Gabe's Strategies for AI

Gabriel Pickard, gpickard@uos.de

University Osnabrück
Institute of Cognitive Science

November 5, 2008

Outline

Introduction
- How to Platform

Everyone is a Programmer
- Visual Programming
- Prose Programming

Reproduction
- Monetary Agent Interaction

Learning
- Value Fields
- Predator-Prey Model for Unsupervised Generative Learning

AI is a hard Problem

- Evolution took really long to reach our intelligence.
- Hunch: Monolithic approaches won't get us there; are too narrow.

Intelligent Systems

Lets first get straight what I mean by "intelligent systems":

- I think language-wise, so to me this means a grip on semantics and pragmatics.
- Basically: access to background-knowledge and a sense of appropriate ways of applying it.
- So I'm less interested in building autonomous robots. Though it's an interesting field - and even there, an idea of semantics and pragmatics can be found.

Big Model

- Provide a sandbox people use all the time (browser/chat-client/smartphone-ware?)
- Make it so that many people work on it
- Multiplicity of Approaches; learned, evolved, hand-coded
- Develop ways (P2P?) to exchange program-parts
- Emphasis on techniques that make *everyone a programmer*

Big Model - Criticism

- Extremely rough idea
- Wants a lot
- Browser-as-Platform might be a direction (but JavaScript...?)
- I've argued that I'd prefer the Instant Messenger as platform

Instant Messenger As Platform

- Built-in contact management
- Dialogue based ⇒ interesting for AI-dialogue systems
- Affinity for parallelized programming languages (erlang)
- Protocolls for push-communication

Instant Messenger As Platform - Criticism

- No unified scripting/extension-system
- As it is today, browser simply rules
- Maybe a morph of both browser & IM-client is of interest.

Visual Programming

Combine several developments to something that's really user-friendly as well as computationally powerful:

- upcoming need for massive multithreading on multi-cores using languages such as erlang
- bigscreen, multi-touch hardware such as microsoft surface
- with visual & dataflow programming techniques.

Visual Programming - A Proposal

- Use erlang as a base (or create some kind of dataflow meta-language)
- Do visual part in something like Adobe AIR or JavaScript
- Possibly extend erlang by some push/pull functionality
- Make plugin-eable (specifically the visual part)

Prose Programming

- Program in natural language
- Blurring distinction between program and data
- Let input sit around; Continuously process input + memory (similar to Hofstadter's copycat)
- Processing should maximise some confidence-measure(s)
- Possible toy problem: recipes and a virtual kitchen

Prose Programming - Recording Activities

The important problem is to bridge the gap between text and machine-instructions:

- First of all you need a well-structured set of instructions ($\Rightarrow$ sandbox)
- Record activity-description correlations
- Activities in sandbox naturally form a dataflow-graph
- Combine these with a unification/anti-unification framework (and re-representation)
- Would be cool to do with speech-input, would make easier for humans to record

Prose Programming - Maximizing Confidence

- Allow for abstract program/data – way how to integrate is sketchy (somehow hierarchical?)
- Different confidence-measures should be defined
- (e.g. based on previous user-feedback, co-occurence in recorded sessions, degrees of re-representation necessary, depth in hierarchy)
- How to deal with un-understood input: try processing in unsupervised manner; let sit; don't panic

Prose Programming - Problems

- Anaphora-resolution
- Probably negation would also be a problem
- How to do – and deal – with error and correction? (In recording and execution)

Reproductive Agents

- Only self-reproduction/maintainance (Autopoiesis) leads to true evolution
- For natural mutation to occur one probably needs a non-crisp environment (?)
- Is it possible to embedd crisp programs into subsymbolic processing?
- This is unfinished work, verily

Monetary Agents

- If using evolutionary (reproductive) agents, one needs selection-pressure
- Monetary systems are a classical method of pressurizing agents to do things
- Auction off computational resources such as processor-time, memory, screen real-estate
- Use LETS-style currencies
- Allow for in-trading & sub-contractors

Memorizing with neural nets is difficult

- Too many parameters
- Too slow learning
- Really not viable for big structures to be maintained for longer time (mid- to long-term memory)
- (that's what my thesis says)

Subsymbolic processing is desirable

- Can be numerically adapted
- More robust
- Easier to morph/mutate

Value Fields

- Give neural nets a subsymbolic repository
- Shape this as a virtual world in which they can move around like robots
- (so they have motor-outputs and read/write access)
- Basically neurally operated continuous-valued cellular automata

Predator-Prey Model for Unsupervised Generative Learning

- In many domains there's lots of positive data and no counter-examples
- Language is a prime example of such a domain
- Train a generator (prey) to trick a classifier (predator) into not being able to distinguish between generator-output
- E.g. generating written text as image using neural nets