Marvin Minsky

Commonsense-Based Interfaces

To build a machine that truly learns by itself will require a commonsense knowledge representing the kinds of things even a small child already knows.

To make our computers easier to use, we must make them more sensitive to our needs. That is, make them understand what we mean when we try to tell them what we want. But how can computers accomplish such things when philosophers struggle endlessly to understand what "meaning" means? My answer is that those efforts failed because meaning is not a single thing. Instead, the activities of human thought engage an enormous society of different structures and processes.

The secret of what *X* means to us lies in how our representations of *X* connects to the other things we know. If you understand something in only one way then you scarcely understand it at all because when something goes wrong, you'll have nowhere to go. But if you use several representations, each integrated with its set of related pieces of knowledge, then when one of them fails you can switch to another. You can turn ideas around in your mind to examine them from different perspectives until you find one that works for you. And that's what we mean by thinking!

If we want our computers to understand us, we'll need to equip them with adequate knowledge. Only then can they become truly concerned with our human affairs. To help us work, they must know what our jobs are. To entertain us they'll need to know what their audiences like or need. Educational systems should understand what their students already know as well as their present misconceptions.

JEAN-FRANCOIS PODEVIN

How do we make a computer program that understands such commonsense things? That's almost the same as asking how to make a machine that can really think. Why has that problem seemed so hard? Partly because a typical program has only one way to deal with a problem, so if something goes wrong, it gets totally stuck. Contrastly, a person will search for a different approach or cunningly change the subject. The trouble with computers today is they're always starting from scratch. To make them more worth dealing with, we'll have to aim toward supplying them with great libraries of commonsense knowledge like the ones inside our children's heads.

For making computers more pleasant to use, we've been seeing a lot of development aimed at trying to make computers react in more natural, friendly, or lifelike ways. An example of this was Microsoft's "Bob"—an animated avatar so intrusive and so virtually useless that most of its users abandoned it. Such approaches are doomed because their users soon see through those bogus illusions of friendliness. Despite the use of emotional tricks to jolly up the interface, the users soon find the systems beneath them and simply too limited to tolerate.

Why don't computers have common sense? Why can't they learn from experience? Some of our earliest programs could solve difficult problems in specialized subjects, yet still no programs today can do most of the things young children can do. Why do our current programs behave in such limited and inflexible ways? Some people regard this as obvious:

Vitalist: Computers do only what they're programmed to do. People have programmed computers to speak, but those machines will never know what those words mean. Meaning is an intuitive thing, it can't be reduced to zeros and ones. Computers can only do logical things, and meanings are not always logical.

Humanist: It's because machines have no hopes or goals, nor any fears or phobias. Not even knowing they exist, they can't have a sense of accomplishment."

Theologist: It's because a machine has no spirit or soul—the essence that only a brain can own.

Instead of blaming them for their faults, we should make our machines more resourceful by furnishing them with more common sense.

Logician: Before you talk more about common sense, you really ought to define it more clearly. Good arguments should start by stating precisely what they are about lest you build on a shaky foundation.

That policy might seem logical, but it's wrong when it comes to psychology. Of course, we don't like to be imprecise, but definitions can make things worse when you don't yet know what you're talking about. It doesn't make sense to limit yourself before you begin to experiment. So instead, we'll take a different approach: We'll try to design (as opposed to define) machines that can do the things we want.

Examples of Commonsense Thinking

Whenever we speak about "commonsense thought," we're referring to things that most people can do, often not even knowing they're doing them. Thus, when you hear a sentence like: "Fred told the waiter he wanted some chips," you will infer all sorts of things. Here are just a few of these, condensed from [2, 3].

- The word "he" means Fred. That is, it's Fred who wants the chips, not the waiter.
- This event took place in a restaurant. Fred was a customer dining there at that time. Fred and the waiter were a few feet apart at the time. The waiter was at work there, waiting on Fred at that time. Fred wants potato chips, not wood chips, cow chips, or bone chips. There's no particular set of chips he wants.
- Fred wants and expects the waiter to bring him a single portion (1–5 ounces, 5–25 chips) in the next few minutes. Fred will start eating the chips very shortly after he gets them.
- Fred accomplishes this by speaking words to the waiter. Fred and the waiter speak the same language. Fred and the waiter are both human

IF WE WANT OUR COMPUTERS TO UNDERSTAND US, WE'LL NEED TO EQUIP THEM WITH ADEQUATE KNOWLEDGE. ONLY THEN CAN THEY BECOME TRULY CONCERNED WITH OUR HUMAN AFFAIRS.

beings. Fred is old enough to talk (2+ years of age). The waiter is old enough to work (4+ years, probably 15+). This event took place after the date of invention of potato chips (in 1853).
- Fred assumes the waiter also infers all those things.

Each child learns to use thousands of words, but no computer knows what those words mean, so no computer can yet understand even a casual conversation. For example, if you were to mention a "string," any child would know what you mean, because of knowing dozens of things that anyone can do with a string, for example, to tie up a package or fly a balloon. The child would also know things like these:

- A string can be used to pull an object, but not to push it.
- It isn't good to eat a string, and you cannot make a balloon with string.
- Before you put a string in a box, you first must be sure to open that box.
- If you steal someone's string, its owner will be annoyed.

How large are our networks of commonsense knowledge? First let's consider our knowledge of language. A child might know 10,000 words each linked in various different ways to hundreds of other knowledge structures. Some are descriptions, while others are processes. Each such link will, in turn, lead to other links—they're all in a semantic network—so that whenever some process gets stuck, you can usually find some alternative.

How large are our mental networks of knowledge? Perhaps a thousand of our most common words have links to thousands of other concepts. That adds up to millions of units of knowledge. But language is only a single one of our large-scale abilities. For each expert skill a person possesses, there must be a similar order of structure for vision, for hearing, for haptic perception, for all sorts of physical manipulations, and for various sorts of social knowledge. How many millions of units of knowledge does a normal person possess? We sometimes hear legends of people who have photographic memories, and several old experiments appeared to reveal such abilities. But researchers can never replicate these, so I suspect they were flawed experiments. There is no replicable demonstration in which a person can later reproduce more than the order of about 1 bit from every two seconds of a prolonged learning interval. If we can learn only a few bits per second—that's just a few megabytes per year—this suggests a rather modest bound on the extents of our memories.

Some Constituents of Commonsense Reasoning

Everyday commonsense thinking involves a huge collection of hard-earned ideas. This includes masses of factual knowledge about the problems we're trying to solve. But we also must learn effective ways to retrieve and apply the relevant knowledge. Many processes must be engaged when we imagine, plan, predict, and decide using multitudes of exceptions and rules. Doing this requires *knowledge about how to think*—how to organize and control those processes—and this must engage such resources as:

Using different representations to describe the same situation. We need ways to convert new experiences into memory structures that represent them. Artificial intelligence researchers have developed many different such representations. Each is useful in certain domains but none of them works well in every domain. Some of these include property lists or polynemes, frames and transframes, frame-arrays, database query languages, explanation-based reasoning, haptic representations, K-trees and level-bands, logic programming, rule-based systems, micronemes, natural language expressions, semantic networks, scripts and stories, and object-based programming.

The most usual way to represent knowledge has been to first select a representation. And that's been the problem! Using any particular representation you'll soon to encounter some limitation or constraint, and these will quickly accumulate until your reasoning starts to fail. It seems to me that we usually need to use several different representations for each fragment of commonsense knowledge about each thing, idea, or relationship. We are also able to swiftly switch between those different methods and representations.[1] To know when to do this, one needs knowledge about which methods and representations are good for solving various kinds of problems. And that, in turn, means we need good ways to characterize "kinds of problems."

Negative expertise. One also needs ways to recognize when each of one's methods is starting to fail. If you recognize the particular way things went wrong, you can use that as a clue for deciding what you should do next. Knowing how each method is likely to fail can be used at a higher, reflective level, for example, by a B-brain that can use such knowledge

[1] I proposed a theory of how we do this in [6], and I'll extend that theory in [9].

to control a mental activity. This could be where we exploit the sort of debugging knowledge described in [11]: "Achieving my first goal interfered with achieving my second goal; I'll reverse their order and try again."

Knowledge retrieval. Retrieving relevant information from a person's huge networks of commonsense knowledge requires ways to recognize which remembered problems or situations most resemble the context of what that person is trying to do. This means that your systems need ways to describe what you're trying to do, and then to reason about those descriptions. I contend this must be largely based on the skillful use of analogies.

Self-reflection. Finally, our machines must keep records that describe the acts and the thinking they've recently done so they'll be able to reflect on the results of what they tried to do. This is surely an important part of what people attribute to consciousness.

In the early years of AI research, we saw programs prove theorems and win games of chess. Why was it simpler to make expert systems than to simulate commonsense patterns of thought? The answer is that each of those specialized experts used procedures that worked inside small, tidy worlds, whereas commonsense thinking often engages a much wider range of knowledge and skills. Few youngsters can design transformers or diagnose renal ailments, but whenever those children speak or play, they use thousands of different kinds of skills. They manipulate all sorts of things, whatever their textures and their shapes; they stack them up and knock them down and learn the dynamics of how they got scattered. Even to build a small toy house of blocks, one must mix and match knowledge of many kinds: about shapes and colors, speed, space and time, support and balance, stress and strain, and the economics of self-management. Why don't our computers know such things or learn them by observation? Do computers lack some qualities that only human brains possess? No, those limitations only persist because we've learned only certain ways to program them. For decades our standard approach to writing a program to solve a problem *P* has been: Find the *best* way to represent *P*; find the *best* way to represent the knowledge needed to solve *P*, and find the *best* procedure for solving problems like *P.*

These rules may seem quite sensible, but there's something basically wrong with them: They lead us to write only specialized programs that cope with solving only that kind of problem. The result is we've ended up with millions of specialized, expert programs. Each can do some circumscribed task, such as playing chess, or designing a bridge, but each can only do the job for which that program was designed. So the price of insisting on the "best" has entailed a great cost in resourcefulness. I think this is what brains do instead: Find several ways to represent each problem and to represent the required knowledge. Then when one method fails to solve a problem, you can quickly switch to another description.

To make machines deal with commonsense things, we must use multiple ways to represent knowledge, acquire huge amounts of that knowledge, and find commonsense ways to reason with it. Consider two examples from [1]:

Mary was invited to Jack's party.
She wondered if he would like a kite.

What leads the reader to understand Mary was thinking about a kite when there was no mention of "birthday" or "present?" In [6], I suggest how a suitable representation of "invited to party" could help one infer she is wondering whether a kite would be a suitable gift for Jack.

Jack needed some money, so he went and shook his piggy bank.
He was disappointed when it made no sound.

Why was Jack disappointed? Clearly because absence of sound meant the bank had no coins in it. One might try to deduce this from a physical audio-haptic simulation or, more likely, by analogy with a remembered description of a suitably similar event. But in either case, one needs representations of typical human goals and reactions. These would be hard to do with logical schemes because it would be hard to encode that kind of knowledge into many small, separate axioms instead of in terms of coherent examples (see [6]).

Which Representations to use for What Purposes?

Little is known about this subject. Here, I propose (condensed from [8]) one way to approach it.

Whenever you see a phenomenon, you can ask how many causes it has and how large are the effects of each cause.

Suppose you are lying down on a soft, comfortable bed. What is keeping you off the floor? Answer: you're supported by millions of very small forces, each supplied by some miniscule fiber that's pushing up on some spot of your skin. Each such force has such a small effect that if any of them should be

removed, you'd never know it was no longer present.

Now, what supports your comfortable bed? It is being held up by just four, strong legs. Each leg pushes up with a powerful force and removing any one of them *would* have a very large effect. So now let's make up a matrix of entries; one axis measures "the numbers of factors or causes involved" and the other axis indicates "how much effect each factor has."

When there are only a few causes, each having a small effect as depicted in the top-left corner cell of the matrix in Figure 1, then the problem will be easy to solve either by exhaustive search, or by simply recalling an answer.

When there are many causes, each with a rather small effect, then statistical methods and neural networks may work well (top-right corner cell of the matrix). However, such systems are likely to break down if those causes have different characters that interact in hard-to-predict ways. Also, because feed-forward systems usually lack internal short-term memories, they break down on sequential problems, and need to be managed by external systems.

Symbolic or logical reasoning (bottom-left corner cell of the matrix) can work very well when there are only a few influential factors—even when each has a large effect—except that, again, the search may exponentiate in the case of sequential processes.

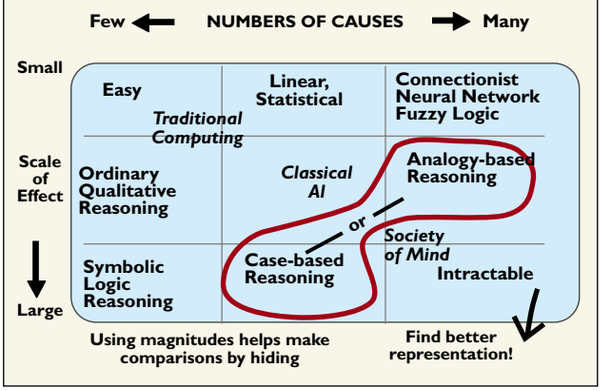
It is rare for any method to work, when many causes have large effects. Unless the system is linear, such problems may seem intractable. Sometimes, though, we can find solutions by reformulating those difficult problems by switching to different representations that emphasize fewer, more pertinent features, so that we can work with simpler descriptions. But it may be hard to reformulate unless we have multiple representations (bottom-right corner cell of the matrix).

Except in the corners of the matrix, we have multiple causes with modest effects and that's where heuristic programs succeed, using knowledge combined with controllable search. This is the realm of *classical AI research*. Here, analytical methods don't usually help, but we may be able to use our knowledge and experience to construct and apply analogies, plans, or case-based reasoning methods (adjacent cells to the bottom-right corner cell of the matrix). Such methods may not work perfectly, but they're frequently useful for practical purposes.

Physics researchers have made great progress by searching for elegant unified theories. But AI must deal with different complex worlds than the ones theoretical physicists face because they must deal with specific things that emerged from more inhomogeneous processes. We cannot expect to find uniform explanations to deal with that much diversity. Instead, we'll have to invent, combine, and reorganize an ever-increasing collection of increasingly incomplete theories.

Conclusion: We should not seek one uniform way to represent commonsense knowledge. Indeed, we'll frequently need to use several representations when we face a difficult problem and then we'll need additional knowledge about when and how to use them. A *causal-diversity* method may help, but eventually it must be replaced by more resourceful, knowledge-based methods that can generate useful new representations.

Figure 1. Causal-diversity matrix



Commonsense Knowledge Needs Multiple Representations

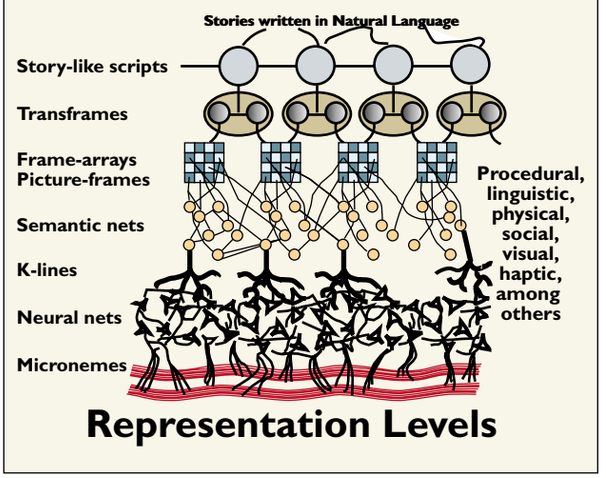
There is no best way to represent knowledge. The present limitations of machine intelligence stem largely from seeking unified theories incapable of reasoning well, while our purely symbolic logical systems lack the uncertain, approximate linkages that can help us make new hypotheses. Human versatility must emerge from a large-scale architecture in which each of several different representations can help overcome the deficiencies of the other ones. To do this, each formally neat type of inference must be complemented with scruffier kinds of machinery that embody the heuristic connections between the knowledge itself and what we hope to do with it.

Consequently, we need to build a *commonsense knowledge-base* that represents knowledge about so many things, like strings, roads, tools, energy, books, houses, or clothing; in other words, everything that most children know. And to use such a commonsense knowledge base, we'll need ways to link each

unit of knowledge to the uses, goals, or functions that each knowledge-unit can serve. To solve a significantly difficult problem, one needs to know ideas and tools useful for that purpose.

An entrepreneur might propose the following question: Why not build a system that searches the entire Web using millions of helpers all over the Net to accumulate all the knowledge it needs? One problem is *much of our commonsense knowledge information has never been recorded at all because it has always seemed so obvious we never thought of describing it.*

Figure 2. Architecture of representations.



Another problem is that we must first design suitable representations. You cannot learn *X* until you possess some suitable way to represent *X.*

A third problem is it's difficult for a system to improve itself until it has enough commonsense judgment to select changes that will lead to improving itself. You can't learn *X* until you have enough knowledge and processes to find a good representation for *X.* (Yes, this can be done, in principle, by an initially simplified evolutionary process. However, that will consume great spans of time, until it evolves good heuristics.)

Finally, a machine that learns such things by itself must know a lot about psychology—both about humans and about itself—to understand what it hears and sees. Consider the following discussion from [3]: "We human beings all get by, today, in the real world, speaking, and writing such terse, ambiguous utterances (such as Fred saying 'I want some chips') because we all draw on the same seven elements of shared context:"

1. The content of the previous sentences that have just gone by, in the dialogue.
2. The form of the previous sentences (word choices, sentence structure, tone, and so on).
3. The underlying substrate of general real-world knowledge that we assume practically everyone knows. In modern America, this encompasses recent history and current affairs, everyday physics, "household" chemistry, famous books and movies and songs and advertisements, famous people, nutrition, addition, weather, and so on. (Of course, in the case of a work of fiction, or an old chronicle, the "real world" means the world in which that utterance was set. For example, the sentences spoken by the narrator in *Dracula* are set in a fictional world akin to 19th century Europe, but with real vampires in that world. Even in that flight of fantasy, 99.9% of all the objects, events, places, or relationships have the same "true real world" structure and rules about them).
4. The underlying substrate of commonsense rules of thumb largely derived from shared experiences (dating, driving, dining, daydreaming) and human cognitive economies/limitations (misremembering, misunderstanding), and shared modes of reasoning both high (induction, intuition, inspiration, incubation) and low (modus ponens, dialectic argument, superficial analogy, pigeonholing).
5. The current short-term real-world situation/problem/task/environs that the speaker (or author) and listener (or reader) are in, or are talking about, and their respective roles in that situation or task, and what each presumes the short-term goals of the other to be in that conversation. (This includes lighting conditions, crowdedness, noisiness, each other's appearance, dress, and stance, among others.)
6. The long-term background/credentials/occupation/role of each party—at least those that the other party is aware of or, more importantly, believes to be true.
7. The history of any memorable experiences they shared together (and the roles they played in those events), any memorable prior conversations they had with each other. (The quality "memorable" often derives from some combination of unexpected, significant, and recent.)

When considering Lenat's elements of shared context, how could we put such diverse knowledge into the programs of our computers? Programmers often argue about whether it's best to represent knowledge with frames, scripts, rule-based systems, or even expressions of natural language. One person

says: "It is best to use logic." The next person says: "No. Use neural networks. Logic is too inflexible." The third person says: "No, neural nets are really more rigid because they try to reduce things to numbers, and don't have good ways to describe abstractions. Another person says: "You should use semantic networks where different ideas are connected by concepts!" And then the first person closes the circle by complaining: "No, semantic nets are too flexible and can lead to inconsistencies. Only formal logic can protect our systems from paradoxes."

The answer is we do not need to make such choices. Our causal-diversity matrix suggests each representation has merits and faults, and the only way to keep all their virtues is to use several different representations inside a single, larger system! So, instead of trying to find the best, we should ask a different kind of question: What kinds of reasoning would be good for this problem and which kinds of representations work well with those methods? Using logic-based reasoning can solve some problems in computer programming. However, most real-world problems need methods better at matching patterns and constructing analogies, or making decisions based on previous experience with examples, or by generalizing from types of explanations that have worked well on similar problems in the past.

Perhaps a good architecture theory based on multiple representations and multimodal reasoning would help us to design better systems that allow us to study and understand commonsense reasoning. Such an architecture would embed such representations as natural language, scripts, transframes, semantic nets, frames and frame arrays, K-lines and polynemes, neural nets, and micronemes. In [9], I describe some ways that such an architecture lends itself to the study of commonsense reasoning.

It is essential we understand multiple representations, but in order to do so we need some good ideas about how such different representations relate. For the purpose of discussion, Figure 2 depicts an architectural diagram that depicts relationships between various representations. I don't mean to say that our representations must be arranged all so hierarchically. However, new structures are usually made from older ones, and this representation tower might be a plausible brain-development scheme. Note the higher-level representations are especially suited for reflective thinking because they can represent mental events as well as external objective events. That's much more difficult to do at the lower, more simply reactive levels because it's very hard for those neural nets to represent their own internal activities, as noted in [7]. However, it's important to recognize there must also be similarly reactive components of our higher-level thinking, too, so try as we may, our introspections will always remain incomplete.

To make our computers personalized and easy to use, we must sensitize them to our needs; computers will need a better commonsense understanding of people and the world we live in. There are many small things we could do to improve the present-day interface. For example, scientists and engineers are working on schemes to enable computers to detect when their users are restless, disturbed, or upset. But we already know why those users fret. Instead of working on software designed to tranquilize those sufferers, let's give our machines enough common sense to make it more pleasant to work with them. C

References

1. Charniak, E. EECS Ph.D. dissertation. MIT, 1974.
2. Lenat, D.B. and Guha, R.V. *Building Large Knowledge-Based Systems.* Addison-Wesley, Reading, PA, 1990.
3. Lenat, D.B. *The Dimensions of Context-Space*. www.cyc.com/publications.htm.
4. McCarthy, J. and Hayes, P.J. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*. B. Meltzer and D. Michie (Eds.). Edinburgh University Press, 1969.
5. Minsky. M. *A Framework for Representing Knowledge*. MIT, 1974. Also, *In The Psychology of Computer Vision.* P.H. Winston (Ed.), McGraw-Hill, New York, 1975.
6. Minsky. M. *The Society of Mind.* Simon & Schuster, New York, 1986.
7. Minsky. M. Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. *Artificial Intelligence at MIT, Vol. 1*. Patrick H. Winston (Ed.), MIT Press, 1990. Reprinted in *AI Magazine,* 1991.
8. Minsky, M. Future of AI technology. *Toshiba Review 47,* 7 (July 1992). Also at media.mit.edu/people/minsky/papers/CausalDiversity.txt.
9. Minsky, M. *The Emotion Machine.* Pantheon, available in 2001.
10. Schank, R. and Abelson, R. *Scripts, Goals, Plans and Understanding.* Erlbaum, 1977.
11. Sussman, G.J. *A Computational Model of Skill Acquisition.* Elsevier, New York, N.Y. 1975. Technical Report AITR-297. MIT AI Lab, August 1973.

For detailed discussion and materials on commonsense reasoning and other related topics, please visit www.media.mit.edu/people/minsky

Marvin Minsky (minsky@media.mit.edu) is Toshiba Professor of Media Arts and Sciences and a professor of electrical engineering and computer science at the MIT Media Lab and MIT AI Lab in Cambridge, MA. He is the recipient of ACM's 1969 A.M. Turing Award.