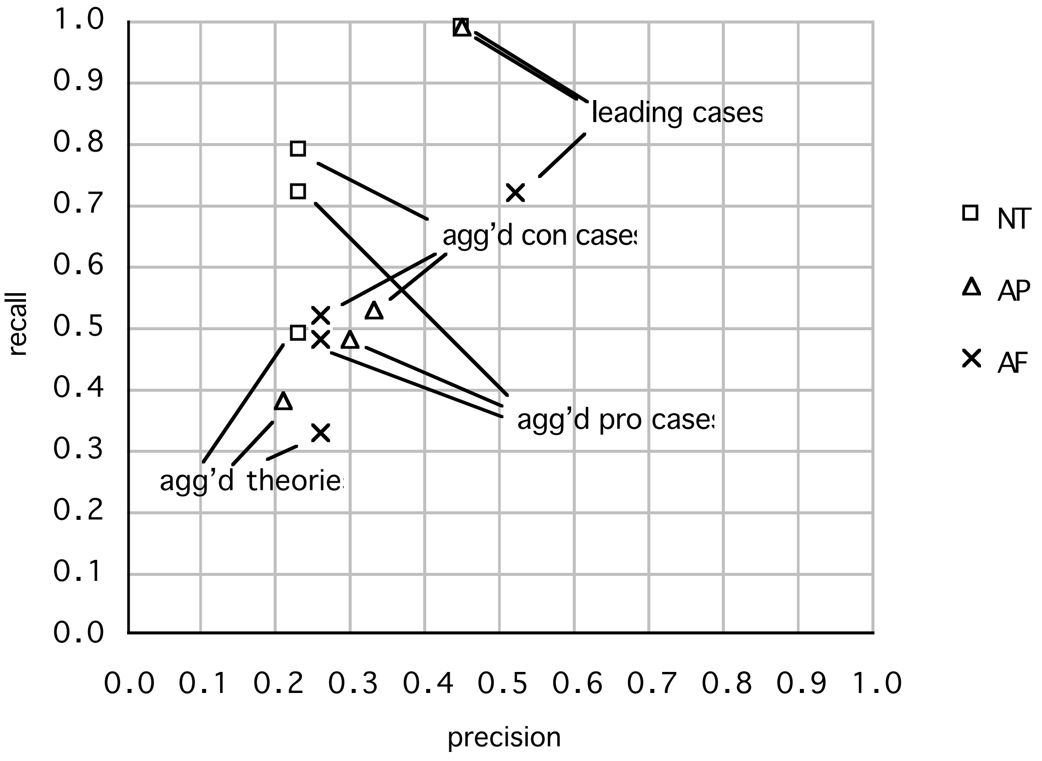

Figure 8. Averaged precision and recall values for the four aggregated argument pieces for BankXX/NT, BankXX/AP, and BankXX/AF. Undefined ratios excluded from the average.

Recall scores for BankXX/NT lie above those for both BankXX/AP and BankXX/AF. In fact, with one very small exception,[22] all the averaged recall values for each of the four aggregated argument pieces for BankXX/NT lie above the corresponding values for BankXX/AP which in turn lie approximately at or a little above those for BankXX/AF. Thus one can say:

r(BankXX/NT) ≥ r(BankXX/AP) ≥ r(BankXX/AF)
(For nearly all argument pieces)

(There are no exceptions when undefined values are ignored.)

[21]The presence of so many empty or nearly empty cases motivated us to re-examine our results by considering only "meaty" cases, those cases having hand-coded answers with numbers of items above a certain threshold. (N.B., p=0 on an empty answer.) See Section 3.4 below. It also motivated us to examine performance with respect to degrees of sparseness. See Section 3.5 below.

[22]*Aggregated-pro-cases* with BankXX/AF has recall 1% higher than with BankXX/AP. See Table 1.

Precision scores for BankXX/NT lie below those for both BankXX/AP and BankXX/AF, with one exception.[23] While precision for *leading-cases* and *aggregated-theories* with BankXX/AP is less than with BankXX/AF, on the two argument pieces concerning cases—*aggregated-con-cases* and *aggregated-pro-cases*—the reverse is true. Thus, there is not a monotonic relation among precision scores as there is with recall scores. (These observations hold regardless of the way in which undefined scores are handled.) Thus:

p(BankXX/NT) ≤ p(BankXX/AP) and p(BankXX/AF)
(For nearly all argument pieces)

Between BankXX/NT and BankXX/AP there are big upwards jumps in precision for *aggregated-con-cases*—where it nearly doubles—and *aggregated-pro-cases* accompanied by big falls in recall. The precision-recall values for *leading-cases* are identical for BankXX/NT and BankXX/AP. For *aggregated-theories,* there is downward change in both precision and recall. (See Table 1.)

Between BankXX/AP and BankXX/AF, significant differences (about 20% relative change) occur for *leading-cases*: precision jumps up and recall down. The same relative changes are seen with *aggregated-theories* but they are not as large for recall. Precision and recall scores on both *aggregated-pro-cases* and *aggregated-con-cases* present exceptions to the general pattern of increasing precision accompanied by decreasing recall. There is a minuscule change in recall but a significant drop in precision on *aggregated-con-cases*. Scores on *aggregated-pro-cases* are nearly the same.

With respect to the individual argument pieces, in all three configurations of BankXX, highest recall was found for *leading-cases*, followed by *aggregated-con-cases, aggregated-pro-cases* then *aggregated-theories*. The same ordering is true except for one small exception,[24] for precision. Thus (no matter how undefined ratios are handled):

leading-cases ≥ aggregated-con-cases ≥ aggregated-pro-cases ≥ aggregated-theories.

We interpret the high recall and precision on *leading-cases* as follows. Since the same small group of leading cases is cited repeatedly in the opinions—that's what makes them leading cases—the chance that a given leading case harvested by BankXX is also mentioned in the opinion is very high. In other words, a harvested leading case is likely to be in the overlap with the hand-coded answer. Furthermore, since leading cases are so well woven into the case-domain graph, BankXX is unlikely to miss any. For the other argument pieces, there is a much wider range in the amount of information mentioned in the opinions and in how well they are tied into the network of domain knowledge: hence a much wider range in precision-recall scores.

[23]*Aggregated-theories* with BankXX/NT has precision 2% higher than with BankXX/AP. See Table 1.

[24]*Aggregated-pro-cases* with BankXX/NT.

We believe that low precision and recall scores on *aggregated-theories* are due to the small OVERLAP between BankXX and the court opinions on *aggregated-theories* in concert with large ADDITIONAL and MISSED sets in the comparisons (see e.g., Fig. 7b). We feel several factors contribute to this situation: (i) many opinions do not cite a large number of theories; (ii) there is a relatively high number of legal theories (18) in BankXX's corpus of 55 cases; and (iii) many theories are very similar (in terms of their defining factors) and are nearly "synonymous" since they refer to the same legal ideas.

Synonymy of theories means that both BankXX and the opinion could be citing essentially the same theory without using the same name. Since the match used in the comparisons is totally literal, such a citation would not show in the overlap, and thus would hurt both recall and precision. The program receives no credit for retrieving a useful but uncited or synonymous theory. A metric to measure the similarity of the retrieved theory to the one actually applied by a court would be required to make this determination.[25]

One way to examine the data further is to look at the results from the 55 cases sorted according to ranges of achieved performance. For instance, one can look at the number of cases that have achieved a recall score of 90% or better on a particular argument piece. Figure 9a presents a graph that shows the number of cases achieving recall performance at or above decile intervals: 0%, 10%, 20%, etc. for BankXX/AP for the four aggregated argument pieces. Figure 9b shows the analogous graph for precision. The distribution of results was similar with the other two evaluation functions.

With regard to recall, the *leading-cited-cases* argument piece, uniformly high recall performance is observed; almost all the problem cases achieved a near perfect level of recall on leading cases. Except for *leading-cases*, levels of recall higher than 0.7 are not frequently observed. For precision, fewer cases achieve similar levels of performance.

Looking at results achieved by 50% of the cases (i.e., at the 27-28 number-of-cases level) provides the medians:

	Median Recall	**Median Precision**
aggregated-theories	50% or better	20% or better
aggregated-pro-cases	55% or better	30% or better
aggregated-con-cases	65% or better	30% or better
leading-cases	95% or better	45% or better

[25]For instance, one could measure similarity in terms of overlaps of defining factors or overlaps of the sets of cases that have applied them.

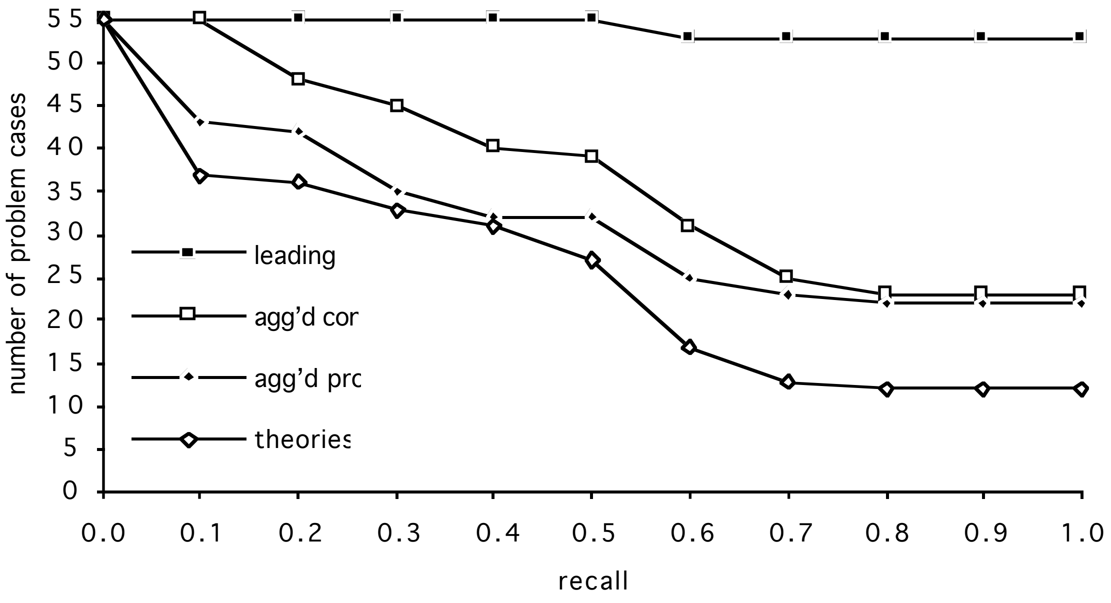

Figure 9a. Graph showing the number of problem cases that display a given or better recall level on the four aggregated argument pieces with BankXX/AP. Recall values of 0/0 are defined to be 1.

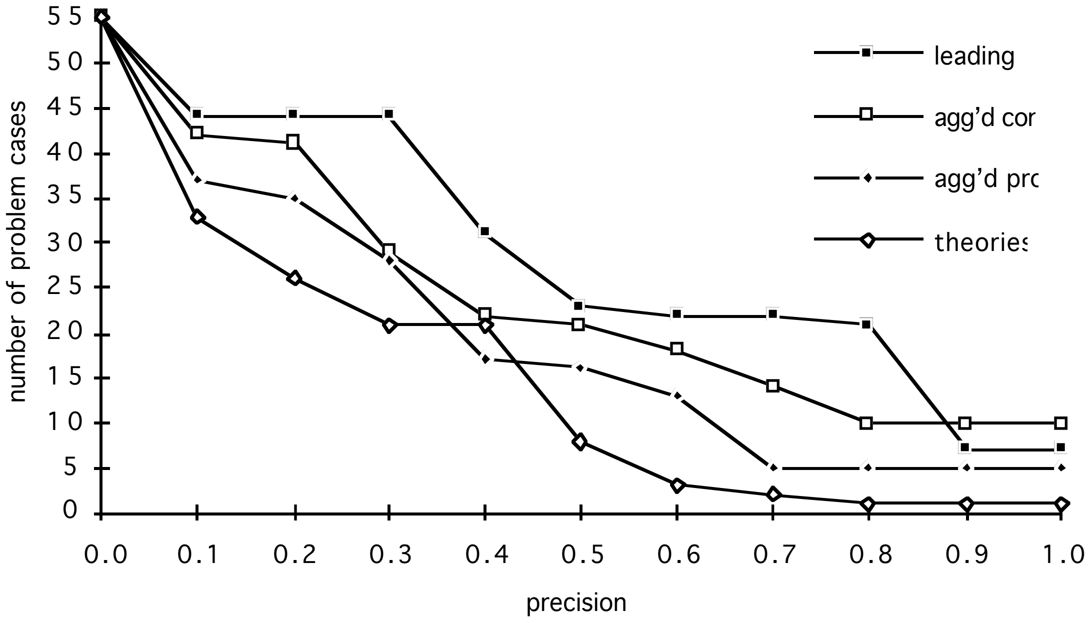

Figure 9b. Graph showing the number of problem cases that display a given or better precision level on the four aggregated argument pieces with BankXX/AP. Precision values of 0/0 are defined to be 1.

Comparing these medians with the averages (in Table 1) gives an indication of the distribution of scores. Except for recall on *aggregated-theories*, and precision on *aggregated-*

con-cases, the medians are about equal to the averages, which means that the distribution of scores is not particularly skewed. For theories, there is more of a concentration of low recall scores; for con cases, a higher concentration of high precision scores. In general, for precision, there is a shift to lower median scores, which indicates that there is a greater concentration of cases with lower precision scores. (See Fig. 9b vs. Fig. 9a)

Summary of Section 3.2

Among the three configurations of BankXX, there was a general pattern for precision and recall scores, averaged over all 55 cases in the BankXX corpus:

***Precision*(BankXX/NT) ≤ *Precision*(BankXX/AP) and *Precision*(BankXX/AF)**
***Recall*(BankXX/NT) ≥ *Recall*(BankXX/AP) ≥ *Recall*(BankXX/AF)**

These inequalities held with rare exception across all argument pieces no matter which way undefined values were handled.[26] For precision, there were too many significant exceptions to allow us to observe a three-way monotonic pattern.[27] Note, in many cases the differences between the scores were small.

Thus, across all argument pieces, BankXX/NT displayed the highest levels of recall and the lowest levels of precision while BankXX/AP and BankXX/AF displayed the highest levels of precision and lowest recall. This shows a trade-off in precision and recall between BankXX/NT and BankXX/AP and/or BankXX/AF. In other words, BankXX/NT was not very discriminating but produced good coverage of the hand-coded answer whereas BankXX/AP and BankXX/AF were discriminating but at the expense of overall coverage.

On individual argument pieces, for both precision and recall scores, on all three configurations of BankXX, we found a consistent pattern of:[28]

leading-cases ≥ aggregated-con-cases ≥ aggregated-pro-cases ≥ aggregated-theories

In addition, BankXX/NT tended to produce the same answer for every problem case, whereas BankXX/AP and Bank/AF were quite problem-sensitive.

[26]There was one small exception to the precision pattern: *aggregated-theories* under BankXX/AP. There was one small exception to the recall pattern: *aggregated-pro-cases* under BankXX/AF.

[27]There are 3 cells presenting exceptions for precision: in BankXX/AP, *aggregated-theories*, and in BankXX/AF, *aggregated-con-cases* and *aggregated-pro-cases*.

[28]There was one small exception: *aggregated-pro-cases* under BankXX/NT.

3.3 Qualitative Observations: A More Detailed Look

Our overall qualitative observations are also borne out by close examination of the items in the OVERLAP, MISSED, and ADDITIONAL categories produced for the BankXX-court comparisons. The observations that follow were made by closely inspecting output from each of the three BankXX configurations on the set of the twenty most frequently cited cases, the so-called *Top 20* cases.

For each of the four aggregated argument pieces, we always found that the sets of items MISSED by BankXX/NT are contained in those MISSED by BankXX/AP as well as in those MISSED by BankXX/AF. Close inspection shows that there were not significant differences for the theories and leading cases missed in the three versions but that there were big differences in pro and con case categories.

In all of the Top20 cases, for the *aggregated-theories* and *leading-case* argument pieces, there was actually a monotonic chain of containments. Theories or leading cases MISSED by BankXX/NT were a subset of those MISSED by BankXX/AP which were in turn a subset of those MISSED by BankXX/AF.

For *aggregated-theories,* the three sets of MISSED theories were very often identical (in 16 of the Top20 cases) and always nearly the same, and thus, the monotonic chain was trivial. Each version of BankXX on a given Top20 case missed the same two or three theories.

For *leading-cases*, the three sets of MISSED cases were identical a little more than half the time (in 12 of the Top20 cases), and in fact, in 11 cases, it was the empty set across the baord. That is, on somewhat more than half of the Top20 cases, no version of BankXX missed any leading cases. In the remaining cases, BankXX/NT and BankXX/AP missed no leading cases but BankXX/AF missed just one or two. Thus, on the *leading-cases* argument piece, BankXX/AF tended to miss a few more leading cases than the other two configurations, which missed none.

To summarize: for *aggregated-theories* and *leading-cases*, all three versions of BankXX missed about the same items: virtually no leading cases were missed and the same small set of theories were missed across the board. This is reflected in the nearly perfect recall scores for *leading-cases* and the low recall scores for *aggregated-theories*. (See Table 2 for precision-recall scores on Top20 cases.)

In the *aggregated-pro-cases* and *aggregated-con-cases* argument pieces, there were some differences in the sets of cases missed. In well over half of the Top20 cases (70%), the monotonically increasing chain held.[29] In about half the Top20 cases, all three versions of BankXX missed exactly the same cases. In about a third of the Top20 cases, no pro or con

[29]The chain held in 14 of the Top20 cases for both the pro and con case categories. However, not the same 14 cases.

cases were missed in any version.[30] In several additional instances, BankXX/NT and BankXX/AF missed none but BankXX/AP did, thus, breaking the chain of containments. On nearly three-quarters of the Top20 cases, BankXX/NT missed no pro or con cases; this was significantly better than either BankXX/AP or BankXX/AF.[31]

To summarize: for *aggregated-pro* and *aggregated-con-cases*, all three versions did rather well in terms of not missing many cases. BankXX/NT did the best. BankXX/AP and BankXX/AF were not that different from each other.

These observations are summed up in Figure 10.

For all argument pieces:

Missed-items(BankXX/NT) $\subseteq$ Missed-items(BankXX/AP)
Missed-items(BankXX/NT) $\subseteq$ Missed-items(BankXX/AF)

For all leading-cases and theories instances and 70% of the pro and con case instances:

Missed-items(BankXX/NT)
$\subseteq$ Missed-items(BankXX/AP)
$\subseteq$ Missed-items(BankXX/AF)

Figure 10. Relationship among the items MISSED by BankXX/NT, BankXX/AP, and BankXX/AF.

One would expect to find the complementary relations between sets of additional cases. We did.

For nearly all of the Top20 cases, we, found that the set of ADDITIONAL items found by BankXX/NT contained those found by BankXX/AP as well as those found by BankXX/AF.[32] The sets of ADDITIONAL items were rarely empty, unlike the sets of MISSED items. The uniform presence of non-empty ADDITIONAL sets demonstrates BankXX's drive to retrieve information.[33]

As for MISSED items, we found that there was often a monotonic chain: ADDITIONAL items found by BankXX/NT contained those found by BankXX/AP which in turn contained those found by BankXX/AF. It was most robust on *leading-cases* and *aggregated-theories*. It held on all 20 Top20 cases for the *leading-cases* category. It held on three-quarters (15) of the 20 Top20 cases in the *aggregated-theories* category, where again, there was a steady stream of 2 or 3 additional theories and a great deal of similarity between the sets ADDITIONAL theories

[30]For the picture across the entire 55 case corpus (with BankXX/AP), see Figs. 7c and 7d.

[31]BankXX/NT missed no pro or con cases in 14 Top20 cases. BankXX/AP and BankXX/AF each missed no pro cases in 9 Top20 cases, and missed no con cases in 8 and 9 Top20 cases, respectively.

[32]There were only 4 exceptions to this (out of 80 possible data points). They only occurred for the *aggregated-pro-cases* argument piece.

[33]The picture is similar across the entire 55 case corpus. See Fig. 7 (for BankXX/AP).

found. We note that for *leading-cases*, BankXX tended to find all 5 leading cases, and thus any leading case not mentioned in the hand-coded answer showed up on the ADDITIONAL set and lowered precision. Since no case mentioned all 5 leading cases in its opinion, this means that BankXX often achieved less than 100% precision.[34]

BankXX/NT always found ADDITIONAL *aggregated-pro-cases* and *aggregated-con-cases.* In nearly every Top20 case, BankXX/AP and BankXX/AF found some ADDITIONAL pro cases. In more than half the Top20 cases, ADDITIONAL cases found by BankXX/NT was much larger than those sets for either BankXX/AP or BankXX/AF. ADDITIONAL con cases occurred more than ADDITIONAL pro cases.[35]

For the *aggregated-pro-cases* and *aggregated-con-cases* categories, the monotonic chain of nesting ADDITIONAL items was found less often. It occurred in about half the Top20 cases. For these, it often happened that those cases found additionally by BankXX/AP were the same as those found additionally by either BankXX/NT or BankXX/AF. In only a very few cases, were the sets of ADDITIONAL cases found with BankXX/NT and BankXX/AF the same.

These observations are summed up in Figure 11.

For all argument pieces:

Additional-items(BankXX/NT) $\supseteq$ Additional-items(BankXX/AP)
Additional-items(BankXX/NT) $\supseteq$ Additional-items(BankXX/AF)

For all leading-cases, 75% of theories, and about 50% of pro and con cases:

Additional-items (BankXX/NT)
$\supseteq$ Additional-items(BankXX/AP)
$\supseteq$ Additional-items(BankXX/AF)

Figure 11. Relationship among sets of ADDITIONAL items found by BankXX/NT, BankXX/AP, and BankXX/AF.

In summary, data on MISSED and ADDITIONAL sets shows that BankXX/NT is less selective than either BankXX/AP and BankXX/AF.[36] There are larger ADDITIONAL and smaller MISSED sets with BankXX/NT than with either BankXX/AP or BankXX/AF.

We posit two reasons for this:

[34]See Table 3 in Section 3.5 for numbers of cases. For example, if in a problem case, BankXX found all 5 leading cases and the answer only mentioned 4, precision would be .80; if only 3 were mentioned, precision would be .6, etc.

[35]A null set of ADDITIONAL cases occurred rarely. Never for BankXX/NT. For BankXX/AP, only 1 time for pro cases and 5 times for con cases. For BankXX/AF, only 3 times for pro cases and 5 times for con cases.

[36]In many cases, BankXX/NT and BankXX/AF harvested very similar sets of cases. N.B., even though they usually followed quite different paths through the case-domain graph in doing so (as shown in the order and content of their OPEN and CLOSED lists).

1. BankXX/NT uses a less discriminating evaluation function than the others;
2. BankXX/NT has no fill limits on the harvesting of items for argument pieces as BankXX/AP does and no significant computational burden as BankXX/AF does.

All three configurations of BankXX tend to exhibit similarities in their sets of MISSED and ADDITIONAL cases for the *leading-cases* and *aggregated-theories* argument pieces. The most variation occurs on *aggregated-pro-cases* and *aggregated-con-cases*. BankXX's persistent tendency to find ADDITIONAL items across all configurations and all argument pieces is reflected in the low precision scores that BankXX received across the board. (See Table 2.)

Our qualitative analysis has shown that BankXX/NT and BankXX/AP or BankXX/AF exhibit a classic trade-off in selectivity and coverage, especially in the pro and con case categories. This is the same trade-off that was seen quantitatively in precision-recall values in the last section (See Table 1.).

Summary for Section 3.3

Detailed analysis of the MISSED and ADDITIONAL cases for each of the 20 cases in the set of Top20 cases showed that BankXX/NT misses less and harvests more from the hand-coded answers than either BankXX/AP or BankXX/AF:

For all argument pieces:

$$\text{Additional-items(BankXX/NT)} \supseteq \text{Additional-items(BankXX/AP)}$$
$$\text{Additional-items(BankXX/NT)} \supseteq \text{Additional-items(BankXX/AF)}$$

and

$$\text{Missed-items(BankXX/NT)} \subseteq \text{Missed-items(BankXX/AP)}$$
$$\text{Missed-items(BankXX/NT)} \subseteq \text{Missed-items(BankXX/AF)}$$

In many situations there is a three-fold nesting. (See Figs. 10 and 11 for details.)

MISSED and ADDITIONAL sets for *leading-cases* and *aggregated-theories* are similar in all three configurations of BankXX. There was much more variation for *aggregated-pro-cases* and a*ggregated-con-cases*.

The detailed analysis exhibits qualitatively what was seen in the quantitative precision-recall observations of the previous section: There is a trade-off in selectivity and coverage between BankXX/NT and both BankXX/AP and BankXX/AF.

3.4 A Closer Look at the Quantitative Analysis: The Subset Experiments

Our goal in the analysis in this section is to investigate BankXX's performance on various types of cases, like appellate cases, and determine whether any generalizations can be made with respect to them. In particular, we wanted to know whether the <u>overall</u> precision-recall patterns of the previous sections held on these particular subsets as well. We especially wanted to examine a group of cases that were not sparse or empty (and thus suffered from problematic precision and recall values). The effect of sparseness is examined in detail in the next section.

In this section, we re-examine BankXX-court comparison data with respect to four different subgroups of cases:

1. ***The Top 20*** —the set of twenty cases most highly ranked cases according to our study of leading citations.

2. ***Meaty cases***—the set of cases defined as having in their hand-coded answers: (a) 3 or more *aggregated-pro-cases*; and
   (b) 3 or more *aggregated-con-cases*; and
   (c) 1 or more *aggregated-theories*; and
   (d) 3 or more *leading cited cases.*
   There are 14 meaty cases in our case base.

3. ***Appellate cases***—there are 14 appellate cases in our case base.

4. ***All***—the entire corpus of 55 cases.

In Table 2 we show averages taken with respect to these different subsets. As before we show averages taken both ways: averages with undefined values not included are shown in parentheses. There are several observations that can be made about these data. Many involve monotonicity.

3.4.1 Analysis with respect to subsets

In this subsection we analyze our data <u>with respect to subsets</u> and investigate patterns that depend on them—***row monotonicity***—in Table 2. For precision, there is a rough monotonicity. For recall, there is no real pattern. Further, for precision, the maximum score is always—for all BankXX configurations, for all argument pieces—achieved on the *Meaty* cases. The situation with recall is very mixed: there is no subset of cases that clearly outscores the others.

<u>Precision:</u> The rows in Table 2 support an overall impression that precision scores on *Meaty* cases are higher than those on *Top20* cases or *Appellate* cases, which in turn are higher than those on *All* cases:[37]

$$p(Meaty) \geq p(Top20), p(Appellate) \geq p(All)$$

(This is true no matter how undefined values are handled.)

In many rows, there is a "full" chain of inequalities:

$$p(Meaty) \geq p(Top20) \geq p(Appellate) \geq p(All)$$

There are several exceptions to the full pattern. For unparenthesized values, it holds for half of the dozen rows—and 30 of the 36 individual comparisons—in the precision portion of Table 2.[38] There are only two exceptions when undefined values are ignored (i.e., with the parenthesized values).[39]

In addition, for most all of the rows in the precision portion of Table 2, there is a general pattern of a significant drop off (over 10%) between *Meaty* and *Top20* cases,[40] the middle two cells in the same ballpark, and another not-so-large drop-off between *Appeals* and *All* cases. The drop-off can be quite dramatic (e.g., over 40% in con cases with BankXX/NT). In general the drop-offs decrease as one proceeds down the blocks of Table 2 representing the different configurations of BankXX.

Examining the raw outputs that underlie these data, we found that a low out-of-pattern score can be due to the presence of a high proportion of items with p=0 precision scores in the cell.[41] For instance, relatively low scores in all the *aggregated-theories* cells for *Top20* cases are due to the presence of relatively many more p=0's in the *Top20* collection than in the other collections.[42]

[37]The 3 exceptions are: *aggregated-theories* with BankXX/NT and *aggregated-con-cases* with BankXX/AF.

[38]There are 6 exceptional cells. In each configuration, there is an out-of-pattern low precision score for *aggregated-theories* on *Top20* cases. In addition, there are exceptions on *aggregated-pro-cases* and *aggregated-con-cases* with BankXX/AP, and on *aggregated-con-cases* with BankXX/AF.

[39]Both occur with out-of-pattern low scores on *Top20* cases: *leading-cases* with BankXX/AP and *aggregated-pro-cases* with BankXX/AF.

[40]The only exception is with *aggregated-pro-cases* under BankXX/AF where the values are flat.

[41]There are many cases (197) where BankXX gets p=0 scores. 90 of these occur because the hand-coded answer is empty. 107 occur when because hand-coded and BankXX answers are skew. Skewness is associated with sparseness in the hand-coded answers.

[42]The percentage of instances for *aggregated-theories* in the *Top20* collection that score p=0 with BankXX/NT is 50%, whereas in *Meaty*, for instance, it's 29%. For BankXX/AP it is also 50% vs. 36%. For BankXX/AF, it is 55% vs. 36%.

In only one row is a break in row monotonicity due to high scores resulting from setting undefined precision ratios to 1.[43] It occurs in the row for *aggregated-con-cases* under BankXX/AP: the score in the *All* cell is high compared with that for *Appeals*. This is due to a combination of proportionally more boosting by default scores of p=1 in *All* than in *Appeals* and more lowering by p=0 scores in *Appeals* than *All*.[44] An indicator of a potential "boosting" problem is a large difference between the two ways of computing with undefined scores (e.g., the scores 33 vs. 42 in the cell for *aggregated-con-cases* under BankXX/AP on *All*). Note that in this row, the scores not using the undefined ratios—shown in ()'s—are in line with each other.

Recall: With respect to subsets, there is no overall pattern of recall values. There are too many exceptions—at least one per row—for there to be a pattern of monotonicity for recall; however the exceptional values are often only a small bit out of line.

Examination of Table 2 shows that most of the exceptions occur with the *Top20* and *All* subsets. Many of the out-of-pattern scores are either a little too low or too high (within a few percentage points) although a few—especially with *aggregated-theories* on *All* cases—are very dramatic. There is also significant gap between *aggregated-pro-cases* between the *Meaty* and *Top20* subsets, especially in BankXX/AP and BankXX/AF. Because of variation in the exceptions, a simple re-arrangement of the columns (e.g., putting *Top20* cases first) won't establish a monotonic pattern.

As with precision, some high out-of-pattern values are due to the use of 1 as a default value for undefined recall scores.[45] The presence of so many empty hand-coded answers in the *All* category that are not members of the other subset categories accounts for many of the upwards jumps in recall scores present in the *All* column entries.

For instance, under BankXX/AP for *aggregated-contrary-cases*, there are 13 cases with recall score set to 1 (because their hand-coded answers are empty) that don't appear in the preceding columns. Furthermore, the proportion of such boosting r=1 scores is higher in the *All* category than in the other categories except *Top20*, which also shows a high recall score, while lowering due to r=0 scores is relatively less in *All* than in the other categories, except *Meaty* cases, which has no r=0 scores.[46]. This cell is typical of the mixed effects of boosting by p=1 and lowering by r=0 scores.

[43]Undefined precision scores are the result of an empty answer produced by BankXX. This does not happen that much (24 times altogether).

[44]The percentages of boosting p=1 scores are: 20% in *Top20*, 13% in *All*, 8% in *Appeals*, and 0% in *Meaty* cases. The percentages of lowering p=0 scores are: 30% in *Appeals*, 25% in *Top20*, 24% in *All* and 0% in *Meaty* cases.

[45]There are 111 instances of undefined recall scores. These are due to empty hand-coded answers.

[46]The percentages of boosting r=1 scores are: 13 (24%) in *All*, 5 (25%) in *Top20*, 2 (14%) in *Appeals* and 0 in *Meaty*. The percentages of lowering r=0 scores are : 7 (13%) in *All*, 4 (20%) in *Top20*, 3 (21%) in *Appeals*, and 0 in *Meaty*. Thus boosting outweighs lowering in *All* and *Top20*; in *Meaty* it is not an issue.

Certain scores do not exhibit such a balanced mixture of boosting and lowering. For instance, the three elevated scores for *aggregated-theories* on *All* cases are due more to the disproportionate proportion of r=0 scores in all the other cells in the theory row than in a high proportion of boosting scores in *All*.[47]

Summary for Section 3.4.1
Comparisons with respect to subsets ("rows" in Table 2)

In all BankXX configurations for all four argument pieces—that is all rows of the precision portion of Table 2—highest precision is achieved on *Meaty* cases, without exception. With only a few exceptions, the lowest is achieved on *All* cases.[48] *Top20* and *Appellate* cases tend to fall in between. Thus, generally (no matter how undefined values are handled):

$$p(Meaty) \geq p(Top20),\ p(Appellate) \geq p(All)$$

About half of the time there is a full chain of inequalities: precision on *Meaty* cases is higher than precision on *Top20* cases, which in turn is higher than precision on *Appellate* cases, which in turn is higher than precision on *All* cases:

$$p(Meaty) \geq p(Top20) \geq Appellate) \geq p(All)$$

The situation with recall is quite mixed. For instance, *Top20* cases often outscore *Meaty* ones.

[47]For instance, for BankXX/NT, the percentage of boosting r=1 scores in *All* is only 7% (4/55); there are no boosted r=1 scores in the other categories. However the proportion of lowering r=0 scores is: 50% in *Top20* and *Appeals*, 36% in *Meaty* and 30% in *All*. Results for the other two BankXX configurations are nearly identical.

[48]There are 3 exceptions: In BankXX/NT: *aggregated-theories* on *Top20* cases. In BankXX/AF: *aggregated-theories* on *Top20* and *aggregated-con-cases* on *Appeals*. There are no exceptions with BankXX/AP.

3.4.2 Analysis with respect to argument pieces

There is an overall pattern of inequalities with respect to argument pieces—***column monotonicity***—in Table 2 that holds across all subsets of the BankXX corpus.

<u>Precision</u>: With respect to argument pieces, for all four subsets and in all three BankXX configurations, there is an overall pattern of precision values. Precision on *leading-cited-cases* is higher than precision on either of the other two case categories, which, in turn, are higher than precision on theories:

$$p(leading) > p(con), p(pro) \geq p(theories)$$

This pattern holds no matter how undefined values are handled.

On the *Meaty* cases, one can break this down further since higher precision is always achieved on *Con* than on *Pro* cases:

$$p(leading) > p(con) \geq p(pro) \geq p(theories)$$

On the set of *All* cases, there is only one exception to this "full" pattern and it is a very small one (with BankXX/NT). For *Meaty* cases, the differences between the scores on the different argument pieces are always dramatic. For *All* cases this is less so. The pattern does not hold for the *Top20* and *Appeals* subsets where there are a few exceptions (with BankXX/NT and BankXX/AF).

BankXX/AP is particularly "well-behaved" (on unparenthesized scores) with respect to this full pattern of since it holds across all the subsets. BankXX/AF pretty much fits the pattern—with the exception of scores on the *Appeals* cases. With BankXX/NT the reverse seems to hold: higher precision values are achieved—with the exception of *Meaty* cases—on *pro* cases. However, in BankXX/NT, the differences between the scores in the rows for *aggregated-pro-cases* and *aggregated-con-cases* are not large. In all three configurations of BankXX, these differences are greater on *Meaty* and *Top20* cases than on *Appeals* and *All* cases.

<u>Recall</u>: With respect to argument pieces, for all subsets and in all BankXX configurations, there is an overall monotonic pattern. Recall on *leading-cases* is higher than recall on other cases, which, in turn, is higher than recall on *aggregated-theories*:

$$r(leading) > r(con), r(pro) > r(theories)$$

This pattern holds for all four subsets of the BankXX corpus in all three BankXX configurations.

For the most part, the pattern can be refined in most cases to be: higher recall on *aggregated-con-cases* than on *aggregated-pro-cases:*

$$r(leading) > r(con) \geq r(pro) > r(theories)$$

There are no exceptions to this pattern for any of the subsets under BankXX/NT, but there are exceptions under BankXX/AP and BankXX/AF with *aggregated-pro-cases* on *Top20* and *Appeals*.

Summary for Section 3.4.2
Comparisons with respect to argument-pieces ("columns" in Table 2)

For all BankXX configurations, for all four subsets of cases, for both precision and recall, highest performance is achieved on *leading-cases*, lowest on *aggregated-theories*, and in between on other cases (no matter how undefined values are handled):

leading-cases > agg'd-con-cases, agg'd-pro-cases ≥ agg'd-theories

For all configurations of BankXX on *Meaty* and *All* cases, there is generally a full pattern of scores for both precision and recall (no matter how undefined values are handled):[49]

leading-cases > aggregated-con-cases ≥ aggregated-pro-cases ≥ aggregated-theories

There are many exceptions to this pattern in the *Top20* and *Appeals* subsets.

BankXX/NT maintains the full pattern on all subsets for recall values and only on *Meaty* for precision values.

BankXX/AP maintains the full pattern on all subsets for unparenthesized precision values and only on *Meaty* and *All* cases for recall values.

BankXX/AF maintains the full pattern for precision and recall values only on *Meaty* and *All* .

[49]There are no exceptiuons with *Meaty* cases, and 1 exception with BankXX/NT in the *All* column.

3.4.3 Analysis with respect to configurations

Precision: For all four argument pieces and all four subsets of cases, there is a general pattern that precision is lower with BankXX/NT than with either BankXX/AP or BankXX/AF:

p(BankXX/NT) ≤ p(BankXX/AP), p(BankXX/AF).

There are two very small exceptions to this pattern.[50] Since there is not a consistent pattern between BankXX/AF and BankXX/AP, there is no clear three-fold monotonic pattern.

Recall: For all four argument pieces and all four subsets of cases, recall is always higher on BankXX/NT than with either BankXX/AP or BankXX/AF:

r(BankXX/NT) ≥ r(BankXX/AP), r(BankXX/AF).

Since there are four occasions where recall on BankXX/AF is higher than with BankXX/AP, there is no clear three-fold monotonic pattern, although the pattern that there is more prevalent than with precision.

Summary for Section 3.4.3
Comparisons with respect to configurations ("blocks" in Table 2)

It is a robust result—**across all four argument pieces and all four subsets of cases**—that BankXX/NT achieves higher recall but lower precision than either BankXX/AP or BankXX/AF:

p(BankXX/NT) ≤ p(BankXX/AP), p(BankXX/AF)
r(BankXX/NT) ≥ r(BankXX/AP), r(BankXX/AF).

Thus, there is a classic recall-precision trade-off between BankXX/NT and BankXX/AP and between BankXX/NT and BankXX/AF.

[50] In BankXX/AP: *aggregated-theories* with *All* cases. In BankXX/AF: *aggregated-theories* on *Top20* cases.

NT-Precision

	Meaty	Top20	Appeals	All
Leading Cases	81	52 (49_{19})	52	46 (45_{54})
Con Cases	45	26	25	23
Pro Cases	38	29	29	24 (23_{54})
Theories	26	22	26	23

NT-Recall

	Meaty	Top20	Appeals	All
Leading Cases	100	97 (96_{15})	100 (100_{11})	99 (99_{43})
Con Cases	83	85 (80_{15})	84 (82_{12})	84 (79_{42})
Pro Cases	77	81 (76_{17})	79 (78_{13})	76 (72_{47})
Theories	39	24	23	53 (49_{51})

AP-Precision

	Meaty	Top20	Appeals	All
Leading Cases	81	52 (49_{19})	52	46 (45_{54})
Con Cases	63	47 (34_{16})	38 (34_{12})	42 (33_{48})
Pro Cases	45	35	38	32 (30_{53})
Theories	26	22	26	21

AP-Recall

	Meaty	Top20	Appeals	All
Leading Cases	100	97 (97_{15})	100 (100_{11})	99 (99_{43})
Con Cases	53	59 (46_{15})	53 (45_{11})	64 (63_{42})
Pro Cases	44	63 (53_{16})	56 (53_{12})	55 (48_{47})
Theories	28	24	23	43 (38_{51})

AF-Precision

	Meaty	Top20	Appeals	All
Leading Cases	92	65 (56_{15})	57 (53_{13})	55 (52_{51})
Con Cases	52	46 (29_{16})	32 (27_{12})	33 (26_{50})
Pro Cases	39	39 (36_{19})	38 (37_{12})	30 (26_{52})
Theories	31	21	28	26

AF-Recall

	Meaty	Top20	Appeals	All
Leading Cases	71	71 (62_{15})	71 (67_{12})	78 (72_{43})
Con Cases	61	58 (44_{15})	58 (51_{12})	63 (52_{42})
Pro Cases	44	61 (51_{16})	58 (55_{13})	56 (48_{47})
Theories	27	16	21	38 (33_{51})

Table 2. Precision and Recall values broken out by BankXX configuration, aggregated argument piece, and special subsets of cases. All values are averages. There are 14 *Meaty* , 20 *Top20*, 14 *Appeals* and 55 *All* cases. Values in parentheses do not include undefined values.

3.5 Chasing down the effect of sparse cases

After examining all the foregoing data, we decided to pursue the question of determining the effect of sparseness on BankXX performance. We had noticed that BankXX performed well on the *meaty* cases (See Section 3.4.2.) Our hypothesis was that BankXX performance was correlated with sparseness, that is, BankXX performed better on cases having more items in the hand-coded answers than on cases having fewer, or no, items.

To explore this idea, we defined a simple measure of sparseness for an individual aggregated argument piece of an individual case as:

content-count = number of items in the hand-coded answer[51]

Meaty cases have a content-count of at least 3 in the three argument pieces containing cases and at least 1 in *aggregated-theories* (see Section 3.4). With respect to a particular argument piece, we can group cases into sets of equally sparse—or contentful—answers. For instance, for the *aggregated-pro-cases* argument piece, 8 of the 55 cases in the BankXX case base have content-count of 2, that is, there are 2 same-side cases in their answers. The distribution of cases according to content-count is given in the Table 3. It is the same, of course, regardless of which version of BankXX is run since it only depends on the hand-coded answer.

content-count	0	1	2	3	4	5	6	7	8	9
pro	8	17	8	6	11	3	1	1	0	0
con	13	14	5	9	8	4	0	1	1	0
leading	12	17	10	1	15	na	na	na	na	na
theories	4	11	17	10	6	4	2	1	0	0

Table 3. Distribution of cases according to content-count levels for each aggregated argument piece.

We averaged precision and recall scores over cases at each content-count level, argument piece by argument piece, that is, over sets of cases affiliated with each cell in Table 3, and then plotted these averaged scores against the content-count level. Since level 0—that is, where there are no items in the hand-coded answers—results in an undefined recall ratio (i.e., 0/0), and thus distorts the averages, no matter how one copes with it, we omitted it.[52] This results in eight plots for each version of BankXX. See Figure 12 below.

[51]Recall, hand-coded answers only include items found in both the opinion and the BankXX knowledge-base.

[52]For the most part, it also results in precision of 0.

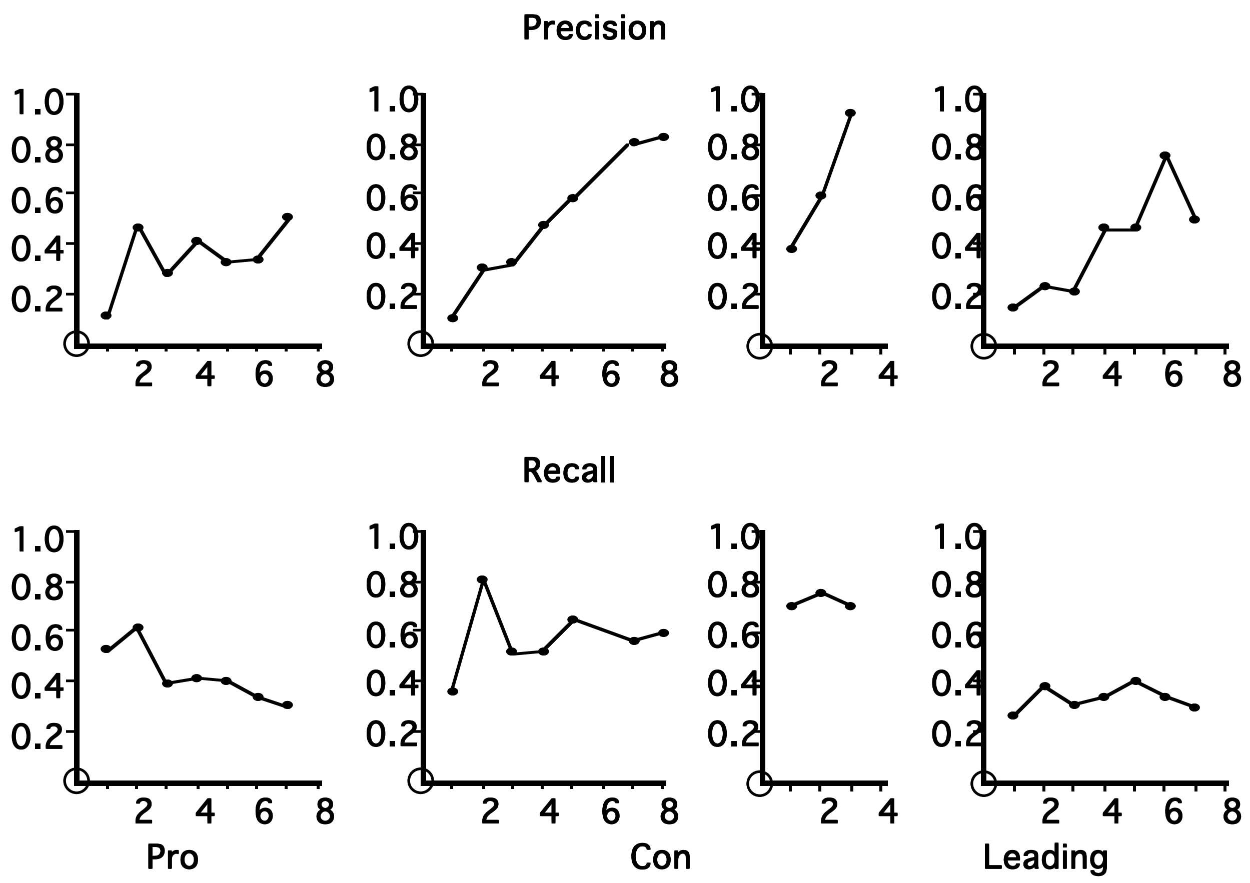

Figure 12. Plots of precision and recall averaged over sets of equal sparseness for BankXX/AF for all four aggregated argument pieces. (Undefined values were excluded.) The y-axis gives the precision or recall scores; the x-axis, the sparseness levels.

While there are not a lot of data points on these plots, all plots show that precision increases with increased content-level, and recall stays flat or falls off slightly. That is, BankXX performs better on cases with more items in the hand-coded answer. This is not a total surprise, of course, since BankXX was designed to operate in exactly this sort of situation where there are too many items for each to be explored, that is, one must judiciously pick and choose. BankXX was not necessarily designed to operate in sparse situations to find the needle-in-the-haystack.

Thus we have come full circle back to our underlying design assumptions. In the context for which it was intended—an abundance of items to explore—BankXX performs best. Of course, the circle we have traveled was somewhat long in that we needed to find out just how well BankXX performed, and on which type of cases it performed best. Just because a program is designed to perform in a certain manner is no guarantee that it will. However in the case of BankXX we did indeed close the loop.

4. Evaluation from Knowledge Representation and CBR Perspectives: Comparing HYPO with BankXX

In this set of experiments we compare BankXX's retrieval performance with that of the well-known and well-regarded legal case-based reasoning program HYPO [Ashley, 1990]. We have not used the HYPO program itself in these comparisons due in part to technical changes in operating system, source language, and hardware, but we have re-implemented the core case-retrieval and case-selection methods of HYPO in a small program we call *μHYPO* (micro-HYPO). μHYPO consists of HYPO's core methods to determine the applicability of dimensions to a fact situation, to create claim lattices, and to extract most on-point and best cases for a given side from the claim lattice. There are a number of functionalities of the original HYPO program that are not present in our μHYPO re-implementation, all of which contributed to the success of HYPO. These important features include a module to generate three-ply arguments and the capacity to create telling hypotheticals that probe the limits of a fact situation.

In this section we describe two experiments. The first experiment evaluates the performance of μHYPO against the hand-coded court opinions. The second experiment attempts to determine whether BankXX retrieves legal information that is present in the hand-coded opinion, but which μHYPO does not retrieve. As the baseline we always use the standard BankXX/AP configuration with the 10-term evaluation function, given in the Technical Appendix, with *Estus* as start node and limits of 30 closed nodes and 1000 billable seconds.

Before we describe these two experiments, some preparatory discussion is required to describe our efforts to put BankXX and μHYPO on the same footing so that their retrieval performance can be compared legitimately. There are two parts to the preliminary discussion. (1) In Section 4.1 we detail how the definition of a "best case" varies in the μHYPO, HYPO, and BankXX programs. These differences are rather small on their face, and somewhat technical, but they are one aspect that varies between the programs we compare, and therefore should at least be noted. We doubt that they account for more than a small portion of the observed differences in retrieval, however. (2) In the second part of the preliminary discussion, Section 4.2, we describe our policy for putting the output of μHYPO in a form that can be compared with the hand-coded opinion and with BankXX. HYPO generated three-ply arguments, not instantiated argument frames like the hand-coded arguments, and therefore some work must be done to capture μHYPO's output in a frame commensurate with the hand-coded arguments.

Unless two programs are very simple and use the same input and output representations, trying to compare them can be fraught with caveats and compromises. We shall note these compromises as we describe the experimental design. Part of the difference in the programs stems from the distinct emphases of the two projects. The BankXX project emphasizes the retrieval phase of case-based reasoning, whereas HYPO emphasizes the importance of inter-case comparison and post-retrieval application of the cases as well.

On the other hand, putting one program in the framework of another can be enlightening. For example, HYPO and μHYPO are not search programs; BankXX is. μHYPO can be placed within the search paradigm, however. This mental exercise reveals that μHYPO and BankXX have different biases when regarded from the search perspective. By search bias, we mean each program's preference for what nodes (items of legal knowledge) to open and close (to explore and harvest) and for a particular ordering of those nodes.

HYPO considers the intersection of dimensions that are present in both a problem case and retrieved case to sort retrieved cases into a claim lattice, which is a directed, acyclic graph. If HYPO were regarded as a search program, it might be thought of as representing the search space with respect to each problem case—dynamically—as a claim lattice graph. In this view, HYPO uses breadth-first search to locate the best and most-on point cases within the lattice. HYPO also terminates the search at depth one since it makes little use of cases that are more than one edge away from the root, which represents the problem node. Alternatively, all cases in the claim lattice can be considered opened in a search space consisting of the entire case base, and only those at depth one are closed. Either way, HYPO demonstrates a bias towards selecting (closing) most on-point or best cases, that is, cases that have many applicable dimensions shared with a problem case. These observations apply to μHYPO as well.

BankXX, on the other hand, reflects a different and complex retrieval bias that is captured by the interaction between its neighbor methods (which determine what nodes will be considered next), its evaluation functions (which order the nodes) and its argument pieces (which determine how the information in closed nodes is amalgamated). For BankXX/AP, there are additional biases stemming from the fill limits. The search space includes the case-domain-graph but is notably larger than it because the neighbor methods greatly expand these nodes when they are applied.

In the next section we continue our discussion preliminary to the experiments by showing how the characterization of "best" and "most on-point" cases varies between HYPO, μHYPO and BankXX.

4.1 μHYPO and BankXX's HYPO submodule

As we just noted, BankXX contains certain key HYPO functionalities, such as the ability to perform dimensional analysis, create a claim lattice, and extract most-on-point and best cases. BankXX's HYPO submodule is the same as μHYPO in all respects except the definition of *best* case that is used. While μHYPO uses a slight relaxation of the original HYPO definition, BankXX's HYPO submodule applies a definition that represents a more radical departure from HYPO's definition.

The precise definition of best case is important to these BankXX experiments because the *aggregated-pro-cases* and *aggregated-contrary-cases* argument pieces include the *best-same-side-cases* and *best-contrary-cases* argument pieces, respectively. The latter argument pieces depend on the definition of best case, of course. Thus the definition of best case affects the retrieval performance of BankXX.

First, μHYPO and BankXX's HYPO module use the same definition as HYPO for a *most on-point* case: a case whose applicable dimensions shared with the problem situation are properly contained in no other case's set of dimensions shared with the problem [Ashley, 1990]. That is, a most on-point case is a maximal case in the on-point ordering of cases by set-inclusion on the sets of applicable dimensions intersected with the dimensions applicable to a problem. The claim lattice captures this ordering. A most on-point case is thus found in the root node, which contains the current problem case, or in the first level below the root node.

However, the BankXX-HYPO module and μHYPO differ in their definitions of best case. A *best* case in μHYPO is a case that is: (1) a *most on-point case* and (2) decided for the side that is citing the case as its best. This is a slightly different sense of "best" case from that used in the HYPO.[53] It is possible for there to be no best cases according to this definition, since no most on-point case may have been decided for the required side.[54]

To ensure that there will be best cases in BankXX, we relaxed the most on-point case requirement. A best case for a given side in BankXX is (1) a maximal case in the on-point ordering in a claim lattice that consists only of (2) cases decided for the given side.[55] In other words, the best cases in BankXX are those decided for a given side that are closest to the root node, even if they are not most on-point cases.

Thus μHYPO (and HYPO) use an absolute sense of best: a best case has to be a most on-point case. BankXX uses a relative sense of best: no better case can be cited for this side, even if cases for the other side are strictly better. The definitions are summarized in Figure 14.

Definitions in μHYPO and BankXX

	μHYPO	**BankXX**
Most On-Point Case	maximal case in claim lattice	maximal case in claim lattice
Best Case	1. most on-point case 2. decided for side citing it	maximal of those decided for the side citing it

Figure 14. Most on-point case and best case definitions in μHYPO and BankXX-HYPO.

[53]HYPO also required that a best case have at least one dimension whose applicability is tagged as advantageous to the point-of-view taken in the analysis.

[54]This can also happen in HYPO. In fact, the requirement of an advantageous dimension makes it even more likely that no case will qualify as a best case.

[55]This is tantamount to first restricting all the cases (in the claim lattice) to those for the desired side and then selecting mopc's from the restricted lattice. The μHYPO and original HYPO definitions reverse the order of applying the restrictions: select mopc's and select same-side cases. The problem is that taking-mopc's and taking-same-side cases do not necessarily commute. Note, the only way the BankXX-HYPO module's definition yields no best cases is if there are no cases for the desired point of view. By comparison, any time there are no same-side cases within the set of mopc's, there will be no best cases for HYPO or μHYPO. While this observation is something of a technicality, it does make a difference in the retrieval of "best" cases.

μHYPO and BankXX's HYPO-submodule also differ in their computational details from the original HYPO implementation in some aspects. For instance, both μHYPO and BankXX use lattice-building algorithms that are derived from recognizing that a claim lattice is an instance of a Hasse diagram, which is a partial ordering of sets by the subset relation. This observation has enabled us to experiment with a number of algorithms for generating Hasse diagrams to generate claim lattices.

In the next section we deal with the last preliminary step to be addressed before giving the results of our experiments on μHYPO: how to place output of μHYPO in an argument frame so that it might be compared with the hand-coded opinions and with BankXX.

4.2 The Argument Record in the BankXX-μHYPO Comparisons

The BankXX-court experiments required putting BankXX's output and court opinions in a form suitable for comparison by an argument-comparing program. This set of experiments requires putting μHYPO output, BankXX output, and hand-coded opinions into a format amenable to direct comparison.

We use the same four aggregated argument pieces as we did before: *aggregated-theories*, *aggregated-pro-cases*, *aggregated-con-cases*, and *leading-cases*. (See Section 3.1.1 Figs. 1 and 2.) The effect of the best case definitions is felt in the *aggregated-pro* and *aggregated-con-cases*. To use these argument pieces with μHYPO, we had to decide what would count as a leading case or a theory in μHYPO and how to limit the set of ordinary same-side and contrary cases. Our definitions are shown in Figure 15.

aggregated-theories	any theory **promulgated** in **any** μHYPO **most on-point case**
same-side-cases	any case for the current viewpoint in the top three levels of the claim lattice up to a limit (8, the sum of the fill limits on *supporting-cases* and *best-supporting-cases* in BankXX/AP)
contrary-cases	any case for the opposing viewpoint in the top three levels of the claim lattice up to a limit (6, the sum of the fill limits for *contrary-cases* and *best-contrary-cases* in BankXX/AP)
leading-cases	any μHYPO **most on-point** case

Figure 15. Definitions of argument pieces applied to μHYPO output .

These definitions are somewhat arbitrary, but some such commensurization was necessary to make a μHYPO-BankXX comparison study possible. The argument pieces used in these experiments only make use of about half of BankXX's standard argument pieces, so there is some awkwardness in placing BankXX in this framework as well. In the next paragraphs we describe the motivation behind these definitions of argument pieces for μHYPO.

Aggregated-theories: μHYPO does not explicitly represent and reason with legal theories. However, it seems reasonable to award to μHYPO any theory that is promulgated by a most on-point case identified by μHYPO. It might be noted that the same courtesy is not extended to BankXX, which must explicitly close a legal-theory node and add it to the *applicable-theories* or *nearly-applicable-theories* argument pieces in order to count it as "found." For BankXX, finding the promulgating case (e.g., *Estus*) does not give the program credit for the corresponding theory (*Estus theory*).

Same-side-cases: Perhaps the obvious choice for cases in a μHYPO same-side-cases argument piece is to use only the μHYPO best cases for the current viewpoint. In practice, however, there were very few best cases (according to μHYPO's fairly strict definition of best) and the recall of μHYPO was unduly low when measured against hand-coded court opinions in preliminary experiments. The solution we chose was to permit μHYPO to incorporate in its argument a set of cases from the claim lattice that includes the best cases. Even though the original HYPO did not make extended use of cases that were not most on-point cases, we decided to include cases that were deeper in the claim lattice into the same-side-cases argument piece definition.

On the other hand, it was not reasonable to include all the cases in the claim lattice for the current side, since almost all cases appear in some level of the claim lattice generally, and including all same-side cases would have severely lowered μHYPO's retrieval precision.

The question then arose as to how many additional cases to include with the best-cases. Since in a later set of experiments we compare BankXX/AP against μHYPO, it would be fair to give the two programs the same number of potential cases that could be selected for this argument piece. BankXX/AP, which has "fill limits" on the argument pieces, is permitted a maximum of eight cases in the same-side-cases piece. Thus we allowed μHYPO to select up to eight same-side cases from the top three levels of the claim lattice, where the root node is at level 1. (Since the claim lattice is a graph, and not a tree, the term "level" does not really apply, and by the top three levels we mean all cases that are in the root node or that are within two edges of the lattice root node.)

Contrary-cases: The same considerations that applied to μHYPO's definition of same-side-cases apply to contrary-cases. The fill limits for BankXX/AP's contrary cases argument piece is six, however, and that limit is incorporated into μHYPO's allotment.

Leading-cases: μHYPO does not provide a sense of leading case in the BankXX sense of a case that is frequently cited. However it did seem reasonable in the BankXX-HYPO experiments, to award to μHYPO as a leading case any most on-point case, since these are the important (leading) cases that drive much of μHYPO's analysis.

Using these generous conventions for awarding theories, cases and leading cases to μHYPO allows us to explore value-added questions, such as *How do BankXX's methods compare with some obvious extensions to HYPO, like those for finding theories and leading cases by reference to most on-point cases?*

Although the fit between the simplified form of argument used to encode output by both BankXX and μHYPO is tight in some places and too loose in others, some common form of commensurable output was required for automatic comparison. Other choices for an inter-program comparison might have been to create a third argument *interlingua* for the two programs or to extend BankXX's output to conform to HYPO's style of active point-counterpoint-rebuttal arguments, inventive hypotheticals, etc. We felt that such an attempt to make a three-ply BankXX argument would have taken this project too far afield from its focus of heuristic retrieval.

Finally, we acknowledge that the BankXX-μHYPO comparison study obviously required some compromises and that the comparison really is not between HYPO and BankXX but rather between scaled-back versions of each. Thus, we must give an explicit caveat that the empirical results cannot be accorded the status that a comparison between totally commensurate systems would.

4.3 Comparing μHYPO Arguments with Hand-Coded Arguments

Before comparing the relative performance of BankXX and μHYPO against the hand-coded opinions, we performed a preliminary experiment to compare μHYPO's output with the hand-coded arguments of the court opinions, just as we did with BankXX in the BankXX-court comparisons. As the baseline we used the standard BankXX/AP configuration.

μHYPO was run on each case in the BankXX case base and the outputs transformed into an argument record for μHYPO output as just described. For each of the four argument pieces we then computed the average precision and recall over all the cases (Figure 16). In giving the precision and recall values we used the policy that where 0/0 arose in the precision or recall computations, the result should be regarded as 1.

For the three aggregated argument pieces concerning cases, BankXX/AP demonstrates higher precision and higher recall than μHYPO. On the other hand, BankXX/AP was designed to find leading cases and so that it outperforms μHYPO on this argument piece is not surprising and, of course, μHYPO incorporates a different implicit sense of "leading case."

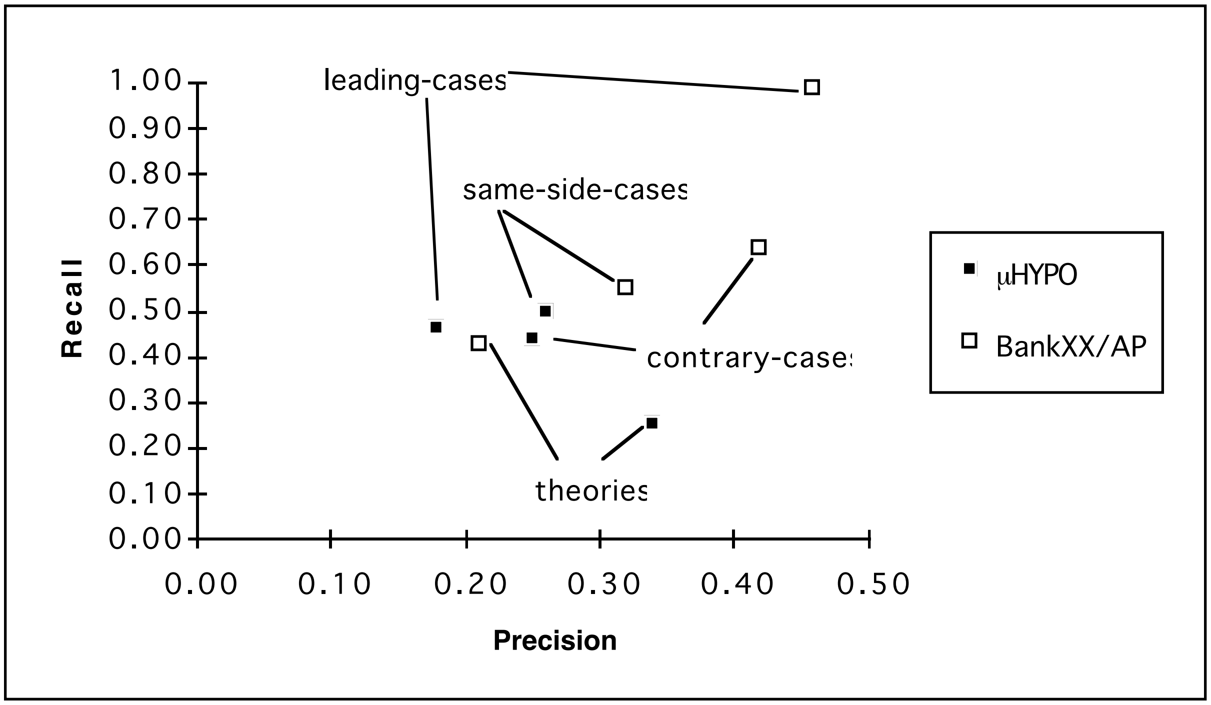

Figure 16. Averaged precision and recall for µHYPO and BankXX/AP with respect to hand-coded argument answer keys. The averages include the value of 1 for undefined precision or recall quotients.

The one surprise in these experiments is that µHYPO achieves a higher level of precision than BankXX/AP on legal theories. BankXX/AP is designed to be inclusive with respect to legal theories, and therefore its theory recall is substantially higher than µHYPO's. Our policy of which theories to award to a µHYPO argument seems to be very precise but misses many that it shouldn't. This high precision tends to show that relevant theories are often promulgated in most on-point cases. This result reinforces our intuition that most on-point cases are indeed highly central to argument. On the other hand, µHYPO's lower recall tends to show that there are other sources of relevant theories and that only looking to most on-point cases for theories is not an adequate retrieval strategy in itself.

Summary for Section 4.3
Comparisons against hand-coded arguments:

With respect to the hand-coded arguments, BankXX/AP outperforms µHYPO on all four aggregated argument pieces for both precision and recall with one very notable and surprising exception (precision on theories).

4.4 Comparing Arguments—BankXX, μHYPO, and Hand-Coded Opinions

Perhaps the most telling way to compare the BankXX and μHYPO programs is to determine for each case what items in the case-domain graph satisfy the following three requirements:

(1) they are cited in the actual case and so are part of the hand-coded argument,

(2) they are retrieved by BankXX,

(3) they are not retrieved by μHYPO.

The number of items that meet these three requirements give one indication of the "value-added" by BankXX over a HYPO-style analysis.

We ran an experiment to determine the average number of instances providing such additional value. Our experimental procedure was first to input each of the 55 cases in the case base as a problem case to BankXX and to μHYPO in the usual *de novo* manner. An argument constituted by the four aggregated argument pieces was created by each program. Then for each such argument piece, the items that appeared in the BankXX and hand-coded arguments but not in the μHYPO argument were computed. As in the previous experiment, to create the BankXX argument we used BankXX/AP in its standard configuration.

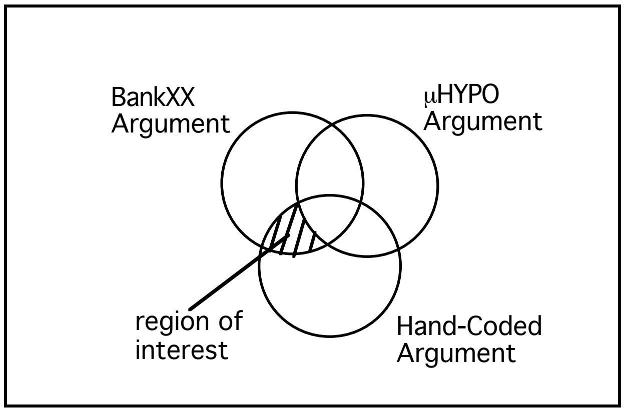

Figure 17. Region of particular interest in this experiment: instances found by BankXX, not by μHYPO , but in the hand-coded arguments.

Table 5 summarizes the experiments. The rows are defined as follows:

(1) Court and BankXX/AP, not μHYPO: Average number of objects in the court opinions, found by BankXX, but not by μHYPO.

(2) BankXX/AP and Court: Average number of objects in the court opinions found by BankXX.

(3) Court and μHYPO, not BankXX/AP: Average number of objects in the court opinions found by μHYPO that were not found by BankXX/AP.

(4) Court: Average number of objects in the court opinions.

	Same side cases	Contrary cases	Leading cases	Theories

Court and BankXX/AP, not µHYPO	0.53	0.58	1.10	0.65
BankXX/AP and Court	0.93	1.04	1.80	0.95
Court and µHYPO, not BankXX/AP	0.54	0.25	0	0.31
Court	2.24	2.15	1.82	2.51

Table 5. Average number of objects found in each of the four aggregated argument pieces satisfying certain conditions, where average is taken over 55 cases. Explanations of the row are given in the text.

Several observations can be made:

(a) As shown in row 1, BankXX finds a substantial number of objects from the court opinions not found by µHYPO. In fact, row 3 shows that no leading cases (according to BankXX's definition of "leading") were found by µHYPO that were not found by BankXX .

(b) By comparing rows 1 and 2, it can be seen that a majority of the items found by BankXX that also appeared in the hand-coded opinion were not found by µHYPO. BankXX's retrieval biases leads to items of interest from the court opinions that were typically not found by µHYPO. Thus, there is solid sense of added value by BankXX when measured against µHYPO-style retrieval.

Given that BankXX uses HYPO-style dimensional analysis and claim lattice creation, BankXX ought to perform at least as well as µHYPO, although it is conceivable that it might not. But is not surprising that the rest of BankXX's machinery yields additional retrieved items. BankXX has additional domain knowledge in the form of neighbor methods and evaluation functions that are not present in µHYPO, as well as more indexing information than µHYPO, particularly in the form of inter-case citation links. By incorporating legal domain knowledge not present in µHYPO on top of the HYPO case analysis framework, BankXX retrieves items appearing in the judicial opinions that were not found by µHYPO.

(c) Row 3 shows µHYPO also finds some items not found by BankXX, although usually fewer than the number found by BankXX but not by µHYPO. However, µHYPO actually finds as many same-side-cases from the court opinions that were not found by BankXX as BankXX finds same-side cases not found by µHYPO. The limit on the µHYPO same-side argument piece (8 cases) was not frugal, but the general lesson is that µHYPO and BankXX demonstrate complementary search biases. One direction for future work would be how to combine the cases retrieved by these two systems to get the benefit of this better combined coverage of the search space without unduly diminishing precision.

Summary for Section 4.4
Value-added comparisons against hand-coded arguments

BankXX generally found information cited in judicial opinions that was not retrieved by μHYPO. Thus, one can say that there is value added by BankXX's approach. BankXX provides a viable complementary search technique for retrieval of cases and legal theories to that used by μHYPO.

5. Evaluation from a Search Perspective: Comparing BankXX's Performance under Different Parameter Settings

There are a large number of input parameters to BankXX, including the start node, the evaluation function to use, the limit on the number of nodes that can be closed, and a time limit. The question of how well BankXX works can be answered in potentially many ways, depending on the input parameters to the program. We have performed a series of experiments to control individual parameters, or in some cases pairs of parameters, to give a sense of the efficiency and accuracy with which BankXX performs its tasks. By running a configuration of BankXX with a given input parameter setting on all the cases, and determining the average number of instances filling the argument pieces, we can isolate the effect of various parameter settings on the performance. In these experiments we only use 11 argument pieces; we disabled the *family-resemblance-prototype*.

This section is quite specific to the particular implementation of BankXX. Readers interested in the experimental results in a general way may want to read quickly the boxed experimental set-up, scan the graphs and note the result summaries. Experimental conclusions should not be taken out of their very specific context, however. We have not investigated the effect of many aspects of this program. Chief among such features are the legal domain, the size of the case base, the representation languages and encodings for the cases and arguments, the particular cases we have selected for the case base, and the links we have specified manually, such as the links between legal theories. Nevertheless, we expect that many of our observations will be borne out by future symbolic AI programs that use a combination of search and indexing to access legal knowledge.

As in all these experiments, each case and its links were excised from the case graph when it was regarded as a *de novo* problem situation for the program. For this set of experiments, we have relied primarily on one measure of evaluation: counting objects found for each argument piece.

In the following experiments the parameter settings to BankXX that are not specified in the *Parameters Varied* boxes are default values, which are:

Start Node: *Estus,*
Number of nodes closed: 30,
Number of Billable Seconds: 1000.

5.1 Start Node

One potential objection to the kind of automatic, knowledge-directed browsing that BankXX performs is that in a large knowledge base, one would not know where to begin. Further, the objection proceeds, search might wander around in a small and not-very-useful portion of the case base, without locating the information needed to support an argument. The following experiments show that for our case base and program implementation, this objection was not supported.

Parameters Varied:

Start Node (*Estus*, random, most on-point case)
Evaluation Function (node-type, argument-piece, argument-factor)

We varied the start node, which is an optional input parameter to BankXX. Three different start node specifications were considered:

(1) the *Estus* case,
(2) a random case, and
(3) a most on-point case.

Since *Estus* is a well-known case in this area of bankruptcy law, we felt almost any research materials consulted by an attorney would soon lead to the *Estus* case, and therefore *Estus* may be considered a realistic and useful starting point. The second start node specification was a random case selected according to a uniform distribution of cases in the case base. Starting at a most on-point case requires creating a claim lattice before search commences, extracting the most on-point cases, and then randomly selecting a most on-point case.[56]

In this experiment, for each of the three evaluation functions the number of instances filling each of the argument pieces was counted after 30 nodes were closed. All cases in the case base were used as *de novo* problems and the average number of objects filling each argument piece was computed.

The results in Figures 18, 19 and 20 show that the choice of start node, which can be considered an initial query to the case base, made little difference to retrieval. For each evaluation function, the average number of objects found for each argument piece is about the same for each of the three start nodes. The only exception to this occurs for the *leading-cited-cases* argument piece with BankXX/NT: the number of leading cases found when starting with *Estus* is almost twice the number found when starting with the current problem or a random case. We believe this finding reflects the common wisdom that starting with a good case makes it easier for a researcher to exhaust the major cases. While the random and most on-point case start nodes did often lead to *Estus*, they did not permit the examination of

[56]In a design decision that we would change, the other most on-point cases are thrown away rather than placed on the open list. Including all the most on-point cases on the open list would have made better use of the resources expended to create the claim lattice and probably provided more fertile cases to initialize the open list.

as many of the leading cases as did beginning with *Estus*, at least within the 30-closed-node limit placed on the program.

We also examined search paths through the case-domain graph to understand why the choice of start node made so little difference. As a broad generalization, no matter where search starts in this case-domain graph of approximately 150 nodes, it soon leads to a highly interconnected region which contains many useful cases and theories (see Figure 21). For example *Estus* and *Rimgale*, 669 F.2d 4226 (7th Cir. 1982), 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 eventually to *Estus*, so to speak. Thus, in browsing a case base of comparable size where there is a sufficiently rich indexing fabric, the initial starting point in case memory may not matter.

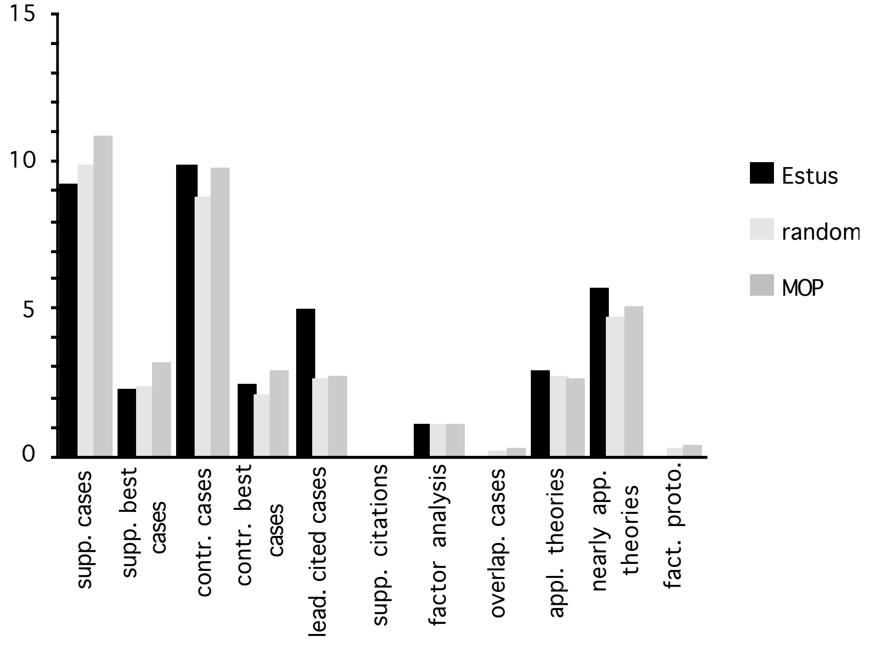

Figure 18. Histograms of number of objects filling each argument piece with BankXX/NT.

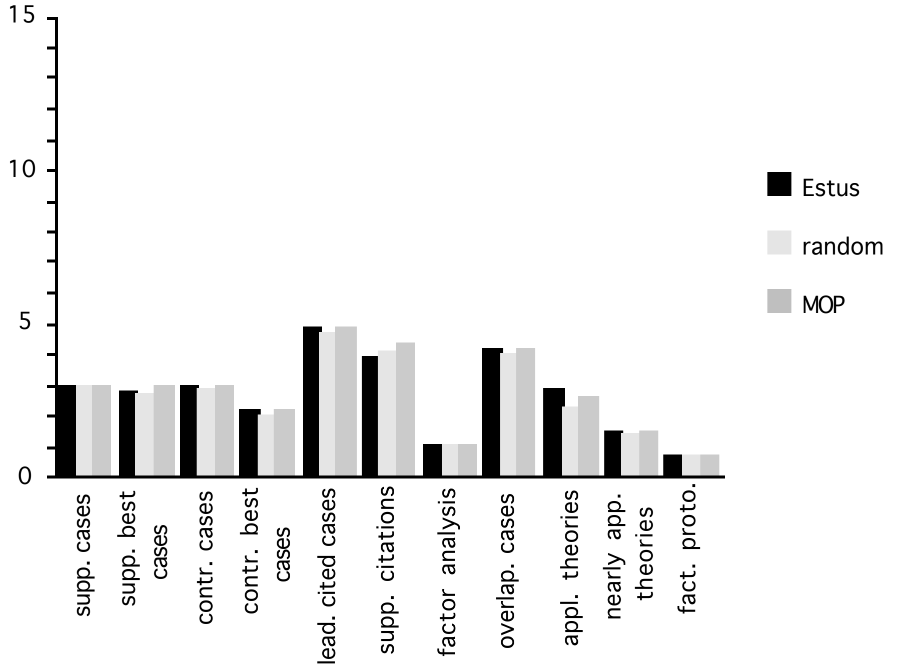

Figure 19. Histogram of number of objects filling each argument piece with BankXX/AP.

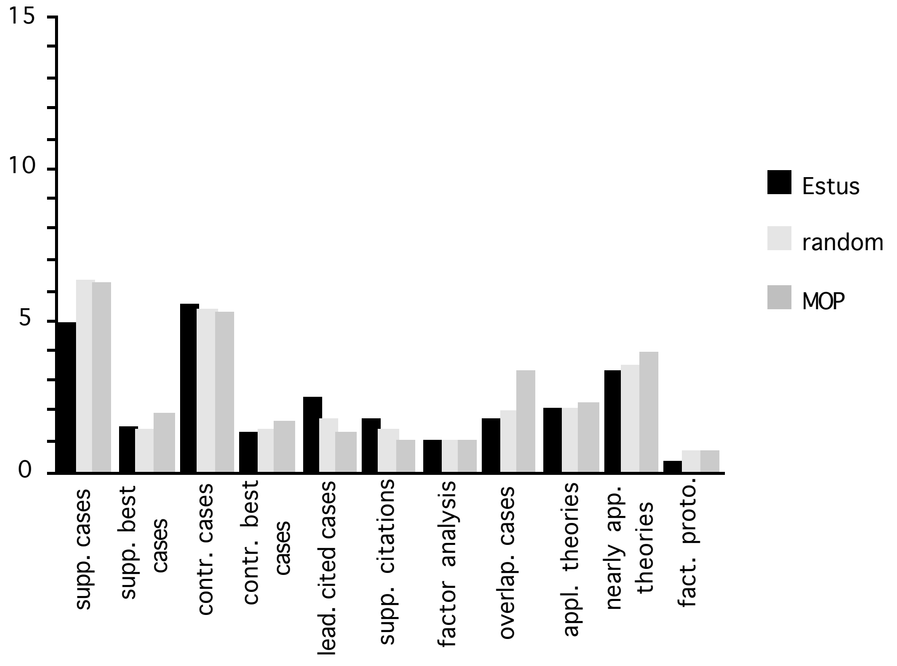

Figure 20. Histogram of number of objects filling each argument piece with BankXX/AF.

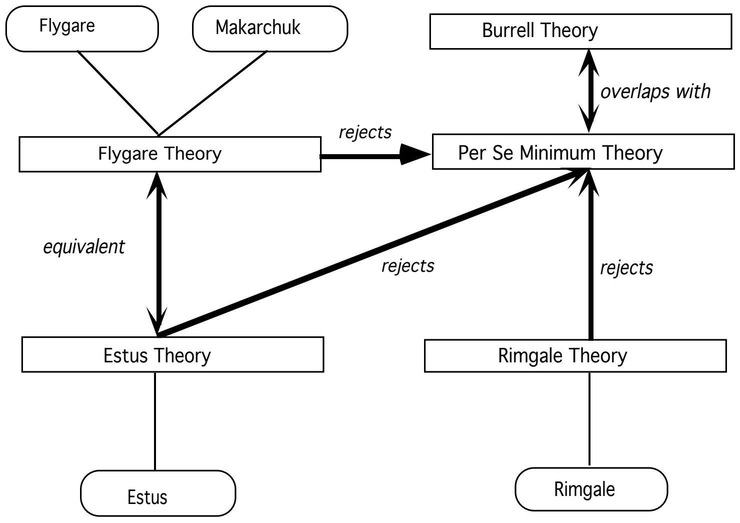

Figure 21. A small subgraph of the case-domain graph.

Summary for Section 5.1
Comparisons with respect to start node

In general, BankXX's performance did not depend on the choice of the start node.

5.2 Evaluation Function and Number of Closed Nodes

BankXX can be configured with any of three evaluation functions to assess the worth of a piece of legal knowledge to an argument. The next set of experiments compares the performance of BankXX/NT, BankXX/AP and BankXX/AF with respect to resource limits from an internal perspective. (Section 3 performed an external comparison with resource limits held constant.)

As we have noted, in real-world legal research there is no clear guide when to terminate the discovery and analysis of legal materials. No objective standard exists for when one has completed research or completed an argument. BankXX has two termination parameters that may be set by the user: (1) the time limit ("billable seconds") spent by the program, and (2) the number of nodes that can be closed. If the open list is emptied within the user's time or space limits, the program halts.

Since the performance of each configuration of BankXX depends on the resources that have been allocated to it, this experiment was explicitly designed as a 3 x 3 experiment in which we varied both the evaluation function and one measure of resource allocation, the number of nodes that can be closed.

Parameters Varied:

Number of Nodes Closed (10, 30, 60)
Evaluation Function (node-type, argument-piece, argument-factor)

As in the other experiments, the experimental procedure was to run each of the cases in the case base in the *de novo* manner. We then averaged the number of nodes filling each argument piece for each problem case. In this 3 x 3 experiment design, we compared the effects of the three different evaluation function at each of three node closing levels. The technical appendix contains the terms and weights for each of the three functions. Figures 22, 23 and 24 show the results.

Ten nodes closed is very few, but it does permit BankXX to make some initial progress. While 10 nodes is too few for any version of BankXX to meaningfully search the case-domain graph, even at this early stage, several general patterns of behavior are apparent. With BankXX/NT, 10 nodes was too few to find any items filling the *domain-factor-analysis-significant-overlap*, *supporting-citations*, or *factual-prototype-story* argument pieces. By comparison, although BankXX/AP barely got started on these argument pieces, it did find some items for them. More notably, the number of *leading-cited-cases* harvested by BankXX/AP is on average almost 5 (N.B., there are only 5 leading cases), whereas with the other two evaluation functions BankXX manages to find approximately only one. A similar difference in the number of *supporting-best-cases* and *contrary-best-cases* is also apparent. On many argument pieces, BankXX/AF performed comparably to the BankXX/NT.

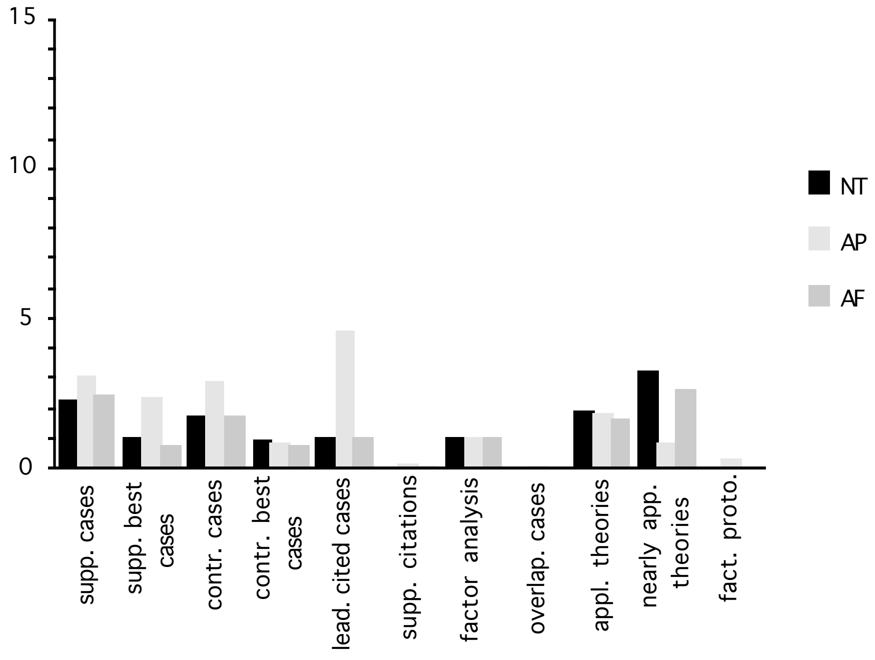

Figure 22. Average number of objects filling each argument piece for each configuration of BankXX with a limit of 10 nodes closed.

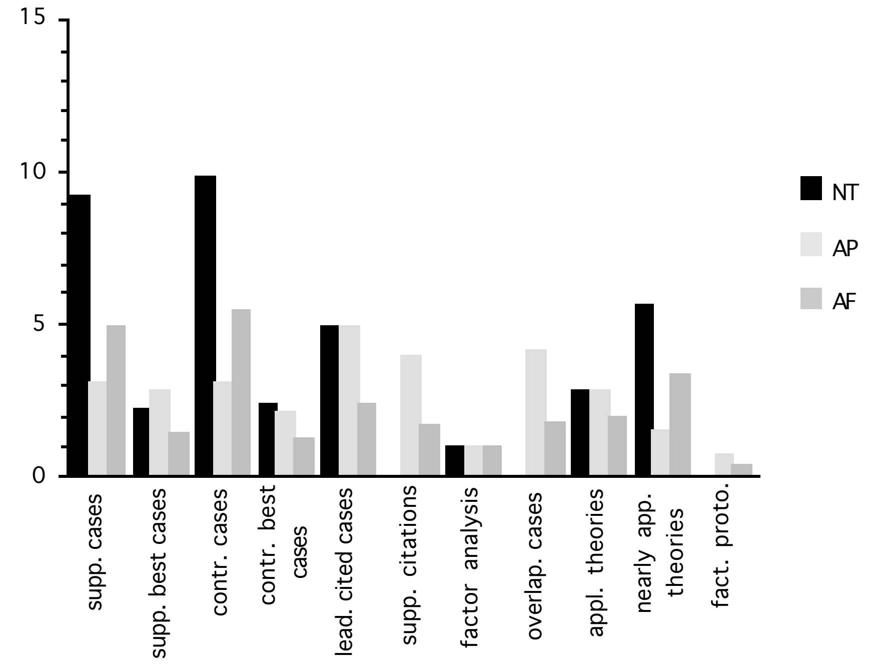

Figure 23. Average number of objects filling each argument piece for each configuration of BankXX with a limit of 30 nodes closed.

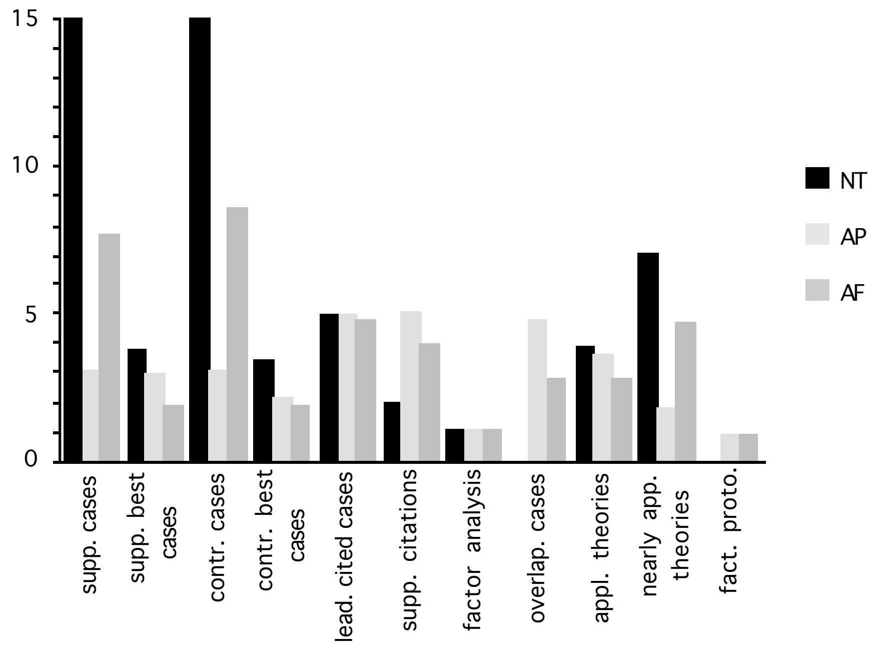

Figure 24. Average number of objects filling each argument piece for each configuration of BankXX with a limit of 60 nodes closed.

At the level of 30 nodes closed, BankXX/NT, which does not have any limits on the number of items that can fill any argument piece, finds many more *contrary-cases* and *same-side-cases* than either BankXX/AP or BankXX/AF but it does so at the expense of finding information for other argument pieces. At 30 nodes, BankXX/NT still has filled in no objects in the *domain-factor-analysis-significant-overlap*, *supporting-citations*, and factual-*prototype-story* argument pieces. BankXX/AP yields more cases whose factor analyses fill the *domain-factor-analysis-significant-overlap* and *supporting-citations* argument pieces than the other two configurations. In general BankXX/AP is much more balanced in harvesting items for the argument pieces than BankXX/NT, which runs away on ordinary pro and con cases. This is a direct result of the fill limits enforced by the argument piece evaluation function on the numbers of items than can be harvested for each piece. Although BankXX/AF does not harvest the same number of items as BankXX/AP it is similar in that it produces a fairly balanced profile.

At the level of 60 nodes closed, the observations made at the level of 30 nodes continue to hold. BankXX/NT concentrates on finding cases at the expense of the other argument pieces and consequently locates many more contrary and supporting cases and is even less balanced in its coverage of the argument pieces. Visiting 60 nodes resulted in many cases being found by BankXX/NT that were not particularly useful, but were merely added to the *contrary-cases* and *same-side-cases* argument pieces. BankXX/NT only found *supporting-citations*, however, in the 60-node run.

Since the total limit on objects filling the argument pieces is 39, BankXX/AP shows no significant progress when the node level limit is raised to 60 nodes. BankXX/AF also merely adds more supporting and contrary cases to the corresponding argument pieces. For each argument piece (except for *current domain factor analysis*, which always contains only the factor analysis of the current problem), BankXX/AF showed an increase in the number of items filling it, in a stair-step pattern, as the number of nodes closed is increased from 10 to 30 to 60.

One lesson of comparing the results of the evaluation functions is the utility of limiting the number of items that can fill any one piece, as is done with the argument piece evaluation function. Such a limitation leads to a more balanced argument, as might be anticipated.

The node-type function uses only the object type of each node and does not limit the number of objects retrieved for any argument piece. Considering its lack of knowledge, BankXX/NT 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. It appears that in a two-phase retrieval, applying 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].

For BankXX/AP with *Estus* as the start node, 10 nodes was too few to fill up many of the argument pieces, and 60 was more than the fill limits allowed. Thus, as a rough guide, 30 nodes seemed an appropriate compromise between more exhaustive search and too shallow an examination of the case-domain graph. Incremental benefits of more search decreased after about 30 nodes. From a case-based reasoning 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] regarding the optimal amount of time to devote to searching a case base in a hybrid planner.

Summary for Section 5.2
Comparisons with respect to number of nodes closed

Limits on the number of objects that can be harvested for the argument pieces, like the fill limits used in BankXX/AP, result in more balanced arguments than those produced without them, as is the case with BankXX/NT and BankXX/AF..

With respect to the number of nodes closed, this set of experiments revealed that there was a point after which additional node closings did not provide additional benefit.

5.3 Number of Billable Seconds

The usefulness of a heuristic evaluation function depends on its efficiency as well as its accuracy. A perfect or near-perfect evaluation algorithm that runs in exponential time in the number of features considered would be little use. That the time to evaluate a node be short is important in information retrieval settings where users expect fast retrieval in real time. In some production-level systems, for example, design constraints require that responses to queries be made in fewer than five seconds. While in an automatic browsing system such as BankXX time constraints are not as severe (particularly since BankXX is a research prototype and not a production-level system), efficiency is still a concern.

In designing the three evaluation functions, we attempted to implement functions that incorporated different levels of abstraction of knowledge about argument. The node-type evaluation function only considered the lowest level of abstraction, the type of knowledge that was under evaluation (case, theory, etc.). The argument piece evaluation function captured a more abstract level, certain kinds of information known to be useful for argument. The argument factor evaluation function incorporated the highest level of abstraction, that of considering the quality of the emerging argument.

But the time to compute each evaluation function also increases with the level of abstraction. The node-type evaluation is quick and easy to apply, since it just checks the type of node that is being evaluated. The argument-piece evaluation requires a more complex computation, evaluating the predicate for each argument piece. Finally, the argument factor evaluation function requires the most complex and costly computation, since it involves evaluating the incremental argument along a number of argument dimensions.

In order to test the efficiency of each evaluation method, we placed three different limits on the amount of time the program was permitted to run, and compared the effects of a time limitation on the three configurations of BankXX. The time limitations also include pre-processing performed by BankXX each time a new problem is presented, which is independent of the evaluation function. At the commencement of processing, the program performs dimensional analysis and creates a claim lattice for the problem situation, for example.

Parameters Varied:

Number of Billable Seconds (1 2 10)
Evaluation Function (node-type, argument-piece, argument-factor)

As in the other experiments, each case was treated as a new problem situation and analyzed by BankXX. Configured with each one of the three evaluation functions, the program was run for 1, 2, and 10 billable seconds on a DEC Alpha running Lispworks in 64MB of physical memory. The average number of items filling each argument piece was computed across the case base. The results are shown in Figures 25, 26 and 27.

BankXX/AF runs the slowest since it involves provisionally extending the existing argument by each node that is on the open list, and computing each of the argument factors on that provisional argument. Creating a temporary, provisional argument for each node evaluation is computationally expensive. When a time limit is placed on BankXX, therefore, BankXX with the argument factor evaluation tends to underperform the other two configurations.

This effect comports with our own experiences in doing legal research. If every time a researcher comes across a new case or theory, he stops to assess exactly where it will fit into the emerging argument, and then evaluates that argument, research will be a time consuming process. We speculate that researchers are more apt to use heuristics, concerning when a case should be read or carefully studied, that make good use of the often short amount of time available to produce an argument.

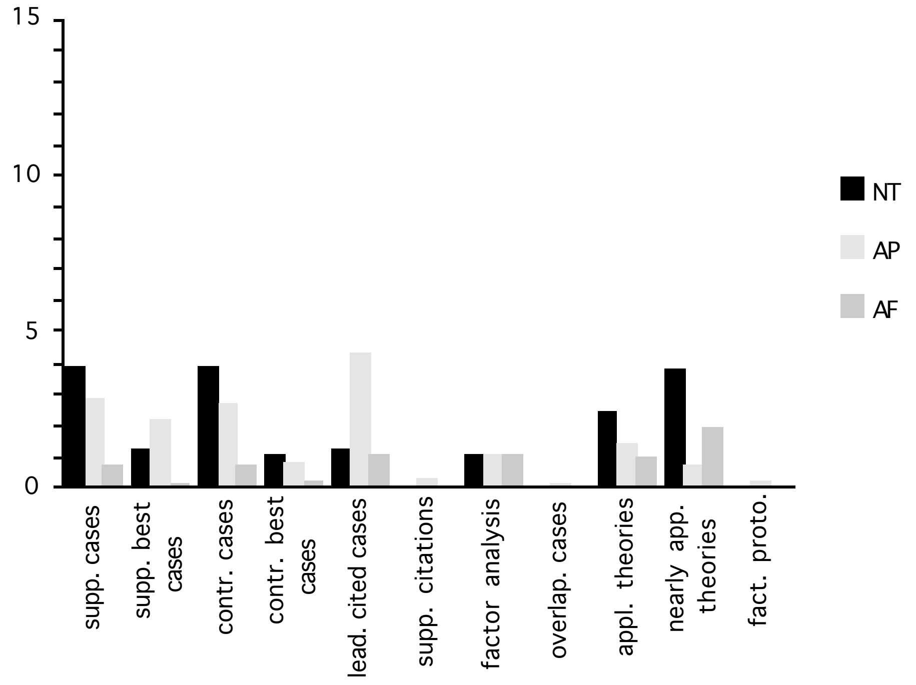

Figure 25. Average number of objects filling each argument piece for each configuration of BankXX with a limit of 1 billable second.

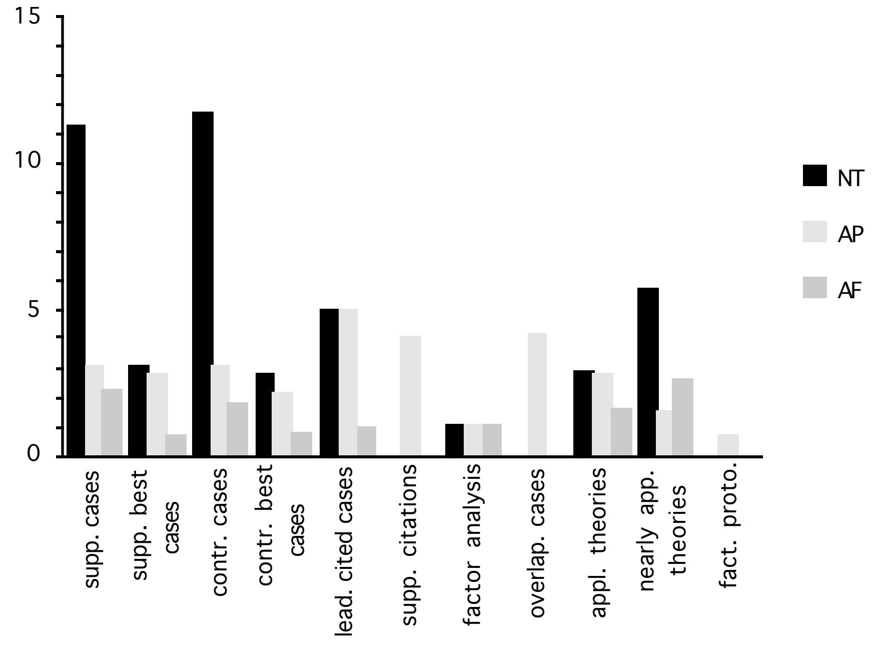

Figure 26. Average number of objects filling each argument piece for each configuration of BankXX with a limit of 2 billable seconds.

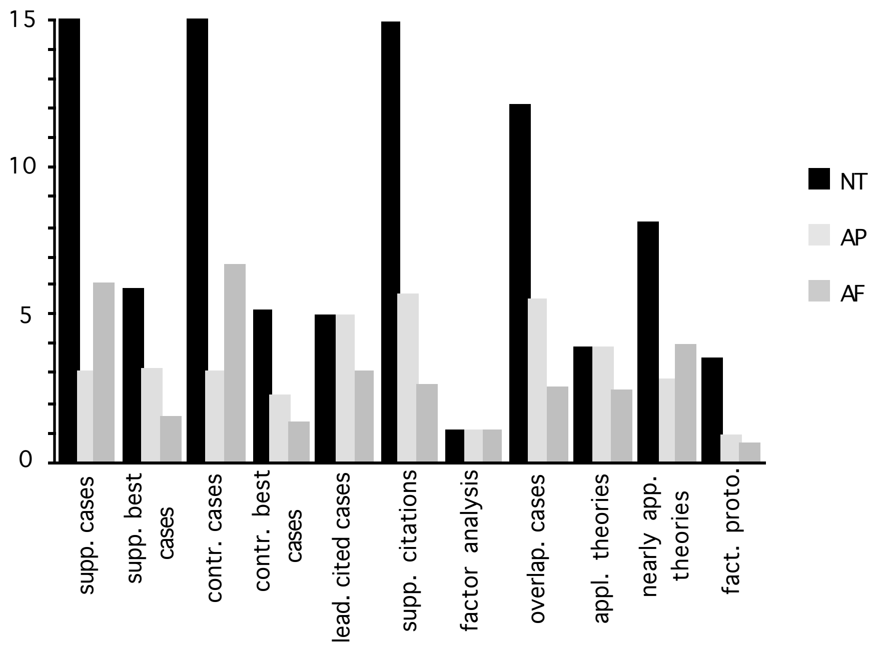

Figure 27. Average number of objects filling each argument piece for each configuration of BankXX with a limit of 10 billable seconds. The actual values for BankXX/NT for the supporting cases, contrary cases and supporting citations argument pieces are all greater than 15.

Note that the overall shape of the 2-billable-seconds histogram in Figure 26 is comparable to that of the 30 node limit histogram (Figure 23). From these experiments, we concluded that a limit of 1000 billable seconds is more than enough for each evaluation function to run until the open list is empty. Thus in the standard configuration of BankXX, we set the time limit at 1000 billable seconds.

Summary for Section 5.3
Comparisons with respect to billable seconds

BankXX with the current argument factor evaluation function was not efficient, and required a large amount of time to determine whether a node was potentially useful to an argument piece.

6. Three Additional Experiments

In this section we describe three additional experiments concerning: (1) portions of the search space frequently explored by BankXX; (2) discovering leading cases by analyzing citation frequencies; and (3) an alternative approach to precision and recall that takes case similarity into account.

In the course of a set of evaluation experiments of a complex program, tangential questions arise that do not speak directly to how well a program works, but rather to *how* a program works. Partly to satisfy our own curiosity about the program's behavior, we performed a handful of side experiments.

The first two experiments reported in this section deal with different instantiations of the same basic question: Is there a way to determine *importance* in this domain from mere frequency counts? In the first experiment, we ask whether some portions of the search space can be distinguished by having been searched more often than others. In the second experiment, we ask whether citation analysis in a larger subcollection of full text bankruptcy cases can lead us to the leading cases in an area.

6.1 Portions of the Search Space that were Closed

The first question we asked was: *Are some nodes in the case-domain graph opened and closed more often than others?* The answer to this question generally may help to represent the structure of a domain to facilitate retrieval and how to conduct the search. From a jurisprudential perspective, recognizing that some nodes are visited more often than others and the patterns of visitations give clues about the conceptual structure of an area of the law. From a technical perspective, a legal retrieval program that automatically browses a large database may not be able to fit the entire set of domain knowledge nodes in main memory at once. This situation in common in any production level information retrieval program that deals potentially with gigabytes of data. Given this constraint, if a developer knows that some portions of a search space are visited more frequently than others, then it might be useful to cache frequently accessed nodes in main memory to avoid thrashing through excessive paging of instances from disk.

To answer our question, for each of the three evaluation functions we compiled a histogram that records the number of times each node was opened and closed as we ran BankXX through all 55 cases in the case base as *de novo* problem cases. These runs were conducted using a randomly chosen most on-point case as start node, 30 nodes closed per search, a time limit of 1000 billable seconds, and no filtering for dates.

Figure 28 shows the cases in the case base most frequently opened and closed by BankXX/NT. The dark gray portion of each bar represents the number of times a node was closed; light gray gives the number of times a node was opened. There is a sizable core of cases (27) that are opened for all of the cases in the case base. However, nodes are closed with a steadily declining frequency with *Estus* and *Rimgale* closed the greatest number of times.

Note that with a fixed start node (which is not the case here since we randomly picked a most on-point case for the problem case being run), BankXX/NT would open and close the same cases in the same order regardless of problem case, when no date filtering is used. This is because BankXX/NT's evaluation function is not sensitive to the problem case but only depends on node types, and if one always starts from the same node, the nodes opened/closed do not change. So after BankXX/NT gets to *Estus*—which it tends to do early since all routes lead to *Estus*—the rest of the search is pretty much the same.

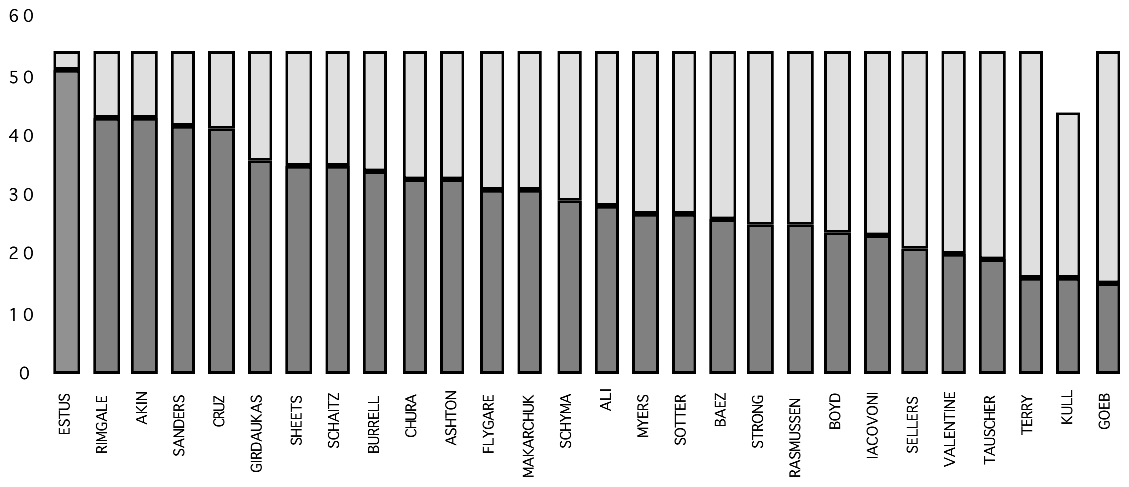

Figure 28. Cumulative number of node openings and closings with BankXX/NT. We show only the (28) most frequently closed cases in the case base. The dark gray portion of each bar represents the number of times a node was closed; light gray gives the number of times a node was opened.

A similar pattern (Figure 29) is shown by BankXX/AP, except that there are about five cases that are very dominant. These are in fact the 5 leading cases: *Deans*, *Iacovoni*, *Estus*, *Goeb* and *Rimgale*. This shows that leading cases are indeed often considered as potentially useful and incorporated into the emerging argument. Fewer cases (20) are opened on every run by BankXX/AP than with BankXX/NT.

In contrast to BankXX/NT, BankXX/AP and BankXX/AF show great variation in the "routes" taken through the case-domain graph. Their routes depend on the problem case.

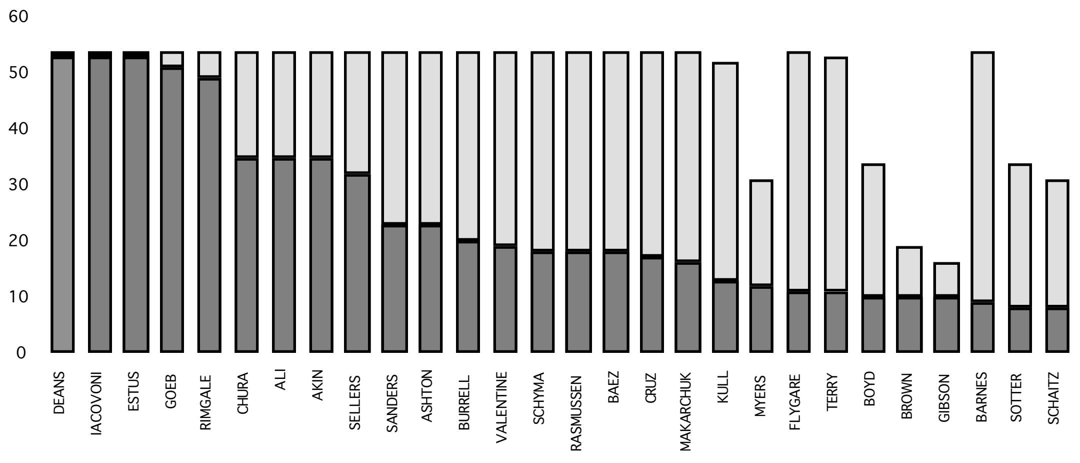

Figure 29. Cumulative number of node openings and closings with BankXX/AP for the most frequently opened cases in the case base.

The results for BankXX/AF were qualitatively different (Figure 30). Whereas BankXX/NT and BankXX/AP showed many cases that were opened by every case, only two cases—*Estus* and *Rimgale*—were opened for every problem case by BankXX/AF. A second distinguishing aspect is that there are very few cases that are closed almost every time in which they were opened.

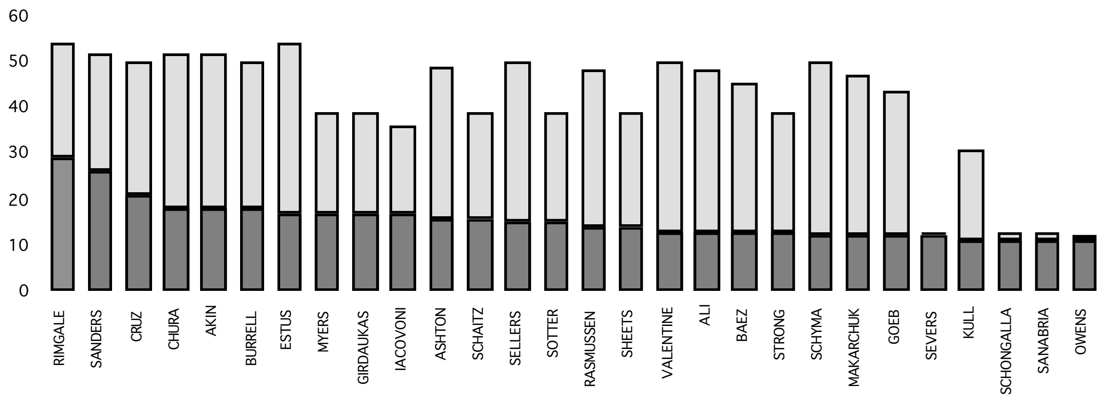

Figure 30. Cumulative number of node openings and closings with BankXX/AF on the most frequently opened cases in the case base.

Summary for Section 6.1
Heavily visited portions of the case-domain graph

BankXX/NT tended to open the same half the case base on every run and BankXX/AP, less so. Both displayed a graded structure for the number of node closings. BankXX/AF showed greatest variation in the number of times cases were opened. With BankXX/AP, the nodes that were closed the greatest number of times were the leading cases.

6.2 Citation Frequencies

One of the measures of importance or centrality of a legal case is the number of citations to it. In order to supplement a researcher's intuitive impression of the what the leading cases are in a legal area it would be helpful to have an empirical and somewhat more objective basis for that determination. It would also be helpful to know which cases are cited in “winning” arguments, although such cases may be a reflection of the facts of the case at bar and not citation strength of the cited cases.

As a side experiment, with the aid of Westlaw, we compiled a citation table, a portion of which appears in Table 6. The table gives, for each case in the case base, the number of cases that cited that case,[57] the year of the original case’s decision, the number of years since the decision, and the number of cites per year to that case.

To construct this table and to establish a baseline figure, a query was put to WestLaw’s Bankruptcy database, “1325(a)(3) and good faith.” The number of cases retrieved was 611. For each of those case opinions, we determined the number of cites to each one of the 55 cases in BankXX’s case base. The number of cites to each BankXX case was determined by additional Westlaw queries, one for each case in BankXX. For example, for the Estus case, *In re Estus*, 695 F.2d 311 (1982) the query “1325(a)(3) and good faith and Estus and 695 F.2d” was put to Westlaw, returning 119 cases. Details on how the experiment was performed, related experiments, and additional analysis, can be found in [Friedman, 1994].

Name	no of cites	year	no of years	cites/year
Rimgale	120	82	12	10.00
Estus	119	82	12	9.92
Goeb	99	82	12	8.25
Deans	85	82	12	7.08
Kitchens	72	83	11	6.55
Memphis	68	82	12	5.67
Flygare	61	83	11	5.55
Iacovoni	73	80	14	5.21
Barnes	56	82	12	4.67
Okoreeh-Baah	27	88	6	4.50

Table 6. Citation frequencies and related statistics for the 10 cases with the greatest citation frequency. U.S. Courts of Appeal cases are given in boldface.

The number of cites per year gives one measure of the citation frequency of a case. This statistic has the advantage over the absolute number of citations because it is normalized for the amount of time elapsed since the decision was handed down. A case such as *In re Okoreeh-Baah*, 836 F.2d 1030 (6th Cir. 1988) has been frequently cited since its decision,

[57]We did not take into account the number of cites <u>within</u> a citing case. (e.g., whether a case cited a case 1 time or 29 times did not matter).

with 27 citations since 1988. Many of the best-known cases in this area, not surprisingly, appear at the top of this list, providing objective confirmation of the importance of these cases. Nine out of ten cases with the highest citation frequency were from the U.S. Courts of Appeal.

The citation frequency case ranking may differ from those used by BankXX because of the larger universe of cases whose citations to BankXX cases are counted. In determining its leading cases, BankXX only counts the number of citations of cases in its knowledge base *by* other BankXX cases. This experiment counts the number of cites to BankXX cases by any case that includes “1325(a)(3) and good faith.”

While such a table is reasonably simple to construct automatically, it provides quantitative information that is only available qualitatively to legal researchers. When confronted with a mass of cases to read, a sorted list could provide one discipline to order one’s research. Second, the table gives some clues to the structure of a domain. An area such as good faith is dominated by a handful of important cases. It is a matter for future empirical research to determine if other legal domains are structured similarly. One might hazard a guess that this domination of an area by a small group of cases is typical, and that the reason is a cognitive bias inherent in human judges and commentators in favor of a graded categorical structure for legal areas and a small set of prototypical representatives. Speculations of this sort might provide interesting hypotheses for future research.

Summary for Section 6.2
Citation analysis and leading cases.

In this domain, leading cases may be identified easily through counting the inter-case citations extracted from the full text of legal opinions.

6.3 An Alternative Approach to Precision-Recall

In this section, we present an alternative approach to calculating precision and recall. In all the other calculations in this paper, BankXX has been credited with finding a case or a theory that appears in a legal opinion only if it finds that precise case or theory, that is, the names match literally. In some circumstances a broader sense of match might be preferable. For example, if a legal opinion contained a cite to a particular case supporting a well-established proposition of law, but BankXX retrieved a different but highly similar case supporting exactly the same proposition, the highly similar cases ought to be considered a “match.” Sets of similar cases—that in their argumentative value are essentially equivalent—are often presented in so-called string cites. Similarly, if BankXX found a legal theory that referred to exactly the same factors as another theory appearing in an opinion, then this is tantamount to finding exactly the same theory. Thus, while the information retrieval field has generally relied on an “exact match” to documents in the answer key to gauge retrieval success, in some circumstances, there are good reasons for using an alternative, slightly relaxed definition. In this section, we briefly explore the idea of awarding full match credit for cases that are highly similar.

In order to use an extended notion of match based on high degree of similarity, we require some measure of when two cases are sufficiently similar to be considered equivalent. Fortunately, legal CBR programs provide just such a measure. For instance, according to the model of case similarity provided by HYPO, two cases are highly similar with respect to a problem case if they appear in the same node of the standard claim lattice built for the problem case, which means that they are equally on-point and address a common set of relevant legal issues.[58]

We can extend the notion of a match by saying that two cases—one in the BankXX output and one in the hand-coded answer—are an ***extended match*** if either:

1. they are the **same**, or
2. they are **highly similar**, where two cases are considered ***highly similar*** if they reside in the same node of the claim lattice for the problem case.

This particular definition of *highly similar* defines an equivalence relation on cases. Motivated by this observation, we can generalize our definition to say that two cases are considered an extended match if they are in the same equivalence class of case similarity. Notice that we could add a third criterion to the extended match definition: that the two cases must be decided for the same viewpoint. This happens automatically in our application since we are only matching cases from aggregates which already reflect the same viewpoint.

For example, suppose the problem case is *Estus*. Suppose *Gunn* is mentioned in the opinion as a pro case, and is therefore in the answer key for *aggregated-pro-cases*. Suppose further that BankXX retrieves another pro case, *Gibson*, which is not mentioned.[59] Under the usual exact match criterion, *Gunn* would not match *Gibson*. However, since *Gibson* and *Gunn* appear in the same node of the *Estus* claim lattice as most on-point cases decided for the creditor, they are an extended match.[60]

---

[58]Of course, one could also use the extended claim lattice that takes note of nearly applicable dimensions as well.

[59]Note both *Gunn* and *Gibson* were decided after *Estus*, so they cannot appear in the hand-coded opinion or in the BankXX output when date-filtering is in effect. They are used here simply for illustration.

[60]Modulo the technicality that *Gunn* had not already been "matched" with another ADDITIONAL case retrieved by BankXX.

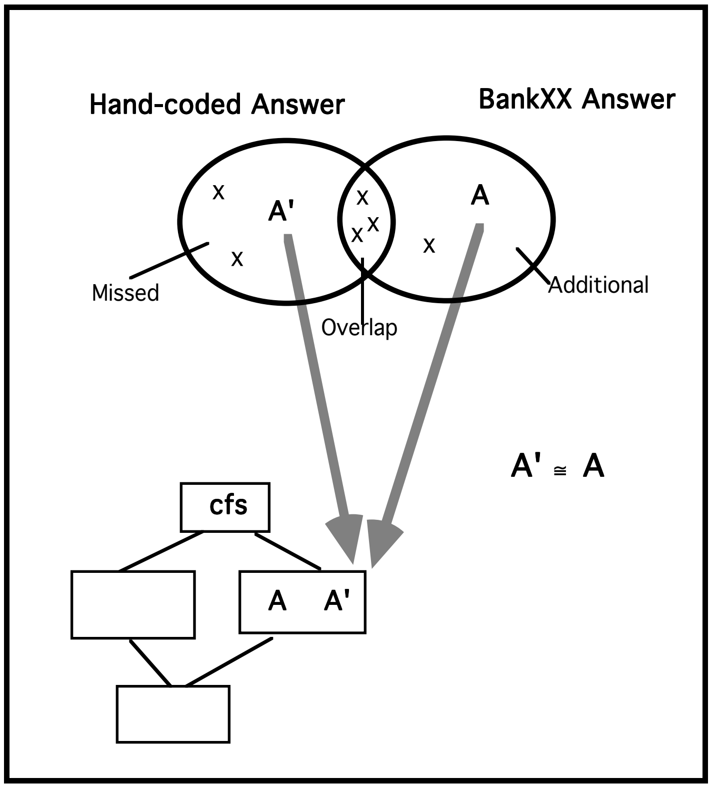

Figure 13. Schematic for extended match similarity between two cases A’ and A.

This notion of extended match gives rise to an alternative way of computing precision and recall, which was used in another set of experiments described here. Essentially we proceeded as usual until we computed precision and recall. That is, for each case in the case base, a version of BankXX was run on that case treated as a new problem situation, with the default parameter settings for BankXX.

We then examined the output of the three aggregated argument pieces that are filled with cases—*aggregated-pro-cases, aggregated-con-cases*, and *leading-cited-cases*—and computed new OVERLAP, MISSED, and ADDITIONAL sets based on extended matches. To do this, we first determined the standard sets of OVERLAP, ADDITIONAL and MISSED cases as usual. Second, we examined the ADDITIONAL and MISSED cases—the cases in the symmetric difference in Figure 13—to see if any pairs of cases—one from ADDITIONAL, one from MISSED—could be matched using the extended match criterion. (Recall, a claim lattice for the problem case is constructed as an initial step in BankXX's processing, so it is readily available for extended matching.)

The exact procedure involves marking cases so as to not allow more than one ADDITIONAL case to match a given MISSED case and vice versa. We want to maintain a one-to-one correspondence between the items we match between the MISSED cases and the ADDITIONAL cases so that BankXX does not receive too little or too much credit, which could occur if we allowed unlimited matches. For instance, we did not want to permit unlimited matches of a case appearing in the hand-coded answer (i.e., a MISSED case) with multiple cases appearing only in BankXX’s answer (i.e., ADDITIONAL cases).

Once a pair is considered an extended match, it is moved to the new OVERLAP (with the name used in the hand-coded answer) and each case of the pair is deleted from its respective ADDITIONAL or MISSED category.

After all cases in the original ADDITIONAL and MISSED sets were examined, and new, modified OVERLAP, MISSED and ADDITIONAL sets determined, we computed ***modified*** precision and recall as usual:

$\tilde{\mathbf{p}}$ = |new OVERLAP|/ |new ADDITIONAL $\cup$ new OVERLAP|

$\tilde{\mathbf{r}}$ = |new OVERLAP|/ |new MISSED $\cup$ new OVERLAP|

For each version of BankXX, we computed average $\tilde{p}$ and $\tilde{r}$ over the entire case base. See Table 4. As before we explored both options for dealing with 0/0 situations.

	AGGREGATED PRO CASES		AGGREGATED CON CASES	
	p, r	$\tilde{p}$, $\tilde{r}$	p, r	$\tilde{p}$, $\tilde{r}$
BankXX/NT	**24** (23_{54}), **76** (72_{47})	**25** (24_{54}), **79** (75_{47})	**23** (23_{55}), **84** (79_{42})	**25** (25_{55}), **91** (88_{42})
BankXX/AP	**32** (30_{53}), **55** (48_{47})	**34** (31_{53}), **57** (49_{47})	**42** (33_{48}), **64** (53_{42})	**44** (36_{48}), **67** (57_{42})
BankXX/AF	**30** (26_{52}), **56** (48_{47})	**38** (34_{52}), **63** (57_{47})	**33** (26_{50}), **63** (52_{42})	**37** (30_{50}), **70** (60_{42})

Table 4. Averaged precision and recall values calculated as usual and with extended matching. Values in boldface are computed with undefined ratios are set to 1. Average values computed only with respect to defined values are given in parentheses; the number used in an average is shown in the subscript.

In Table 4, we only show data for *aggregated-same-side-cases* and *aggregated-contrary cases* since for *leading-cited-cases* both methods of computing precision and recall produce nearly identical scores. (See Table 1 or Table 2 for precision-recall data computed in the usual way.) This is no surprise since there are only five cases that can possibly be listed as a *leading-cited-cases* and BankXX does very well on finding them, and thus there are few opportunities to use the extended match criterion.

For the most part, the differences between the usual and the extended method are all within a few (absolute) percentage points for BankXX/NT and BankXX/AP. The relative changes vary between 3.6% to 8.6%, when 0/0 is set to 1, and 2% to 11.3%, when undefined values are excluded. With only one exception, the relative changes are all less than 10%. Thus, the differences here are not always significant.

However the numbers do change significantly for BankXX/AF. The relative differences range from 11% to 26.6%, when 0/0 is set to 1, and 15.3% to 38.3%, when undefined values are excluded. There is a tremendous jump in precision for *aggregated-pro-cases*. These relative changes can be considered significant.

The higher performance scores for BankXX/AF under the modified precision-recall scheme shows that BankXX/AF is actually finding cases that are dimensionally highly similar to the ones cited in the opinions. This is evidence that BankXX/AF—and the argument factors that drive it—are having greater success than meets the eye under the usual precision-recall standard.

In situations where there are more cases that can be considered, there is a greater difference between precision and recall scores based on the extended match criterion and those based on the usual literal match criterion. In our experiments, this occurs typically with later cases, where more cases survive date filtering.

We did not experiment with an extended match approach on *aggregated-theories* because we had not defined a metric to measure similarity between theories. One obvious approach is to sort theories according to the degree of overlap between their sets of defining factors in a manner analogous to HYPO-style case similarity. Another approach would be to measure theory similarity by measuring the similarity of the cases applying the theory. There are many possibilities.

This experiment shows that more perspicacious means of computing retrieval success may be available when additional domain knowledge is used to inform an extended matching metric. The modified scheme discussed here is one of many possibilities. It is, we believe, more in the spirit of the way arguments are probably evaluated by legal professionals and law professors—sometimes one case is just as good as another for supporting a legal proposition.

7. Concluding Remarks

There are some general lessons to be learned from our experiences in this evaluation study, which, we believe, is the first such in-depth study in AI and Law. First, evaluation is a long, painstaking process. Second, using the evaluation standards taken from some other area of computer science, like information retrieval, has its pros and cons. On one hand, metrics like traditional precision and recall are widely used. On the other hand, they may not be totally appropriate in domains such as the law where the existence of an unassailable correct answer is doubtful. We need new ways to assess performance. Furthermore, there is a tendency to lift such measures out of context. Without context and comparable data from other systems on commensurable problems and data sets, it is impossible to say whether a particular level of precision or recall is "good" or "bad." However, we feel that evaluation is important and beneficial. We have tried to demonstrate how the problem of evaluation can be tackled.

In general, we found that many of our original hypothesis were true, but some in more qualified terms:

> • BankXX/AP and BankXX/AF did perform better than BankXX/NT but only with regard to precision. In fact, we found a classic trade-off: The knowledge-poorer BankXX/NT achieved higher recall but the knowledge-richer BankXX/AP and BankXX/AF achieved higher precision.
>
> • BankXX/AP did produce much more balanced arguments than BankXX/NT.
>
> • BankXX did exhibit performance improvement over HYPO (re-implemented as μHYPO) on the task of retrieval in the sense that BankXX retrieves information used in the court opinions not found by HYPO. In direct comparison with hand-coded answers, BankXX achieved higher precision and recall in all categories with one surprising exception: HYPO was more precise than BankXX on legal theories.
>
> • With BankXX/NT and BankXX/AP, more resources did not appear to provide much marginal benefit after a point. With BankXX/AF, additional time is a help because the calculation of some of the argument factors is so slow.

Some additional specific lessons from our experiments are:

• <u>There was variation in BankXX performance on different types of sets.</u> In particular, there was high performance on *meaty* and other contentful cases and low performance on *sparse* cases. This suggests that in building future case bases, we would bias our acquisition towards meatier cases and be aware of the difficulties that can arise with sparse ones.

• <u>As a corollary, statistics aggregated over different types of cases were quite different.</u> Understanding the statistics often required detailed examination of the program's product on individual problems, and even sometimes of its internal data structures (e.g., open, closed lists). This reinforces the old wisdom that averaging is no "excuse" for detailed examination of the data.

• There was a surprising amount of monotonicity for precision-recall scores calculated with respect to various subsets of cases, different aggregated argument pieces, and the three BankXX configurations. There were analogous nesting relationships among the sets of OVERLAP, ADDITIONAL, and MISSED cases. These pointed to various trade-offs (regarding the evaluation functions) and correlations (regarding sparse and contentful cases) in performance.

• The fill limits used by BankXX/AP had profound effects. On the positive side, they caused BankXX/AP to harvest a nicely balanced array of information. On the negative side, they caused BankXX/AP to forgo harvesting some very useful information.

• Surprisingly good performance can be achieved with BankXX's two-stage process of first retrieving objects and second amalgamating them into argument pieces, even when the first stage is relatively knowledge-poor, as is the case with BankXX/NT. This demonstrates the efficacy of the generate-and-test or "many are called/few are chosen" (MAC/FAC) approach for legal information retrieval.

• BankXX/AP and BankXX/AF were very sensitive to problem-solving context. By contrast, BankXX/NT was a rote, brute force retriever. Surprisingly, BankXX/NT did quite well—better in fact than the other two—on recall.

• Different versions of BankXX had different retrieval biases. These reflected the biases of the evaluation functions.

• There was a trade-off in computational costs. The knowledge-rich BankXX/AF was much slower than either BankXX/NT or BankXX/AP.

• The setting of some internal parameters did not make much difference. The start node did not matter particularly (in our highly interconnected space of information). But there was a point at which additional resources of time and space did not provide additional benefits.

• BankXX and HYPO exhibited different and complementary retrieval biases. One can say that there was value added by BankXX's approach but more importantly perhaps, BankXX and HYPO could complement each other in some larger composite program.

As a general conclusion, we feel that BankXX/AP with its standard configuration provides good value and is "well-behaved" from many points of view. BankXX/AP imports an intermediate level of representation of argument into the process of retrieval through its argument piece evaluation function. BankXX/NT uses a coarser representation, and BankXX/AF, a finer one. BankXX/NT is a little too quick-and-dirty and BankXX/AF is a little too thoughtful. BankXX/AP seems about right.

It may be a common experience for researchers to perform an extensive series of experiments on a program, only to realize that they should have implemented some features differently. Certainly that has been our experience. Small design decisions long since made and forgotten were dredged up through close examination of the product of the program. One small example is our decision not to add all the most on-point cases to the open list when initializing the search. Currently, only a random most on-point case node is placed on the list. Another is our decision not to reason about the date, jurisdiction, or pedigree of a case. The rigors of evaluation forced us to address the date of a case even though the problem of post-dated cases does not arise in the intended usage of BankXX on truly new problem cases, where everything known by the program should be available to it. These are examples of design decisions we would now modify.

Of course, there are always variations and extensions to a program that could be made and evaluated. For instance, based on our experiences we would now use a different set of weights and fill limits on certain items. In particular, we would increase the limits on the numbers of ordinary pro and con cases that can be harvested. We would also use a concomitant standard configuration of BankXX/AP such that the limit on closed nodes is larger than sum total of the individual fill limits.

In addition, introspection suggests that legal researchers change heuristic evaluation of legal information as search progresses. At the beginning of research, a variety cases may be read, to get a sense of the area. Once the skeleton of the argument is formed, however, the researcher's information needs (e.g., "the argument needs more applicable legal theories") become more precise. As resources are about to run out, the emerging argument might dictate very much more closely what kinds of cases (e.g., a trumping appeals case from our circuit) and other support are needed. These observations suggest possible modifications to the overall architecture of our system. For instance, our program could change its evaluation function in mid-run after a certain threshold amount of time or space has been searched, perhaps analogous to the opening, mid-game and end-game stages in game-playing. In the "opening," it could use the node type evaluation function to get a quick-and-dirty start on retrieval. In the "mid-game" (probably, the longest-lasting phase), it could use the argument piece evaluation to retrieve a well-balanced set of information that is not overloaded in any one category or empty in others. In the "end game," it could use the judiciously deliberative BankXX/AF to find just the right sort of information to strengthen the argument. Our program might also use its runs with the less smart evaluation function (BankXX/NT) as a "learning experience" for subsequent runs.

Although we performed a wide variety of experiments with the BankXX program, it may well take future experiments with similar programs to extract the BankXX-dependent and domain-dependent conclusions from more general lessons about legal retrieval and argument.

General lessons that we would expect to see emphasized by work with other programs and domains include the utility of incorporating a variety of types of legal knowledge in a knowledge base, together with a rich vocabulary of inter-connections. We would expect to see the benefit of retrieving information for targeted categories, such as those captured in our argument pieces since this imports the high level purposes for doing the retrieval—the user's information needs—into the retrieval process itself. We would expect also to see in various forms the domination of small sets of cases and the importance of leading cited cases to structuring a legal area. Finally, we would expect to see that argument creation involves the complex interaction of pieces of knowledge represented and evaluated at various levels of abstraction.

Acknowledgments

This work was supported in part by the Air Force Office of Scientific Research under contract 90-0359 and the National Science Foundation under grant EEC-9209623, State/Industry/University Cooperative Research on Intelligent Information Retrieval. For assistance we thank Matt Cornell and Kate Sanders, and Elaine Whitlock and Barbara Fell-Johnson of the Hampshire County Law Library, Northampton, Massachusetts. Jody Daniels wrote the claim lattice algorithms used by BankXX. We thank the West Publishing Co. for providing access to their legal databases through the University of Massachusetts Center for Intelligent Information Retrieval.

References

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

Branting, L. K. (1991). Building Explanations from Rules and Structured Cases. *International Journal of Man-Machine Studies*, 34, 797-837.

Eisenberg, M. A. (1982). *Gilbert Law Summaries: Contracts*. New York, NY: Harcourt Brace Jovanovich Legal and Professional Publications,

Fox, J. & Clarke, M. (1991). Towards a Formalization of Arguments in Decision Making. *AAAI Spring Symposium Series, 1991, Argument and Belief*, 92-99. Palo Alto, CA.

Friedman, M. T. (1994). *Information Retrieval Techniques Applied to Indexing for a Case-Based Reasoning System*. Unpublished Manuscript, Dept. of Computer Science, University of Massachusetts, Amherst, MA.

Gardner, A. vdl. (1987). An Artificial Intelligence Approach to Legal Reasoning . MIT Press, Cambridge.

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

Perelman, C. & Olbrechts-Tyteca, L. (1969). *The New Rhetoric: A Treatise on Argumentation* . Notre Dame, Indiana: University of Notre Dame Press.

Rissland, E. L. (1990). Dimension-based Analysis of Hypotheticals from Supreme Court Oral Argument. *Proceedings of the Second International Conference on AI and Law*, 111-120. Vancouver, BC.

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., Valcarce, E. M. & Ashley, K. D. (1984). Explaining and Arguing with Examples. AAAI-84, Proceedings of the National Conference on Artificial Intelligence. Austin, TX. American Association for Artificial Intelligence.

Salton, G. (1989). Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer . Reading, MA: Addison-Wesley.

Skalak, D. B. & Rissland, E. L. (1992). Arguments and Cases: An Inevitable Intertwining. Artificial Intelligence and Law: An International Journal, 1(1), 3-48.

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

Technical Appendix

Here we give for reference the weights and terms used by each evaluation method. The companion article contains a full description of the evaluation functions.

1. *Theory-nodes*	8
2. *Cases-as-facts nodes*	6
3. *Citation-bundle nodes*	5
4. *Domain-factor nodes*	4
5. *Story-prototype nodes*	3

The terms and weights used in the node type evaluation function.

1. *supporting cases*:	weight=2	[limit=3]
2. *best supporting cases*:	7 [5]	
3. *contrary cases:*	1 [3]	
4. *best contrary cases:*	5 [3]	
5. *leading cases*:	6 [5]	
6. *supporting citations*:	1 [5]	
7. *overlapping cases*:	1 [5]	
8. *applicable legal theories*:	8 [6]	
9. *nearly applicable legal theories*:		6 [3]
10. *factual prototype stories*:	6 [1]	

The terms, weights, and piece limits (given in brackets) used in the argument piece evaluation function.

1.centrality-of-theory	8
2. win-record-of-theory	8
3. win-record-of-theory-for-factual-prototype	8
4. strength-of-best-case-analogies	5
5. centrality-of-best-cases	5
6. equally-on-point-casees 4	
7. strength-of-citations	4
8. strength-of-factual-prototype	3

The terms and weights used in the argument factor evaluation function.

Notes:

2. 26,500 text words; 2,200 footnote words; 28,500 total words