



MMP

Student Position Paper

Contributors: David Dalrymple, Catherine Havasi, Dustin Smith, Bo Morgan, Rob Speer, Ken Arnold

Participants: Forrest Green, Scott Greenwald, Peter Schmidt-Nielsen, Alexey Radul, Katie Thomas

Preface

The MIT Mind Machine Project is composed of numerous groups across campus, each of which has a different pre-existing research agenda, but to be successful, we must all agree on a single, unified research agenda to which each of us can contribute. To that end, we gathered to determine our common interests, free from departmental politics and financial concerns. This document represents the consensus from that gathering.

Introduction

It is an unfortunate paradox that the primary characteristic of a strong AI system is that it can solve problems we hadn't thought of in advance; so to decide in advance which problem(s) we will measure success by is to ultimately doom ourselves to failure—we would create a weak AI to solve the specific problem(s). Thus, in proposing a vision and milestones for success, we must tiptoe around the tempting possibility of stating a tractable example problem which it shall be our goal to solve.

Instead, we present a landscape of ideas that we agree should be a part of the MMP effort, and applications of systems with those features, but we do not, at this stage in the project, consider any of the applications as ends in themselves.

Virtual Universes

One way to help alleviate the problem-specification issue is to propose a broad, possibly infinite class of problems, from which example problems can be randomly generated to test the efficacy of our systems. Even here, however, many classes of problems (such as for instance parsing any context-free grammar) would lead to a shortcut algorithm that does not exhibit the intelligence we ought to aim for. A metaclass of problem types that might exhibit more intellectual richness is problem-solving in virtual universes - textual/symbolic, physical simulation, or most likely, somewhere in between.

Furthermore, it may contribute significantly to a system's potential to achieve intelligent behavior if it has a capability for affecting some external objects and observing the results, whether this be on the Internet, in a robotic embodiment, or in a simulated world. The latter is likely the method of lowest cost and least risk.

Virtual worlds might consist of physical simulations or more symbolic, common-sense simulations, where certain actions are hard-coded to make discrete changes in the states of objects or other actors. (For instance, a virtual kitchen universe has been created that randomly selects and places common kitchen objects in space, where each reacts to actions like "open" or "turn on.")

At some point, it will become necessary for our systems to interact with humans from within the virtual universes by means of human-controlled avatars. This would enable learning by example from human interaction; without any intelligence to interact with, it would be much harder for such a "fishtank system" to develop human-like behaviors. We can begin by implementing a tiny subset of versatile natural language—a few prepositions, both spatial and temporal, and a collection of simple nouns and verbs—and gradually teaching the machine more symbols.

Stories

When an intelligent system has a goal, performs an action, and observes the consequences, the result is represented as a "story." While the word "story" is fuzzy and indicates a strong connection to human language, we all agree that a critical type of knowledge exists which could reasonably be referred to as a story: in particular, an archetypical reduction of the relationships between a certain set of events. This information lies between our current purely formal programming languages and the ambiguous, historically diluted soup of languages such as English. However, such a structure might be generated from parsing a linear narrative in English, and a system which does this reliably would be significant progress towards passing the Minsky-not-Turing test of understanding a children's story and answering questions about it.

When we apply our intelligence to solve problems, we don't reason through the entire problem from scratch. We can often solve much of the problem simply by relating it to our past experiences, and thinking about what has worked before. An intelligent system that understands stories is one that can solve new problems by making an analogy to a similar problem that it has solved before.

Non-sequential programs

If a story is an archetypical reduction of the relationships between events acting on objects, that sounds a lot like an asynchronous program: each event would be a computation, with the relationships being analogous to passing data. However, like stories, although in many cases they can be and are totally ordered, this need not be the case, and leads to a great deal more flexibility in representation and expressive abstracting power. Thus, developing further our asynchronous, directed-graph programming models is central to both the code and data representation for MMP.

To combine many ways of reasoning, we need an architecture that can link different reasoning processes so that they can share information and control each other. The usual

ways of separating concerns have complex interaction requirements (exact name alignment, tight sequencing requirements, restrictive types of data, fragile links). Our architecture must support panalogical reasoning from the ground up.

Pattern recognition; multi-modeling

When there are many stories in our system's knowledge base, it is necessary to identify which are relevant at any given time by analyzing the inputs and efficiently recognizing fragments of the patterns described by the stories. (In the simplest case, this would be a simple iterative search; high degrees of parallelism are also clearly possible, and there may be very clever forms of indexing—bordering on compilation—that assimilate the stories into a specialized recognition algorithm.) Once brought to the system's attention, the remaining part of a story then gives some (hopefully) useful information about what computations or actions to perform. However, if many stories are relevant, the system should be able to utilize the information from all of them, depending on context, even if they are contradictory. Furthermore, we all agree that an intelligent system must be able to easily translate between representations as appropriate. Essentially, a system for combining many, many different algorithms without requiring explicit rules for which algorithm to choose is desired.

Reflective thinking

Changing representations is only a specific reaction to failure. Eventually, our systems must be able not only to switch between hard-coded techniques, but to examine and improve their workings through experiencing failures. They should include a mechanism for reporting in some language the processes that led to these failures, and also a mechanism for being told this by an expert annotator in some language.

Logic and probability

We believe that, while tempting for a "reboot" of artificial intelligence, it is a mistake to assume, simply since many people utilize logic and probability theory in incorrect ways and get bad and/or hard-to-explain results, that they are useless as mathematical tools and should be categorically banned from our project. This does mean that we must be very careful and skeptical when we employ them—lest we fall into the same traps—but in the context of a system which can employ numerous methods for problem-solving, there are certainly some cases in which deductive or Bayesian inference can be useful, or even necessary.

In the long view, the distinction between symbolic and numerical reasoning is quite fuzzy - after all, where are the symbols in the brain? The answer is clear: symbols emerge from sparseness of input data. A properly architected system that uses only numerical values passed along a graph could give rise to symbolic behavior, just like the brain. On the other hand, a properly architected system that uses only symbolic values could give rise to numeric behavior, like a computer. Our systems should include aspects of both phenomena.

There are other false dichotomies besides "symbolic" and "connectionist" and recurring mistakes (poor software engineering infrastructure for large projects, overspecialized toy problems) in AI that we need to be aware of, while also not overlooking the contributions from the specialist sub-fields of AI in the past 30 years: object recognition, statistical machine learning, inductive logic programming, planning, etc.

Collaboration with other projects

Some people have asked us if we are collaborating with other (non-MIT) groups working on strong AI, such as OpenCog/Novamente, Numenta, and Cog-X. We think that in many or most cases, other AI groups tend to take a "silver bullet" approach, which we find unlikely to be correct (this is why the MMP "reboot" is necessary). However, we also believe that open communication channels and exchange of ideas with other researchers in the same field is invaluable, as we may be able to integrate some of their ideas into our broader context, as well as perturb the community to consider more balanced AI architectures.

Another common point that has been brought up in Internet reactions to our press release is that we should be anticipating memristor technology as it makes its way to commercial viability. While, in fact, our architecture and programming models should be able to transition to such hardware with surprising ease, it might be a good idea to get in touch with HP Labs (and/or IBM regarding ferromagnetic transistors) and plan a more concrete roadmap toward using such "exotic" hardware, within the next five years.

Finally, we should make some connections with cognitive science, cognitive neuroscience, and developmental psychology, as sources of experimentally tested ideas from our current understanding of how the human mind might work.

Applications

We all agree that our application space consists of five basic segments: helping us to solve new problems, assisting those with mental illness, teaching us, entertaining us, modifying interfaces to facilitate user interaction.

Problem Solving

A computer program that can recognize when a problem that is being solved resembles a previously solved problem, either directly or through an analogy. The program can then suggest a solution, either by projecting the analogical solution into the target domain or by finding a solution that previously worked.

Medical Applications

The field of mental health does not understand the complex spectrum disorders of schizophrenia, autism, psychopathy, etc. Many of these illnesses appear to be complex combinations of perceptual, motor, knowledge, planning, reflection, and social/mental skills. How exactly these illnesses may be understood in terms of a more complete theory of human mentality is exciting to even consider. First, we need to program such a complete computational system that works in theory before proving any empirical results.

Teaching

One example is a pedagogical representation of scientific common sense (for instance, objects fall because they are gravitationally attracted to the very massive Earth). By representing these semantics, students can not only experiment with simulations, but be explained to by the system about what is going on in natural language.

Entertaining

By representing stories in an abstract, non-linear fashion, we might someday be able to build a generative video game in which the path of the narrative is computed on-the-fly as the player(s) make(s) certain decisions.

Modifying User Interfaces

A computer program or Web site that can recognize common goals of its users could generate context-sensitive options or change its interface gradually over time to match its usage pattern.

For some applications (e.g. representing human-like non-player characters in a video game) we may want a system that replicates human flaws in cognition. For others (e.g. user interface, education) we may want a system that does not make human flaws in its own cognitive processes, but can emulate or describe human idiosyncracies of thought and thereby resolve ambiguities created by its human users.

Next Steps

- Fortnightly students' meeting
- Optional lunch twice weekly
- Launchpad account
- Regularly updated MMP blog (the modern form of a tech report series)
- Sysadminion(s)/codemonkey(s) (Katie) and other UROPs
- Student server - Xen instance, available from NeCSys