The Richmond Journal of Law and Technology

Volume IX, Issue 2,
Winter 2002-2003

Smoke and Mirrors or Science? Teaching Law with Computers - A Reply to Cass Sunstein on Artificial Intelligence and Legal Science

By Eric Engle*

The application of computer science in the law has largely, and productively, centered on educational programs[1] and programs generating and managing databases and data management. Some limited work, however, has been done in the use of artificial intelligence ("AI") to present models[2] of legal decision-making.[3] The majority of the work involving AI in the law, as the majority of work in AI generally, has focused on developing expert systems.[4] An expert system attempts to solve one problem, or one class of problems well and should be distinguished from general systems, which seek to solve any problem. While databases and didactic applications involve the most productive work being done, AI presents more complex and more interesting problems having the potential to further increase productivity. Therefore, while AI may still be in its infancy (as was all of computer science until twenty years ago), its intellectual vistas are potentially limitless.

It was thus with mixed reactions that I read a recent article by Cass Sunstein, arguing that AI in law is of very limited utility and, at present, can only solve very limited classes of legal problems. Sunstein likens computer AI in law at present to a glorified LEXIS.[5] This is not the case because AI does more than automatically sort data. Sunstein argues that computers cannot reason analogically because a computer cannot make an evaluative[6] (moral) choice[7] and thus

---

[1] *See, e.g.,* Dan Hunter, *Teaching Artificial Intelligence to Law Students*, 3 LAW TECH. J. 3 (Oct. 1994), *available at* http://www.law.warwick.ac.uk/ltj/3-3h.html (discussing the methodological problems involved, especially the problems of developing sylabi for teaching law and AI).

[2] *See* Alan L. Tyree, *Fred Keller Studies Intellectual Property*, *at* http://www.law.usyd.edu.au/~alant/alta92-1.html (last modified Dec. 20, 1997) (discussing self-paced instructional programs).

[3] *See, e.g.,* Alan L. Tyree, *FINDER: An Expert System*, *at* http://www.law.usyd.edu.au/~alant/aulsa85.html (last modified Dec. 20, 1997).

[4] *See, e.g.,* Carol E. Brown and Daniel E. O'Leary, *Introduction to Artificial Intelligence and Expert Systems*, *at* http://accounting.rutgers.edu/raw/aies/www.bus.orst.edu/faculty/brownc/es_tutor/es_tutor.htm (last modified 1994).

[5] "My conclusion is that artificial intelligence is, in the domain of legal reasoning, a kind of upscale LEXIS or WESTLAW – bearing, perhaps, the same relationship to these services as LEXIS and WESTLAW have to Shepherd's." CASS R. SUNSTEIN, OF ARTIFICIAL INTELLIGENCE AND LEGAL REASONING 7 (Chicago Public Law and Legal Theory Working Paper No. 18, 2001), *available at* http://www.law.uchicago.edu/academics/publiclaw/resources/18.crs.computers.pdf.

[6] "I have emphasized that those who cannot make evaluative arguments cannot engage in analogical reasoning as it occurs in law. Computer programs do not yet reason analogically." *Id.* at 9.

cannot infer new rules of production from old rules of production. Sunstein argues that because computers cannot reason analogically, they cannot develop AI,[8] at least not yet.[9] As we shall see, this position is simply untenable.[10]

There is some cause to focus, as Sunstein appears to,[11] on expert systems rather than general systems: the problems facing the elaboration of an expert system are much more manageable than those facing the development of a general system. However, AI programs are not simply split between expert systems and general systems but also between "static" systems with fixed (pre-programmed) rules of production and "dynamic" systems which develop their own rules of production based upon experience.[12]

---

[7] "I believe that the strong version [of claims for machine-based AI] is wrong, because it misses a central point about analogical reasoning: its inevitably evaluative, value-driven character.*" Id.* at 5.

[8] "Can computers, or artificial intelligence, reason by analogy? This essay urges that they cannot, because they are unable to engage in the crucial task of identifying the normative principle that links or separates cases." *Id.* at 2. Sunstein appears to be grappling with the so-called law of Hume (the idea that moral choice cannot be deduced, but is instead arbitrary), yet does not address Hume. E.g., "[S]ince HYPO can only retrieve cases, and identify similarities and differences, HYPO cannot really reason analogically. The reason is that HYPO has no special expertise is [sic] making good evaluative judgments. Indeed, there is no reason to think that HYPO can make evaluative judgments at all." *Id.* at 6.

[9] "[W]e cannot exclude the possibility that eventually, computer programs will be able both to generate competing principles for analogical reasoning and to give grounds for thinking that one or another principle is best." *Id.* at 8.

[10] "A trained [neural] network is capable of producing correct responses for the input data it has already 'seen', and is also capable of 'generalisation', namely the ability to guess correctly the output for inputs it has never seen." Farrukh Alavi, *A Survey of Neural Networks: Part I*, 2 IOPWE CIRCULAR (Int'l Org. of Pakistani Women Eng'rs), (July 1997), *available at* http://www.iopwe.org/JUL97/neural1.html.

[11] "What I am going to urge here is that there is a weak and strong version of the claims for artificial intelligence in legal reasoning; that we should accept the weak version; and that we should reject the strong version, because it is based on an inadequate account of what legal reasoning is." SUNSTEIN, *supra* note 5, at 4. While it is true that we can look at the fact that general systems are much more difficult to construct than expert systems to support this statement, Sunstein does not. Nor does Sunstein support this point by noting the difference between "dynamic" AI, which truly learns and generates new rules of production based on iterative experience, from "static" AI, which merely represents pre-programmed rules of production. Rather Sunstein supports this point, which is for the two aforementioned reasons tenable, with a third point, which is not; Sunstein relies on a dichotomy of deductive reasoning and analogical reasoning to explain the limitations of AI at present. In fact, the problem Sunstein has identified is either that of Mill (that inductive ampliative arguments are not necessary, only probable, as opposed to deductive arguments, which are either necessary or invalid) or that of Hume. Hume considers the logic and science amoral and implies that moral choice is arbitrary. In fact, this author disagrees with the popular representation of Hume, but that point is outside the scope of this paper.

[12] Two major avenues of research have emerged over the last two decades . . . artificial intelligence (AI) and artificial neural networks (ANNs). While there are some similarities in both the origins and scope of both of these disciplines, there are also fundamental differences, particularly in the level of human intervention required for a working system: AI requires that all the relevant information be pre-programmed into a database, whereas ANNs can learn any required data autonomously. Consequently, AI expert systems are today used in applications where the

Static AI programs, such as the one I present here, use invariable pre-programmed rules of production. They do not learn because their rules of production are "fixed." For example, though chess programs are the most often-cited example of successful AI, they are, in fact, an example of "static" AI (albeit highly-developed). This is because most chess programs do not adapt their algorithms to take advantage of the weaknesses of their opponent, but rather use a pre-programmed algorithm.

On the other hand, "dynamic" computer programs generate their own rules of production and are consequently "trainable."[13] Programs which "learn" by dynamically developing rules of production do exist[14] and truly "learn." Such programs must derive principles (for example, the shape of a letter) from cases (a given set of experiences that are either provided by the user or are pre-programmed) – which is the very type of reasoning that Sunstein argues a computer cannot do.[15]

The most well-known work in AI that actively "learns" is in the field of neural networks.[16] Neural networks are best known for being used to recognize characters. This field of AI is well-developed and, therefore, no longer in its infancy. Programs such as Apple's Inkwell do indeed "learn" to recognize characters and words depending on input and correction by the user.[17]

Rather than focusing on the split between "active" programs, which "learn," and "static" programs, which do not, Sunstein focuses on a split between analogical reasoning (inductive

---

> underlying knowledge base does not significantly change with time (e.g. medical diagnostic systems), and ANNs are more suitable when the input dataset can evolve with time (e.g. real-time control systems).

Alavi, *supra* note 10.

[13] "In 1986, Rumelhart, Hinton and Williams published the 'back-propagation' algorithm, which showed that it was possible to train a multi-layer neural architecture using a simple iterative procedure." *Id*.

[14] *See id.*

[15] *See generally* SUNSTEIN, *supra* note 5.

[16] *See, e.g.,* Dan Hunter, *Commercialising Legal Neural Networks*, 2 J. INFO. LAW & TECH. (May 7, 1996), *at* http://elj.warwick.ac.uk/jilt/artifint/2hunter/.

[17] *See* Alavi, *supra* note 10.

logic) and deductive reasoning. Deductive reasoning is necessarily true but develops no new propositions. Inductive reasoning is not necessarily true but develops new propositions. Sunstein argues that computer programs, or at least computer programs in law, cannot learn to properly apply analogical reasoning[18] because he argues that they cannot derive principles from cases - which, as this article will show, is false.[19] Algorithms for deriving new principles from existing cases do exist[20] and usually invoke iterative and/or pseudo-random functions or abduction to do so. As Aikenhead notes, "[n]umerous systems have been constructed that simulate aspects of legal analogizing"[21] including SHYSTER, a case-based legal expert system.[22] Aikenhead's arguments are much more refined and computationally accurate[23] than Sunstein's.

---

[18] *See* SUNSTEIN, *supra* quotation accompanying note 6.

[19] *See* Michael Aikenhead, *A Discourse on Law and Artificial Intelligence*, 5 LAW TECH. J. 1 (June 1996), *available at* http://www.law.warwick.ac.uk/ltj/5-1c.html) (discussing the different theories about modeling law using AI in the law, including the use of models for analogical reasoning).

[20]
> If a knowledge base does not have all of the necessary clauses for reasoning, ordinary hypothetical reasoning systems are unable to explain observations. In this case, it is necessary to explain such observations by abductive reasoning, supplemental reasoning, or approximate reasoning. In fact, it is somewhat difficult to find clauses to explain an observation without hints being given.
>
> Therefore, I use an abductive strategy (CMS) to find missing clauses, and to generate plausible hypotheses to explain an observation from these clauses while referring to other clauses in the knowledge base. In this page, I show two types of inferences and combines [sic] them. One is a type of approximate inference that explains an observation using clauses that are analogous to abduced clauses without which the inference would fail. The other is a type of exact inference that explains an observation by generating clauses that are analogous to clauses in the knowledge base.

Akinori Abe, *Abductive Analogical Reasoning*, *at* http://www.kecl.ntt.co.jp/icl/about/ave/aar.html (last visited Nov. 25, 2002).

[21] Michael Aikenhead, *Legal Principles and Analogical Reasoning*, PROC. SIXTH INT'L CONF. ARTIFICIAL INTELLIGENCE & LAW: POSTER PROC., 1-10 (extended abstract available at http://www.dur.ac.uk/~dla0www/centre/ic6_exab.html); *Simulation in Legal Education* (with Widdison R.C. and Allen T.F.W.), 5 INT'L J.L. & INFO. TECH. 279-307.

[22] James Popple, *SHYSTER* (SHYSTER is a C source code available under GPL), *at* http://cs.anu.edu.au/software/shyster/ (last modified Apr. 30, 1995).

[23] Aikenhead acknowledges limitations in existing models but does not categorically dismiss them and, unlike Sunstein, presents critiques which will lead future work in this field. For example, Aikenhead writes:
> In this context the CBR system, GREBE is particularly interesting. Branting's system addresses a major problem that arises in case based reasoning; of determining when two cases are sufficiently similar to allow analogical reasoning to occur. For two cases to be similar, their constituent elements must be sufficiently similar. How is the existence of this similarity determined? Branting's innovation is to allow GREBE to recursively use precedents to generate arguments as to why elements of cases are similar. Once the elements of cases can be said to be similar, the cases themselves can be regarded as similar. The system thus generates an argument as to why cases are similar by recursively generating an argument as to why constituent elements of cases are similar.

Programmatic representations of inductive inference, like most computational problems, are, in fact, trivial[24] (in the computational sense of the word). For example, presume that we have three known cases, each having three attributes with the following values:

Case	Values
I	1,2,3
II	A,B,C
III	A,2,3

Given a Case IV, with values 1,2,C, analogical reasoning would compare the values and determine that Case IV is most similar to Case I and apply the same rule in Case IV.

Inductive amplification does not merely apply existing rules by analogy to a new case. Rather, it derives a new rule from existing cases. It is this type of reasoning which Sunstein argues a computer is incapable of doing. Using the same example, the derivation of a new rule would be that any new case will be decided according to the rule used in the case with the greatest number of similar elements. Such a rule could be initially programmed into the computer as a "static" rule of production, or could be "learned" through trial and error as a dynamic rule of production.

Let us consider the example of a node in a neural network with three inputs that is suddenly given a new fourth input. The new input must be tested and controlled to determine whether the new input is to favor or disfavor a certain outcome and to determine what weight should be applied to the new input. That control can either be pre-programmed or, more

---

> While GREBE only operates in the domain of CBR and is not a complete argumentation system it does illustrate how useful argumentation processes can be self-referential. What is needed is an extension of this approach; applying the full theory of legal discourse. In such an approach, arguments would recursively be generated as to how and why rules or cases should apply.

Aikenhead, *supra* note 19.

[24] Paul Brna, *Imitating Analogical Reasoning*, *at* http://www.comp.lancs.ac.uk/computing/research/aai-aied/people/paulb/old243prolog/section3_9.html (Jan. 9, 1996)(using Prolog to illustrate such algorithms).

efficiently, be supplied by the user. Statically, the program would be given at least two rules of production. Say that new factors shall have, for example, 1/n weight (where n equals the total number of factors to be considered) and that factor n (the new factor) should favor the party with no burden of proof unless instructed otherwise. To these rules of production could be added a third human "oversight" function: by running, for example, three trials where the human oversees the outcome, verifying or denying the computer response, and programmatically applies the rule of production that the human taught the computer, the computer would be able to correctly apply the new rule of production and apply it simulating analogical reasoning. Such a control function could be pre-programmed, however, by iterating[25] through all combinations, first favoring, then disfavoring, the outcome and using different weights and comparing the result to existing cases.

If Sunstein's position is, so far as programming computers goes, untenable, what about its validity in law? There are valid grounds for questioning Sunstein's assumptions about law. Legal scholars should recognize that, although the common law relies heavily on analogical reasoning, the civil law relies equally heavily on deductive reasoning.[26] Sunstein would have us ignore this.[27] Since Sunstein does not appear to question the ability of a program to represent deductive reasoning, it seems his claim that it is impossible, at present, for a program to represent the law is ill-considered.

Sunstein hedges his position. First, he argues that his position that an analogical inference engine is impossible cannot, in fact, be verified,[28] which takes his opinion out of the realm of

---

[25] *See* Alavi, *supra* note 10.

[26] Wim Voermans et al., *Introducing: AI and Law*, Inst. for Language Tech. and Artificial Intelligence, *at* http://cwis.kub.nl/~fdl/research/ti/docs/think/3-2/intro.htm (last visited Nov. 25, 2002).

[27] "What is legal reasoning? Let us agree that it is often analogical." SUNSTEIN, *supra* note 5, at 5. In civil law jurisdictions legal reasoning is more often deductive than analogical. Therefore, there is good reason to reject Sunstein's assumption about legal reasoning.

[28] "We should reject the strong version [of legal reasoning via AI] because artificial intelligence is, in principle, incapable of doing what the strong version requires (there is no way to answer that question, in principle). . . ." SUNSTEIN, *supra* note 5, at 4. This directly contradicts a later hedge of Sunstein's when he argues that future technology may make computer AI possible. Either it is impossible and/or unverifiable or it is possible either with existing or future technology. In fact, while the speed of microprocessors and the amount of data which can be stored

science. Second, Sunstein argues that "things might change."[29] These hedges appear to be contradictory. Sunstein has said that his claim cannot be verified or falsified, yet implies that new technology may one day permit machines to model analogical reasoning.[30] Even if we could grant both these (contradictory) hedges, Sunstein's argument goes too far and can be demonstrated to be erroneous using a trivial example of analogical reasoning.

Let us presume that two Cases with three elements each will be similarly decided if two of the elements are the same (that is, if the two cases are analogically similar). Programmatically, analogical reasoning could be represented thus:

```
//Assign values to our known case
Case1.Argument1 = 1;
Case1.Argument2 = 1;
Case1.Argument3 = 1;
Case1.Outcome = "Not Guilty";
//Initialize our unknown case
Case2.Outcome = "Unknown";
//Trivial example of analogical reasoning
if (case2.Argument1 = case1.Argument1)
{
        if (Case2.Argument2 == case1.Argument2)
        {
        Case2.Outcome == Case1.Outcome;
        }
        if (Case2.Argument3 == case1.Argument3)
        {
        Case2.Outcome == Case1.Outcome;
        }
}

if (Case2.Argument2 == Case1.Argument2)
{
        if (Case2.Argument3 == Case1.Argument3)
        {
        Case2.Outcome == Case1.Outcome;
        }
}
```

---

has changed radically in the last twenty years, processor architecutre (a bus and registers) remains basically the same as that of forty years ago (just with larger busses and more registers). Thus the technological change Sunstein awaits would, and can, occur at the software level and need not occur at the hardware level.

[29] *Id.* at 2.

[30] *See* SUNSTEIN, *supra* quotation accompanying note 9.

Alternatively, this could be represented using weighted balances rather than binary values, for example, presuming the plaintiff has the burden of proof:

```
If (plaintiff.arguments[weight] > defendant.arguments[weight])
        {
        decisionForPlaintiff();
        }
```

These reasons give us cause to question Sunstein, both as programmer and legal theorist. Sunstein would have done better to argue that computation of law is impossible because building and training a neural network to operate as a general system is too complex. Such an argument might hold water. Complexity in programming a general system of AI can be found in the fact that creation of a general system would first require a natural language parser. However, such parsers have existed for at least twenty years, and are constantly improving. Presuming that an adequate parser could be found or built, the next difficulty would be developing self-learning neural networks. This is difficult, but certainly not impossible. Such a neural network should include a human "control" to test and correct the engine's inferences, as that would generate a better engine more quickly than one based purely on pre-programmed rules of production. The next difficulty would be to generate a sufficiently large field of cases to teach the engine. Again, this is not an insurmountable task. Though this task would, like adapting a natural language parser, be time-consuming. A final difficulty might be the computational speed or memory required. This is the least of the problems, as enormous expert systems are well within the computational power of contemporary desktop computers. Although general systems in AI are

the exception, they are not computationally impossible and would not require some new breakthrough in microprocessor architecture or memory[31] as Sunstein argues.

This contentious introduction is intended to set the stage for my very unambitious static inference engine. I have written a program to assist in the teaching (and perhaps practice) of tort law. This program (unlike my next) is well within the hedges that Sunstein places on his bets. It does not use a neural network, and does not "learn" a new system of rules (but it could have by relying on either a neural-node model or using a tree model with either head or tail recursion). Instead, the program provided is "static." I have given a set of rules of production (which represent the basics of tort law) and then the program asks the user a series of question to determine whether a tort has taken place and giving reasons for its decision. The program essentially uses a series of "binary" tests as rules of production.

Another approach to static AI modeling of law (which I used in a program to determine the tax residence of a person according to the French-U.S. tax treaty[32]) uses a weighted "balancing" approach. One advantage to the binary method is that when the law can be represented as "either/or," it is highly methodologically defensable as a model of law. In contrast, the law only rarely quantifies the weight given to its nodes when balancing competing interests. If there were any computational problems in generating AI models of law, it is, in my opinion, here: the quantification of factors that may be quantifiable but generally are not quantified by courts. Even this difficulty is not insurmountable. The law sometimes uses quantifiable economic data to calculate the weight of factors in legal balancing tests. So, using a "balancing" test is computationally and legally defensible.

---

[31] Such a breakthrough, while not necessary (neural networks can be programmed using existing CPUs), is possible. "As an information-processing system, the brain is to be viewed as an asynchronous massively-parallel distributed computing structure, quite distinct from the synchronous sequential von Neumann model that is the basis for most of today's CPUs." Alavi, *supra* note 10.

[32] Eric Engle, « La convention franco américaine et la modélisation du droit fiscal par l'informatique », 131 Fiscalite Europeenne - Droit International Des Affaires (2003).

The program is basically self-explanatory and includes a brief essay on tort law, which, along with the source code, is reproduced below. The program was written in xTalk, a derivative of Pascal, using the MetaCard engine. I chose xTalk rather than C or a C-derived language because it is "pseudo" English and because the auto-formatting of the metaTalk editor makes the source code readable even for non-programmers. Given the current computational speed and power of CPUs, I see no real advantage to developing AI using a lower-level language such as C or assembler. However, the fact that there are plenty of C AI libraries explains why programmers may wish to port them to Pascal or xTalk, so that non-programmers can correctly assess the utility of AI in representing the law for purposes of teaching, research, and even legal practice. A great deal of useful work can be done using computers to model law either in education or data management or in legal practice including, eventually, general AI systems. Whether the position that Professor Sunstein takes is defensible will be revealed in time.