Analogical Reasoning, Generalization, and Rule Learning for Common Law Reasoning

Joseph Blass
Qualitative Reasoning Group
Department of Computer Science & School of Law
Northwestern University
Evanston, Illinois, USA
joeblass@u.northwestern.edu

Kenneth D. Forbus
Qualitative Reasoning Group
Department of Computer Science
Northwestern University
Evanston, Illinois, USA
forbus@northwestern.edu

ABSTRACT

Research in AI & Law has sought to model common-law case-based reasoning by creating analogies from cases, extracting and applying rules from cases, or both. This paper presents a new approach to extracting legal information from cases and several methods to apply that information to new cases, including by analogy and by converting what the system has learned into logical rules. It evaluates these approaches on a recently introduced legal dataset and compares results to off-the-shelf machine-learning techniques. We conclude that abstract legal information can be extracted from similar cases through analogical generalization, and that the extracted legal schemas can be used to reason about and solve subsequent cases both by analogy and by rules.

KEYWORDS

Analogy, Precedential Reasoning, Rule Learning, Legal Schemas

1 Introduction and Background

In Common Law legal systems, cases are resolved by reference to and consistent with prior decisions settling similar claims. While there is debate within the legal academy over whether precedents are applied by analogy to new cases [1], encode rules revealed through precedential analysis [2, 3], or declare rules outright (with analogies used for illustration) [4, 5], it is clear that if a prior case in a jurisdiction concerns the same legal issues as some case at bar, the prior reasoning and decision governs the latter. As one of the central pillars of the Western legal tradition, common law legal reasoning has been a focus of the AI & Law research community.

This paper introduces a new approach to computational precedential reasoning, inspired by and adapting tools from cognitive science. Our three algorithms perform legal reasoning using analogical generalization, analogical case-based reasoning, and rule-based reasoning. Taken together, these three algorithms define a process by which a body of case law can go from being a disconnected set of cases applied directly to a new case at bar, to an abstract representation of what those cases have in common, and finally to formal rules governing those cases, even as the cases continue to be used for reasoning. We evaluate these algorithms using the Illinois Intentional Tort Qualitative Dataset [6], a dataset of structured, relational cases in predicate logic, extracted from Illinois tort cases using a natural language understanding system.

This research builds upon AI & Law research on precedential reasoning, analogy, and rule-learning. GREBE used first-principles and analogical case-based reasoning to evaluate claims for worker-compensation [7]. Given case facts in propositional logic, GREBE used first principles to derive facts known to be relevant to such claims, then used backchaining and an implementation of the Structure Mapping model of analogical reasoning [8] to create mappings to precedents. The result of the precedent mapping the greatest portion of the current and past cases was applied to the new case. Precedential reasoning was used to derive intermediate propositions in the larger rule-based reasoning system.

The HYPO family of legal reasoning, case-based reasoning, and argumentation resolves cases through an analogical dialogue [9-11]. HYPO-style reasoners operate over *factors*, legally relevant concepts—essentially labeled fact-patterns—that may be present in a situation. Given a case (encoded as factors), a HYPO algorithm operates in three steps. The case in the library sharing the most factors with the new case is first retrieved as an analogue, and its outcome proposed for the new case. The system then argues against itself, by distinguishing the retrieved case, proposing a different analogue, or reinterpreting the mapping. Finally, the system responds to its counterargument. Factors have become a primary representational formalism in AI & Law. Research using factors has relied on humans to identify the set of possible factors, though once identified factors can be automatically associated with case descriptions using AI textual analysis techniques [12].

ICAIL 2023, June 19–23, 2023, Braga, Portugal
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0197-9/23/06…$15.00
https://doi.org/10.1145/3594536.3595121

Other research represents cases as logical statements that, along with the outcomes of the case, encode rules and arguments. Horty [13] developed a logical formalism that weights rules extracted from precedents and preferentially applies them to new cases using formal logic. Verheij [14] uses sets of cases that are logically consistent, different, and mutually compatible, to learn *case models* that capture doctrinal rules that the cases collectively encode. Cases can be applied through the rules they encode, or by applying a case's outcome based on statements it shares with the new case. These approaches allow the rules governing a system to be used without having access to the entire rule set. They demonstrate how to leverage legal case bases and to extract rules from formalized cases, but assume a representational formalism not easily extracted from natural language case descriptions.

Van Woerkom and colleagues [15] developed a system that reasons over a dataset of real-world recidivism cases represented as *dimensions*, which are like continuous factors. This system determines from the data which dimension values favor which outcome, then uses that information to identify *landmark* cases which should maximally constrain other cases in the dataset. This approach may be effective for domains where case information is highly formalized (the data in question included information about age, sex, number of prior offenses, etc.). That said, the system does not learn rules across the cases, and representing cases in other domains as dimensions may require human involvement.

Our research departs from these prior approaches by avoiding using human-encoded specialized legal knowledge while maintaining the expressivity of encoding cases in predicate logic. Instead, legally-relevant fact patterns are identified by comparing the raw events of precedent cases. This case comparison, using a computational model of analogy inspired by human analogical reasoning, reveals which facts are significant.

2 Structure Mapping

When lawyers search through precedents to find one on point for their case, they do not forget each precedent they examine before moving on to the next. Humans naturally recognize analogies and form schemas when examining similar cases, a comparison which invites abstraction and helps discover the principles governing those cases [16]. Thus, regardless of the role analogy may play in resolving a new case given its precedents, analogical learning across the precedents themselves might be a means to reveal the rules that govern them and therefore apply to the new case.

Regardless of whether this is how human lawyers extract rules from precedents, it also provides a method to determine what the content of rules actually are, as opposed to what a judge has said a rule is in any given case. That is, a judge may declare a rule that factors A, B, and C lead to outcome X in some case, only for some future judge in a case lacking factor C to say that the prior judge misunderstood their own rule, and that factors A and B are sufficient for outcome X. An approach that discerns rules by comparing multiple precedents could provide a more accurate understanding of what the rule governing some domain is, rather than treating a single declaration of the rule as dispositive.

The Structure Mapping Engine (SME) [17] is a computational model of analogy and similarity based on Structure Mapping Theory [8]. SME takes two structured, relational cases—*base* and *target*—and computes up to three *mappings* between them. A mapping contains the alignment between the cases (entity and expression *correspondences*), *candidate inferences* (CIs) suggested by the alignment, and a *similarity score* measuring match quality. If projecting a CI involves an entity not in the case, it will be postulated as a skolem (i.e., a new variable). Our systems reason using an ontology and knowledge base built atop OpenCyc [18].

Running SME across all cases in memory is prohibitively expensive and cognitively implausible. MAC/FAC [17] models analogical retrieval based on evidence regarding human memory retrieval. It uses a *probe* case to perform a *reminding* over a case library, outputting SME mappings between the probe and retrieved cases. MAC/FAC first efficiently computes dot products between content vectors of the probe and each case in the library (a coarse measure of similarity). Up to three cases are then passed to SME, which returns up to three mappings between them and the probe.

SAGE [19] is a model of analogical generalization built on SME and MAC/FAC. Given a new case, SAGE uses MAC/FAC to retrieve a similar case or generalization. If the SME score between the retrieved case and the probe is above an assimilation threshold, SAGE generalizes the cases together using that mapping; else the case is added to the library ungeneralized. SAGE generalizations are a joint distribution over the facts of all constituent cases, with each fact stored together with its probability (the proportion of cases in that generalization that contain it). Generalizations can be used as cases for SME comparisons, with facts whose probability falls below a preset threshold (the *probability cutoff*) excluded from the generalization's reasoning case. SAGE reasoning occurs within a particular *generalization pool* (gpool), a case library containing generalizations and ungeneralized outliers.

This research uses a modified version of SAGE called Conclusion-verified Analogical Schema Induction (CASI) [20]. CASI involves knowing the form of a case's *conclusion*—a target predicate—and excluding it during retrieval and mapping. Instead of using similarity scores to control generalization, CASI checks whether the case's conclusion is amongst the candidate inferences in the mapping from the retrieved case to the probe. If so, the mapping would be sufficient to solve the probe case were the conclusion unknown. CASI uses that mapping to generalize the case (and its conclusion) with the retrieved case.

3 Analogical Generalization and Reasoning for Common Law Reasoning in Tort

Here we describe our three experimental algorithms. All three algorithms involve using CASI to build generalizations of legal cases before reasoning about held-out cases in light of what the system has learned through generalization. These algorithms together illustrate a cycle by which rules can be extracted from a body of case law even as the cases are used for non-rule-based precedential reasoning. The first algorithm is Purely Analogical Precedential Reasoning (PAPR), which involves using SME to reason by direct analogy about held-out cases in light of the learned generalizations and ungeneralized cases. The second algorithm, Analogical Reasoning with Positive Generalizations (ARPG), also uses SME to reason about held-out cases, but only in light of positive generalizations, not negative ones or ungeneralized examples. ARPG resolves cases by reasoning *about* the constructed analogies, not directly with them. The third algorithm, Reasoning with Rules Learned from Generalizations (RRLG), transforms the positive generalizations used in ARPG into Horn clauses, and applies those rules to held-out cases with back-chaining.

3.1 Purely Analogical Precedential Reasoning

Our first algorithm, Purely Analogical Precedential Reasoning (PAPR), is presented in Figure 1. PAPR starts by gathering all cases in the dataset in the same doctrine as the case at bar and creating generalizations from them using CASI. (The doctrine is always known: one can know of what one is accused without knowing what the outcome of the case will be). One gpool is used for positive cases (cases where a party was found liable) and another for negative cases (where no party was liable). PAPR performs a reminding over the union of positive and negative gpools using the case at bar as a probe, retrieving cases from the library and outputting mappings from those cases to the one at bar. The CIs from the top mapping that represent conclusion statements (i.e., hypothesized solutions) are inspected, and inferences with skolem variables are rejected. The algorithm returns as its case outcome the skolem-free conclusion CI from its top-scored mapping. To separate the system's capacity to get the correct answer from its ability to get it right the first time, we also ran a condition where the system was able to check its answer against the held-out truth and examine other mappings or perform more retrievals. In this latter condition the system examines up to six mappings that contain a grounded conclusion statement, for Precision@6.

```
Given case c, case set cs, conclusion predicate cp:
PAPR(c, cs):
  1. pos = positiveCases(cs); neg = negativeCases (cs)
  2. Gpools pg = CASIgeneralize(pos);
            ng = CASIgeneralize(neg)
  3. Probe pc = nonConclusionFacts(c)
  4. Gpool Case Library cl = union(pg, ng)
  5. Reminding r = reminding(pc, cl)
    6. If r:
      7. For mapping m in r:
        8. Inferences CIs = candidateInferences(m)
        9. groundCIs = {i in CIs if not containsSkolem(i)}
        10. If groundCIs:
          11. For inf ∈ groundCIs:
            12. If predicate(inf) = cp:
              13. Return inf as conclusion to c
        14. Else: [if here, m did not solve c, try others]
    15. Else: [no mapping in r solved c; strike rejected cases]
      16. Retrieved cases retr = retrievedCases(r)
      17. Updated Case Library cl = cl – retr
      18. [go to step 5].

Note: For Precision@N testing, replace step 13 with these:
              13(a). depth += 1 [depth is initialized to 0]
              13(b). If depth > N: Return fail. [N reached]
              13(c). Conclusions SC = {conclusion(c)}
              13(d). If inf ∈ SC: Return inf
```

Figure 1: Purely Analogical Precedential Reasoning (PAPR) Pseudocode

3.2 Analogical Reasoning with Positive Generalizations

The second algorithm, Analogical Reasoning with Positive Generalizations (ARPG), is presented in Figure 2 and reflects the fact that negative cases are united largely by not being positive. Negative cases might help define the boundaries of a legal generalization [19], but it is positive cases that demonstrate what a legal claim is. ARPG reasons from positive generalizations, measuring whether learned schemas distill facts relevant to a legal claim. And consider what these generalizations contain: only the high-probability facts common to the cases that participate in the generalization. We hypothesize that legal cases are sufficiently varied that a broad variety of positive legal cases should *only* have in common the facts relevant to the legal claim, with facts idiosyncratic to particular cases but irrelevant to the claim stripped away through the learning process. The generalizations should be self-contained legal principles that associate a legal conclusion with an abstract set of facts that lead to it. (Note that we do not use "legal principles" to refer to norms or rules, but simply sets of abstract facts associated with outcomes.) Analogizing an unsolved legal case that contains all the elements of some claim to that generalization should generate only one CI: the case conclusion.

ARPG starts by creating generalizations of positive cases (excluding the case at bar), then discards ungeneralized examples. It uses the case at bar as a probe to retrieve over the generalizations, looking for skolem-free conclusion CIs. ARPG then takes the best mapping with a grounded case conclusion CI and examines how many *other* CIs it contains. If it has other CIs, ARPG concludes that the case is a negative case, with the other CIs corresponding to the missing elements of the claim. If the conclusion statement is the only CI, ARPG concludes it is a positive case.

Note that in general the presence of additional CIs should not count against whether a generalization is a good analogical match for a new case. Schema-building does not generally remove all idiosyncratic case facts, since some irrelevant facts may be highly

```
Given case c, case set cs, conclusion predicate cp, number of extra
candidate inferences tolerated error:
ARPG(c, cs):
  1. pos = positiveCases(cs)
  2. gpool of all cases gAll = CASIgeneralize(pos)
  3. reasoning gpool g = generalizationsInGpool(gAll)
  4. Probe pc = nonConclusionFacts(c)
  5. Reminding r = reminding(pc, g)
    6. If r:
      7. For mapping m in r:
        8. Inferences CIs = candidateInferences(m)
        9. groundCIs = {i in CIs if not containsSkolem(i)}
        10. If groundCIs:
          11. For inf ∈ groundCIs:
            12. If predicate(inf) = cp:
              13.  otherCIs = CIs - inf
              14. If count(otherCIs) > error:
                   [If # non-conclusion CIs > error, negative]
                15. Return (not inf) as conclusion to c
              16. Else: Return inf as conclusion to c
        17. Else: [if here, top mapping did not solve c, try others]
    18. Else: [if here, no mapping in r solved c; strike reject cases]
      19. Retrieved cases retr = retrievedCases(r)
      20. Updated Case Library g = g - retr
      21. [go to step 5].

Note: For Precision@N testing, replace steps 13–16 with these:
              13. depth += 1 [depth is initialized to 0]
              14. If depth > N: Return fail. [N reached]
              15. otherCIs = CIs – inf
              16. If count(otherCIs) > error:
                17. If isNegativeCase(c):
                  18. Return (not inf) as conclusion to c
                19. Else: [Wrong answer, keep trying]
              20. Else: [The conclusion CI is the only CI]
                  21. Conclusions SC = {conclusion(c)}
                  22. If inf ∈ SC: Return inf
```

Figure 2: Analogical Reasoning with Positive Generalizations (ARPG) Pseudocode

correlated with case outcomes and therefore will be present in the schema. If so, they eventually will get hypothesized as a candidate inference for a case where the correlate happens not to be true. For example, if building a schema about the concept "dinner," one high-probability fact may be that it is an evening meal; when using the schema to reason about a festive afternoon dinner, SME might hypothesize that the meal occurs in the evening. The presence of that CI does not mean that afternoon meal is not dinner. But by hypothesis the legal domain is an exception, because the legally-relevant facts that unite legal cases are fairly abstract. Abstract facts are less situationally-specific and will thus share fewer correlative facts across cases. And even if this hypothesis about legal schemas is wrong, ARPG's reasoning method can still be useful—it will just require some process after schema-building to identify and remove those extraneous facts from the generalization.

ARPG can use a partial truth check by tolerating extra CIs, i.e., whether extra CIs besides the case conclusion can be tolerated before concluding that the case is negative. Tolerating extra inferences allows the system to reason with generalizations that may not have stripped away all legally irrelevant facts. As with PAPR, in addition to evaluating the ARPG on its first returned answer, we evaluated it using Precision@6: in this condition, if ARPG was either mistaken about what kind of case (positive or negative) the case at bar is, or if it correctly concludes that it is a positive case but generates the wrong conclusion, it will move on to other mappings or perform a re-retrieval, examining up to six mappings with grounded case conclusions.

Where PAPR's performance can be properly understood to measure the extent to which analogy can be used to solve legal cases (by analogy to other cases and generalizations), ARPG is more a measure of the extent to which analogical generalization can learn accurate and useful legal principles. ARPG simply will not work if it does not have sufficiently clean generalizations of legal principles (this point is further explored in [20]). Thus PAPR is a measure of the usability of SME as a legal reasoning technique, whereas ARPG measures the effectiveness of analogical generalization in learning legal concepts.

In this dataset, each solution takes the form of a ternary predicate, expressing who did what to whom. For example,

```
    (assaultsPartyByDoing Fred Rick punch123)
```

means that `Fred` assaulted `Rick` when he performed the action `punch123`; similarly,

```
    (not (trespassOnPropertyByAction Carl
                               lawn456 walk789))
```

means `Carl` did not trespass on the property `lawn456` by taking action `walk789`. Given this, we can evaluate PAPR and ARPG using a *partial truth check* to measure of the *extent* to which a mapping produces the right answer. The partial truth check requires the first argument in the ternary predicate to be correct (i.e., the accused must be correctly identified), but then is satisfied with only one of the two remaining arguments. That is, it would rate

```
    (assaultsPartyByDoing Fred Rick kick246)
```

and

```
    (not (trespassOnPropertyByAction Carl
                               house357 walk789))
```

as partially true statements of the above conclusions.

3.3 Reasoning with Rules Learned from Generalizations

The final algorithm, Reasoning with Rules Learned from Generalizations (RRLG), closes the loop on how legal *rules* (not just *principles*) are extracted from precedents and applied to a case at bar, by converting ARPG's positive generalizations into rules and using them to reason about a new case. RRLG is presented in Figure 3. RRLG begins with the same generalizations generated for ARPG. It discards all generalization facts below ARPG's probability threshold, preserving the same facts that would be put into a generalization's case for reasoning by analogy; our experiments used a probability cutoff value of 0.6. Next, RRLG converts all generalized entities into logical variables. Although it was not relevant here, RRLG does not replace ungeneralized entities with variables, since any ungeneralized entity participating in a high-probability fact is involved in several cases, and is therefore presumably important to the principle governing them. Next RRLG produces a Horn clause, installing the variablized legal conclusion as the consequent of the rule and all other facts as antecedents. (The conclusion is again identifiable based on its predicate.) Finally, RRLG filters rules whose antecedents do not bind all consequent variables. To apply RRLG to a case at bar, the algorithm queries for the case at bar's conclusion in a query context that shares the facts of the case and RRLG's learned rules. For positive cases, it labels the case as being correct if it is able to derive the correct legal conclusion; for negative cases, it labels the case as being correct if it is unable to derive the legal conclusion.

These three algorithms define a spectrum regarding the role of analogy in legal reasoning. All three use analogical generalization to find principles common to legal cases, but how those principles are used varies by technique. In PAPR the system seeks to resolve cases purely by analogy: generalizations might improve analogical reasoning, but generalized and ungeneralized—and positive and negative—cases are equally informative. ARPG still reasons by analogy but restricts itself to the learned principles and reasons *about* the analogies, not just *using* the analogies. Finally, in RRLG the case at bar is not reasoned about by analogy at all, but using rules extracted from the learned legal principles.

```
Given case c, case set cs, conclusion predicate cp, probcutoff p:
RRLG(c, cs):
  1. ruleset = {}
  2. pos = positiveCases(cs)
  2. gpool of all cases rawG = CASIgeneralize(pos)
  4. reasoning gpool g = rawG - outliers(rawG)
  5. For genl ∈ g:
     6. highProbFacts = {f for f in genlFacts(genl) where P(f)>p}
     7. genEntVars = generalizedEntities(genl)
     8. logicVars = {(i, logicVar(i))} for i in genEntVars
     9. rawConc = conclusion(genl) [of form cp(X)]
     10. rawAntes = {highProbFacts – rawConc}
     11. ruleConc = replaceVars(rawConc, logicVars)
     12. antes = {replaceVars(a, logicVars)} for a in rawAntes
     13. Horn = makeHornClause(ruleConc, antes).
     14. If ∀ variables(ruleConc) ∈ variables(antes):
        15. ruleset += Horn
  16. Case facts cf = nonConclusionFacts(c)
  17. Conclusions Concs = {conclusion(c)}
  18. For conc ∈ Concs:
     [query the case conclusions given case facts and learned rules]
     19. ans = query(conc, cf + ruleset)
     20. If ans: Return ans
  21. Else: Return "c is a negative case" [no conclusions derived]
```

Figure 3: Reasoning with Rules Learned from Generalizations (RRLG) Pseudocode

4 Experimental Validation

We tested our methods against each other and several off-the-shelf machine learning baselines, on Assault, Trespass, and Battery cases from the Illinois Intentional Tort Qualitative Dataset [6]. This dataset includes historical Illinois common law cases in tort, including the original case facts and legal conclusions, a simplified English description of those case facts that is machine-readable by the CNLU natural language understanding system [21], the predicate logic representations CNLU extracts from the simplified descriptions of those case facts, and predicate logic statements of the case's legal conclusions. The experimental dataset includes 17 assault cases (12 positive, 5 negative), 40 battery cases (30 positive, 10 negative), and 43 trespass cases (29 positive, 14 negative). The greater number of positive than negative cases reflects the fact that judges are more likely to publish case opinions in cases where a party is found liable than those where they were not.

We ran RRLG, PAPR, ARPG and several baselines on these cases, varying PAPR and ARPG's parameters. First, we ran both PAPR and ARPG using both the strict and the partial truth tests. We also ran ARPG with a tolerance for either 0, 1, or 2 extra CIs allowed when inferring a positive case, to determine whether the generalizations generated might be noisy. Finally, we tested PAPR and ARPG using both Precision@1 and Precision@6 testing. For Precision@1 the system would return the first answer it generated with a grounded conclusion inference, but for Precision@6, the system would check its answer against the held-out ground truth: if it was wrong and the system had not yet checked six mappings, it would move on to other mappings and retrievals. (RRLG can only generate one answer to its queries, so Precision@6 is impossible).

SAGE, like humans, is sensitive to the order in which it receives cases. We tested both PAPR and ARPG on generalizations from a randomized case order (randomized for each new case at bar, i.e., not the same random order across cases), and on generalizations from a hand-selected case order that grouped like cases together for input into CASI. Because the non-random order was more effective, we only ran RRLG using the hand-selected case order.

For baselines, we used two BERT models and two GPT models. All models were retrieved from HuggingFace's library and were tested on the simplified English descriptions of the cases that had been used to generate the predicate logic representations operated over in our experimental conditions. Our BERT models were roBERTa and legalBERT, a version of roBERTa pretrained on legal text. To test our BERT models we turned our cases into multiple-choice tests and had the model select the best answer. Three false solutions to each case were generated by taking the original case solution and reversing the parties' roles, negating the original conclusion, and negating and reversing the conclusion. Like RRLG, our BERT model always returned the same answer, so we did not test it using Precision@N.

Technique Name	Algorithm	Training Order	Truth Check (Strict/Partial)	# Extra CIs Allowed?	Precision@1 Testing?	Precision@6 Testing?
RRLG	RRLG	Nonrandom	Strict Truth	-	Yes	No
PAPR-NR-ST	PAPR	Nonrandom	Strict Truth	-	Yes	Yes
PAPR-NR-PT	PAPR	Nonrandom	Partial Truth	-	Yes	Yes
ARPG-NR-ST-0CIs	ARPG	Nonrandom	Strict Truth	0	Yes	Yes
ARPG-NR-ST-1Cis	ARPG	Nonrandom	Strict Truth	1	Yes	Yes
ARPG-NR-ST-2CIs	ARPG	Nonrandom	Strict Truth	2	Yes	Yes
ARPG-NR-PT-0CIs	ARPG	Nonrandom	Partial Truth	0	Yes	Yes
ARPG-NR-PT-1Cis	ARPG	Nonrandom	Partial Truth	1	Yes	Yes
ARPG-NR-PT-2CIs	ARPG	Nonrandom	Partial Truth	2	Yes	Yes
legalBERT	l-BERT	Random	Strict Truth	-	Yes	No
GPT-J-ST	GPT-J	Random	Strict Truth	No contradictions	Yes	Yes
GPT-J-PT	GPT-J	Random	Partial Truth	Any	Yes	Yes

Table 1. Reasoning Techniques Tested (sans roBERTa, GPT-2, and PAPR or ARPG with random training order)

The GPT models were GPT-2 and GPT-J, a model based on GPT-3 with 1.3 billion parameters. We fine-tuned and tested our GPT models using 5-fold cross-validation. We tested GPT models using the case problem as a prompt and having the model generate up to 100 tokens as a completion six different times. We examined case completions to determine whether they contained the correct solution to the case, both looking only at the first answer generated (Precision@1) and at all six (Precision@6). We defined an answer as partially true if it contained the correct solution regardless of whether it also contradicted that solution, and strictly true if it did not contradict itself. All told we had 17 variations of our experimental conditions and 5 baselines; with both Precision@1 and Precision@6 testing, we examined a total of 32 conditions.

Techniques were compared using proportion tests. As an initial matter, our analysis demonstrated that legalBERT and GPT-J (strict truth test) significantly outperformed roBERTa and GPT-2 (both $p < 0.0001$). Furthermore, when directly comparing our own experimental methods (PAPR and ARPG) on randomized vs. non-randomized training sets, we found that using our hand-selected nonrandom order consistently trended better than using random order, but never significantly outperformed the random order (either overall or when looking at individual doctrines or case valences). To simplify reporting results (and to report only the higher-performing version of each method), we only report results on legalBERT and GPT-J baselines, and only on nonrandom training orders for our own techniques. The reported techniques are identified in Table 1. Table 2 shows results from each technique, presenting performance by raw score and percentage accuracy, overall and broken down by doctrine and case valence, and using Precision@1 and Precision@6. Head-to-head comparisons of techniques are presented in Tables 3 and 4, which present results using Precision@1 and Precision@6, respectively.

4.1 Precision@1 vs. Precision@6

Before comparing the methods' performance, we offer two more general observations regarding our evaluation. As noted above, we tested methods capable of generating multiple answers using both Precision@1 (taking the first answer as the system's output) and Precision@6 (having the system check its answer against the held-out truth and try again if it was wrong, up to six times) to separate a method's ability to generate the right answer from its ability to generate the right answer on the first try. Separating these is critical: because the MAC/FAC case retrieval system is not a core claim of our reasoning approaches, it is important to know whether the failure of an analogical reasoning system is attributable to the reasoning system itself, or to the system that retrieves the case for reasoning. In the real world precision@6 testing is impossible because it requires knowing the 'correct' outcome of a case, but using a historical dataset allowed our systems to check their own work (on the assumption that every case in the dataset was decided correctly, which is not necessarily true). As Table 2 demonstrates, evaluating using Precision@6 leads to improvement over using Precision@1 across the board, on both our own methods and on the GPT-J baselines. The difference in overall performance was significantly improved by testing with Precision@6 for every technique except ARPG with 0 additional CIs allowed.

But interestingly, it was not only these methods' performance relative to themselves that changed, but also performance relative to each other. GPT-J significantly underperformed nearly all other methods using Precision@1, even as it *outperformed* many methods using Precision@6. But more subtle differences appear. For example, when both were evaluated using Precision@1, ARPG with a strict truth test and 0 extra CIs significantly outperformed PAPR with a strict truth test; when evaluated using Precision@6, PAPR outperformed ARPG. These results suggest that, for our analogical reasoning methods, performance can be improved not only by improving the reasoning itself, but by finding ways to retrieve the right case the first time around.

4.2 Strict Truth vs. Partial Truth

As with Precision@6, the partial truth test and the tolerance for additional candidate inferences when using ARPG were designed

	Overall		Assault		Battery		Trespass		Positive		Negative	
Technique Name	P@1 # (%)	P@6 # (%)	P@1 # (%)	P@6 # (%)	P@1 # (%)	P@6 # (%)	P@1 # (%)	P@6 # (%)	P@1 # (%)	P@6 # (%)	P@1 # (%)	P@6 # (%)
RRLG	47 (47%)	-	8 (47%)	-	21 (53%)	-	18 (42%)	-	23 (33%)	-	24 (83%)	-
PAPR-NR-ST	17 (17%)	46 (46%)	3 (18%)	11 (65%)	9 (23%)	20 (50%)	5 (12%)	15 (35%)	14 (20%)	38 (54%)	3 (10%)	8 (28%)
PAPR-NR-PT	28 (28%)	72 (72%)	6 (35%)	16 (94%)	11 (28%)	27 (68%)	11 (26%)	29 (67%)	20 (28%)	53 (75%)	8 (28%)	19 (66%)
ARPG-NR-ST-0CIs	28 (28%)	35 (35%)	5 (29%)	6 (35%)	9 (23%)	10 (25%)	14 (33%)	19 (44%)	2 (3%)	7 (10%)	26 (90%)	28 (97%)
ARPG-NR-ST-1Cis	29 (29%)	47 (47%)	6 (35%)	9 (53%)	9 (23%)	16 (40%)	14 (33%)	22 (51%)	4 (6%)	19 (27%)	25 (86%)	28 (97%)
ARPG-NR-ST-2CIs	32 (32%)	58 (58%)	6 (35%)	9 (53%)	13 (33%)	22 (55%)	13 (30%)	27 (63%)	11 (16%)	30 (42%)	21 (72%)	28 (97%)
ARPG-NR-PT-0CIs	32 (32%)	43 (43%)	5 (29%)	7 (41%)	10 (25%)	12 (30%)	17 (40%)	24 (56%)	6 (9%)	15 (21%)	26 (90%)	28 (97%)
ARPG-NR-PT-1Cis	36 (36%)	58 (58%)	6 (35%)	10 (59%)	12 (30%)	20 (50%)	18 (42%)	28 (65%)	11 (16%)	30 (42%)	25 (86%)	28 (97%)
ARPG-NR-PT-2CIs	41 (41%)	71 (71%)	6 (35%)	10 (59%)	15 (38%)	26 (65%)	20 (47%)	35 (81%)	20 (28%)	43 (61%)	21 (72%)	28 (97%)
legalBERT	33 (33%)	-	9 (53%)	-	14 (36%)	-	10 (23%)	-	23 (33%)	-	10 (35%)	-
GPT-J-ST	12 (12%)	52 (52%)	1 (6%)	6 (35%)	3 (8%)	20 (50%)	8 (19%)	26 (60%)	11 (16%)	44 (62%)	1 (3%)	8 (28%)
GPT-J-PT	12 (12%)	61 (61%)	1 (6%)	6 (35%)	3 (8%)	23 (58%)	8 (19%)	32 (74%)	11 (16%)	52 (73%)	1 (3%)	9 (31%)

Table 2. Technique performance overall, by doctrine, and by valence; raw scores and percent accuracy. Compares PAPR, ARPG, & GPT-J with Precision@1 & Precision@6. RRLG and legalBERT only generate one answer so are only evaluated at Precision@1.

to examine whether the legal reasoning methods partially captured the relevant information in the cases about which they were reasoning. We therefore expected these relaxed standards to lead to improved performance, and they did. PAPR performed significantly better with a partial truth test than a strict one regardless of Precision@1 or @6, both overall and when broken down by doctrine. When using Precision@1 ARPG's performance improved when *both* the partial truth test was used *and* additional CIs were tolerated, but while each relaxed standard used alone trended better, those trends were mostly nonsignificant. However, when evaluating using Precision@6 it was more of a mixed bag: ARPG sometimes improved with a partial truth test alone, sometimes only by increasing the number of CIs tolerated, and sometimes both. Notably ARPG with 2 extra CIs and strict truth test outperformed ARPG with 0 CIs and a partial truth test. This suggests that the tolerance of additional CIs had more to do with the improved performance than the partial truth test, suggesting that focusing on stripping away extra facts that are still present in those generalizations is an important area of future work if ARPG is going to be a useful legal reasoning technique.

We also assessed partial truth in GPT-J, but it made no difference when testing with Precision@1, and only a non-significant trend of improvement when testing with Precision@6.

4.3 Comparing Methods

We first compare our reasoning techniques to each other, then to the baselines. (All measures reported as significant are at $p < 0.05$; detailed results available upon request). These comparisons are in Tables 3 and 4, which report results using Precision@1 and Precision@6 respectively, with cell values indicating which technique, if either, performed significantly better.

Our strictest measure for learning and reasoning using analogy is Reasoning with Positive Generalizations, using a strict truth test and allowing no extra candidate inferences in a mapping to conclude that a case is a positive example (ARPG-ST-0CIs). Indeed, this was the lowest-scoring of our experimental approaches, both when testing using Precision@1 and Precision@6. When testing using Precision@1 there was little improvement from allowing additional candidate inferences or using a partial truth test, but when using Precision@6, allowing one or two additional candidate inferences created a significant improvement over allowing 0 extra CIs. The improvement in performance from allowing one extra CI to allowing 2 was not significant, at $p=0.059$, although nearly as many additional cases were solved with each additional CI permitted.

Technique	GPT-J-PT P@1	GPT-J-ST P@1	legalBERT	ARPG PT 2CIs P@1	ARPG PT 1CIs P@1	ARPG PT 0CIs P@1	ARPG ST 2CIs P@1	ARPG ST 1CIs P@1	ARPG ST 0CIs P@1	PAPR PT P@1	PAPR ST P@1
RRLG	RRLG	RRLG	RRLG	---	---	RRLG	RRLG	RRLG	RRLG	RRLG	RRLG
PAPR-ST P@1	---	---	BERT	ARPG	ARPG	ARPG	ARPG	ARPG	ARPG	PT	
PAPR-PT P@1	PAPR	PAPR	---	ARPG	---	---	---	---	---		
ARPG-ST-0CIs P@1	ARPG	ARPG	---	PT 2CIs	---	---	---	---			
ARPG-ST-1CIs P@1	ARPG	ARPG	---	PT 2CIs	---	---	---				
ARPG-ST-2CIs P@1	ARPG	ARPG	---	---	---	---					
ARPG-PT-0CIs P@1	ARPG	ARPG	---	---	---						
ARPG-PT-1CIs P@1	ARPG	ARPG	---	---							
ARPG-PT-2CIs P@1	ARPG	ARPG	---								
legalBERT	BERT	BERT									
GPT-J-ST P@1	---										

Table 3. Performance of Reasoning Techniques Compared, Precision @1. Cell values indicate winning techniques.

When using Precision@6 and a strict truth test, PAPR outperformed ARPG with 0 extra CIs allowed. However, ARPG with 1 extra CI allowed performed on par with PAPR, and ARPG with 2 extra CIs allowed significantly outperformed PAPR. When using Precision@1 and a strict truth test, ARPG consistently outperformed PAPR. As to using a partial truth test, when using Precision@1 PAPR and ARPG performed substantially the same. However, when using a partial truth test and evaluating using Precision@6, PAPR outperformed nearly every other condition, including most ARPG conditions. From this we conclude that, without any help or second chances, ARPG is more able to identify and leverage legally relevant information than PAPR, but only under circumstances where both methods' performance leaves significant room for improvement. But the dramatic improvement of both approaches when using Precision@6 and partial truth tests demonstrates that these methods are capable of capturing legally relevant information, just not all of it and not on the first try.

There are two differences between PAPR and ARPG that potentially confound the source of the difference in their performance: that PAPR reasons with both negative cases and with outliers, whereas ARPG reasons with neither. PAPR heavily relied on these outliers: Of the 17 cases that PAPR correctly solved using a strict truth test and evaluated using Precision@1, 11 were solved by analogy to an ungeneralized outlier (9/14 positives, 2/3 negatives). Though not a majority, of the 72 cases that PAPR correctly solved using a partial truth test and evaluated using Precision@6, 30 were solved by analogy to an outlier (18/53 positives, 12/19 negatives). Furthermore, ARPG's superior performance was driven by its high performance on negative cases; PAPR outperformed it on positive cases. These results demonstrate that ungeneralized cases remain an important resource for reasoning directly by analogy. This may especially be true for negative cases, since ungeneralized examples were used in the majority of negative cases that PAPR solved.

RRLG significantly outperformed nearly every other condition when we evaluated our analogical methods and GPT-J using Precision@1, but the improved performance largely disappeared when using Precision@6, and RRLG was outperformed by methods using partial truth tests. When using Precision@6, RRLG performed significantly better than ARPG-ST-0CIs, and on par with PAPR-ST. RRLG consistently outperformed the analogy techniques on negative cases and was outperformed on positive cases. However, ARPG-PT-2CIs and PAPR-PT—techniques with relaxed standards for concluding cases were correct—both significantly outperformed RRLG.

We now turn to comparisons with baselines. When the system could only generate one answer (i.e., Precision@1 testing), RRLG beat all baselines, all ARPG conditions beat the GPT-J baselines, and PAPR beat the GPT-J baselines when using a partial truth test. When using a strict truth test, PAPR did not perform significantly differently from GPT-J, and was outperformed by BERT.

Using Precision@6, however, was again a mixed bag: GPT-J with a partial truth test outperformed most other conditions, although GPT-J with a strict truth test only beat the strictest ARPG condition; it was beaten by PAPR using a partial truth test and ARPG using a partial truth test and tolerating 2 additional CIs, and did not otherwise perform significantly differently from our methods. Additionally, all our methods except ARPG with 0 extra CIs tolerated (strict or partial truth test) significantly outperformed legalBERT. ARPG with 0 extra CIs outperformed legalBERT on

Technique	GPT-J-PT P@6	GPT-J-ST P@6	legalBERT	ARPG PT 2CIs P@6	ARPG PT 1CIs P@6	ARPG PT 0CIs P@6	ARPG ST 2CIs P@6	ARPG ST 1CIs P@6	ARPG ST 0CIs P@6	PAPR PT P@6	PAPR ST P@6
RRLG	GPT	---	RRLG	ARPG	---	---	---	---	RRLG	PAPR	---
PAPR-ST P@6	GPT	---	PAPR	ARPG	ARPG	---	ARPG	---	PAPR	PT	
PAPR-PT P@6	PAPR	PAPR	PAPR	---	PAPR	PAPR	PAPR	PAPR	PAPR		
ARPG-ST-0CIs P@6	GPT	GPT	---	PT 2CIs	PT 1CIs	---	2CIs	1 CIs			
ARPG-ST-1CIs P@6	GPT	---	ARPG	PT 2CIs	---	---	---				
ARPG-ST-2CIs P@6	---	---	ARPG	PT	---	ST 2CIs					
ARPG-PT-0CIs P@6	GPT	---	---	2CIs	1CIs						
ARPG-PT-1Cis P@6	---	---	ARPG	2CIs							
ARPG-PT-2CIs P@6	---	ARPG	ARPG								
legalBERT	GPT	GPT									
GPT-J-ST P@6	---										

Table 4. Performance of Reasoning Techniques Compared, Precision @6. Cell values indicate winning techniques.

Trespass cases and negative cases, while legalBERT did better on positive cases. However, loosening ARPG's standard of correctness led to significant improvement over legalBERT. GPT-J's substantially improved performance using Precision@6 led it to close the gap with RRLG when using a strict truth test, and to outperform RRLG when using a partial truth test.

The two baselines' relative performance changed when evaluating GPT-J using Precision@1 vs. @6. When using Precision@1, legalBERT beat GPT-J using strict and partial truth test. Using Precision@6, both GPT-J methods beat legalBERT.

Finally, comparing techniques' performance to themselves, ARPG and RRLG do better on negative than positive cases; PAPR does better on positive cases (though not when using a partial truth check), GPT-J does better on positive cases, and legalBERT does not significantly outperform itself on one type of case over another.

5 Discussion and Future Work

There are several things to note from these results. Most obviously, the improvement in our analogical reasoning techniques and in GPT-J when testing using Precision@6 demonstrates that these algorithms are currently more capable of generating correct answers than they are of generating them the first time around. This performance cannot be accounted for as the system simply exhausting all its possible mappings: when we ran our techniques with no depth limit (results not reported here), PAPR would regularly examine over a dozen mappings, and at times derived the correct answer after doing so. These methods' strong performance using Precision@6 testing therefore shows that good mappings are being retrieved and reasoned about in the first few cases, just not always as *the* first case. These results suggest to us that one of the most fruitful directions for future work in this area will come from focusing on the retrieval system, either by affecting what cases get returned first, or by investigating how retrieved cases can be validated before being used to reason about some case at bar.

The improvement in both PAPR and ARPG when using looser standards of correctness suggest these algorithms are capturing information about what governs and how to solve legal claims, but that the learned generalizations are noisy. Research into refining those generalizations will be useful. In particular, it may well be the case that the assumption described above—that the analogical generalization process alone will strip away irrelevant facts in the legal domain—is false. (We would not conclude that it was true or false based on these results alone.) If that hypothesis is false, that does not mean that ARPG's reasoning method is useless, but that the CASI schema-building process is not by itself sufficient to generate clean schemas, and the schemas will have to be modified by some other process to strip those additional irrelevant facts away. If ARPG is improved by improving generalizations, RRLG (which uses the same ones) should also be improved.

At nearly 50% accuracy, RRLG is already quite high-performing, and is our highest-performing method when testing using Precision@1. Using a system with Precision@1 is equivalent to using it when the true outcome of the case is unknown, so we expect RRLG to be the best-performing condition on truly new cases. One possible explanation for RRLG's improved performance when compared to analogy techniques using Precision@1 but not using Precision@6 is that RRLG, unlike analogy, actually *does* exhaustively search through all learned knowledge, by firing all the rules derived from its schemas and seeing if any of them work. Our analogical reasoning techniques must instead use the schemas one at a time, and so can be stymied if they pick the wrong one with which to reason first.

Our research techniques involve generating—not selecting—an answer. That means that, with the exception of legalBERT, there is no notion of “chance” performance for our systems, and the performance of each system must be evaluated relative to the others rather than in terms of its absolute performance. That our techniques outperformed legalBERT’s multiple-choice method is therefore promising. Though GPT-J performed commensurately with PAPR (strict truth) and ARPG with extra CIs permitted when using Precision@6, had we required GPT-J to generate only *true* statements or to label events as our methods did, then it only gets a single question correct using Precision@1 and only eleven correct using Precision@6. Also, since most training cases are positive, with the defendant as the accused, it is generally a good guess when using this dataset that the defendant behaved tortiously towards the plaintiff. This may explain why GPT-J performed better on positive cases. It also might explain why our techniques generally outperformed GPT-J on negative cases; another explanation is that ARPG’s and RRLG’s performance on negative cases may be inflated, because they correctly solve negative cases by failing to derive positive conclusions, and they might perform worse on negative cases as they do better on positive ones.

Unlike statistical methods, our techniques can be inspected to understand why an answer was generated. Though recent work on statistical methods promises greater explainability [22], there is still no substitute for examining a system’s internals. One can inspect the analogy to determine what entities and expressions were placed into alignment, to determine whether the system stumbled into a correct answer through blind luck or derived it by properly placing the case at bar into alignment with the prior case. Inspectability and explainability are not useful only for research debugging, but because a legal reasoning system that can explain itself should be more trustworthy and thus more useful than one that cannot.

Which of our three algorithms is most appropriate for a given task depends on the user’s needs. The analogical reasoning approaches are more flexible than the rule-reasoning system because they do not require strict unification, but this flexibility could be seen as a liability in domains where logical correctness is prized. ARPG has the advantage over PAPR of stripping away facts that are known to be incidental to cases in favor of the facts shared across cases, but cannot reason directly about negative precedents.

Several areas of future work have already been mentioned: one is to generate cleaner generalizations; another concerns improving retrieval to ensure that the best case is retrieved initially. New large-language-models are constantly being released, and it would be instructive to evaluate the performance of, for example, ChatGPT on our dataset instead of just GPT-J.

Our methods also currently reason in a single step, but legal reasoning can involve many steps, for example when addressing affirmative defenses or responding to arguments. Our dataset includes self-defense cases, and we wish to extend our methods to handle such cases. We are also working to adapt our methods to generate arguments. And we wish to investigate performance when testing cases against true precedents, i.e., only temporally prior cases. Finally, we want to leverage the ontology and knowledge base to rerepresent cases and reason about novel situations, to allow us bring the full power of analogical reasoning to bear on the legal domain as discussed in [23].

ACKNOWLEDGMENTS

This research was supported by the Computational Cognition and Machine Intelligence Program of the Air Force Office of Scientific Research under award #FA9550-20-1-0091.

REFERENCES

[1] Weinreb, L. (2016). Legal Reason: the Use of Analogy in Legal Argument, 2d ed. Cambridge University Press
[2] Sunstein, C.R. (1996). Legal Reasoning and Political Conflict. Oxford University Press.
[3] Levi, E.H. (1949). An Intro. to Legal Reasoning. Univ. Chicago Press.
[4] Posner, R. (2006). Reasoning by Analogy (Review of [1]). *Cornell Law Review* 91, 761–774.
[5] Alexander, L. & Sherwin, E. (2008). Demystifying Legal Reasoning. Cambridge University Press.
[6] Blass, J.A., & Forbus, K.D. (2022). The Illinois Intentional Tort Qualitative Dataset. In *Proc’s of the 35th JURIX Int’l Conference on Legal Knowledge and Information Systems*. Saarbrucken, Germany.
[7] Branting, L.K. (1991). Building Explanations from Rules & Structured Cases. *International Journal of Man-Machine Studies*, 797
[8] Gentner, D., (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2), p155.
[9] Ashley, K. D. (1989). Toward a comprehensive theory of arguing with precedents. In *Proc’s of the 2d Int’l Conference on AI and Law*, 93.
[10] Ashley, K. (1999). Designing Electronic Casebooks that Talk Back: The CATO Program. *Jurimetrics* 40, p275
[11] Rissland, E., Skalak, D. & Friedman, M. (1996). BankXX: Supporting Legal Arguments through Heuristic Retrieval. *AI & Law*.
[12] Gray, M., Savelka, J., Oliver, W., & Ashley, K. (2022). Toward Automatically Identifying Legally Relevant Facts. *JURIX 2022*.
[13] Horty, J.F. (2011). Reasons and Precedent. In *Proceedings of the 13th International Conference on AI and Law*, Pittsburgh, PA.
[14] Verheij, B. (2017). Formalizing Arg’ts, Rules and Cases. *ICAIL 2017*.
[15] Van Woerkom, W., Grossi, D., Prakken, H., & Verheij, B. (2022). Landmarks in Case-Based Reas’g: From Theory to Data. *HHAI 2022*.
[16] Gentner, D., Loewenstein, J., Thompson, L., & Forbus, K. (2009). Reviving Inert Knowledge: Analogical Abstraction Supports Relational Retrieval of Past Events. *Cognitive Science* 33, p.1343.
[17] Forbus, K. D., Ferguson, R. W., Lovett, A., & Gentner, D. (2017). Extending SME to handle large-scale cog’ve model’g. *Cog.Sci. 41*(5).
[18] Lenat, D. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of ACM*, *38*(11) p.33.
[19] McLure, M.D., Friedman, S.E., & Forbus, K.D. (2015). Extending Analogical Generalization with Near-Misses. In *Proceedings of the 29th AAAI Conf. on Artificial Intelligence*, Austin, TX.
[20] Blass, J.A., & Forbus, K.D. (2022). Conclusion-verified Analogical Schema Induction. In *Proc’s of the 2022 Advances in Cog’ve Systems.*
[21] Tomai, E., & Forbus, K. (2009). EA NLU: Practical Language Understanding for Cog. Modeling. In *Proc’s of 2009 FLAIRS Conf.*
[22] Branting, K., Weiss, B., Brown, B., Pfeifer, C., Chakraborty, A., Ferro, L., & Yeh, A. (2019). Semi-supervised methods for explainable legal prediction. *ICAIL 2019,*, 22.
[23] Atkinson, K. & Bench-Capon, T. (2019) Reasoning with legal cases: Analogy or rule application? In *Procs of the 17th International Conference on AI and Law.*