Lecture 5
A Brief History of AI:
The Rise of Knowledge-based and Expert Systems (1969-89)

Rob Gaizauskas

Lecture Outline

- Historical Overview
  - Precursors (… – 1943)
  - Gestation and Birth (1943 – 1956)
  - Golden Early Years (1956-1969)
  - The First “AI Winter” (1966-73)
  - Rise of Knowledge-based and Expert Systems (1969-1989)
  - New Paradigms: Connectionism; Intelligent Agents; Embodied AI (1986 – present)
  - Scientific Method and Big Data (1987 – present)
- Reading:
  - Russell and Norvig (2010), Chapter 1 “Introduction”
  - Wikipedia: History of Artificial Intelligence. http://en.wikipedia.org/wiki History_of_artificial_intelligence

Lecture Outline

- Historical Overview
  - Precursors (… – 1943)
  - Gestation and Birth (1943 – 1956)
  - Golden Early Years (1956-1969)
  - The First "AI Winter" (1966-73)
  - Rise of Knowledge-based and Expert Systems (1969-1989)
  - New Paradigms: Connectionism; Intelligent Agents; Embodied AI (1986 – present)
  - Scientific Method and Big Data (1987 – present)
- Reading: * = mandatory
  - *Russell and Norvig (2010), Chapter 1 "Introduction"
  - *Wikipedia: History of Artificial Intelligence. http://en.wikipedia.org/wiki History_of_artificial_intelligence

Rise of Knowledge-based and Expert Systems (1969-1989)

- Need for knowledge
- DENDRAL
- MYCIN
- Schank – Scripts
- Minsky – Frames
- Commercial Expert Systems
- Return of Research Funding

The Need for Knowledge

- First 10-15 years of AI research showed that weak methods were inadequate
  - I.e. general-purpose search methods reasoning in elementary steps did not scale up to large or difficult problems (combinatorial explosion, frame and qualification problems)
- Alternative to weak methods is to use domain-specific knowledge and reason in large steps in narrow areas of expertise
- Thus, next stage in development of AI was characterised by the acquisition and application of domain-specific expertise in knowledge-based/expert systems

DENDRAL

- DENDRAL (1969) developed by Buchanan, Feigenbaum and Lederberg at Stanford
- Goal was a system that could infer the structure of a molecule from information provided by a mass spectrometer
- DENDRAL is given
  - The elementary formula of a molecule (e.g. $C_6H_{13}N0_2$)
  - Mass spectrum giving masses of molecule fragments generated when bombarded by electron beam (e.g. mass spectrum might have peak at m = 15 corresponding to the mass of a methyl fragment CH_3)

DENDRAL (cont)

- Naïve version of programme generated all possible structures consistent with formula, predicted what mass spectrum would be for each and compared this with observed spectrum
  - Intractable for all but small molecules
- Instead consulted analytical chemists and found they worked by looking for patterns of peaks in the spectrum that suggested common substructures. E.g.

  If there are 2 peaks at x_1 and x_2 such that

  a) $X_1 + x_2 = M + 28$ (M = mass of whole molecule)

  b) $X_1 - 28$ is a high peak

  c) $X_2 - 28$ is a high peak

  d) At least one of x_1 and x_2 is high

  then there is a ketone subgroup

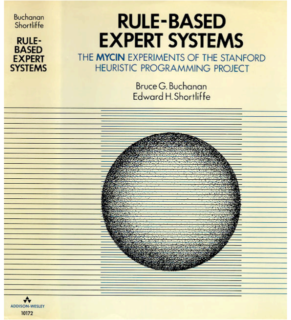
- By recognising particular substructures, the number of possible candidate structures is reduced enormously

DENDRAL (cont)

- DENDRAL powerful because theoretical knowledge needed to solve problem had been mapped from general, “first principle” forms
    - e.g. “find all possible structures consistent with formula and compare with formula”

  to specialized “cookbook recipes”
    - like ‘if-then’ rule on previous slide
- I.e. shift from general reasoning (weak methods) over axioms representing first principles in the domain to use of experts’ rules that chunk/compile large amounts of knowledge in the domain into specific rules
- DENDRAL the first knowledge-intensive system
    - I.e. its expertise stemmed from large numbers of special-purpose rules

MYCIN

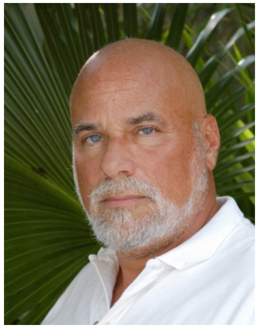


- Following success of DENDRAL, Feigenbaum and others at Stanford explored how expert systems could be developed in other areas
- MYCIN (1972) designed to diagnose blood infections and recommend antibiotics + dosages
  - Contained inference engine + ~600 rules
  - Queried physician with long list of yes/no questions
  - Output list of candidate culprit bacteria ranked by probability of each diagnosis, system's confidence in each diagnosis, reasoning behind diagnosis and recommended drug treatment

  See http://en.wikipedia.org/wiki/MYCIN
- Results: performed as well as some experts and outperformed junior doctors (recommended correct treatment in 69% of cases)

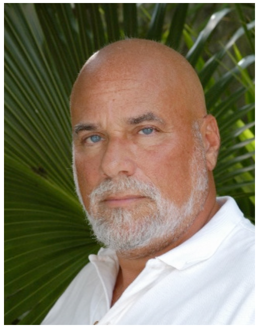
MYCIN

- Two major differences from DENDRAL
  1. No general theoretical model from which MYCIN rules could be deduced
     - Rules had to be acquired from experts
  2. Rules had to reflect uncertainty associated with medical knowledge
     - MYCIN used an approach based on certainty factors
     - Dispute about whether this was an appropriate way to address uncertainty (why not Bayesian statistics?)
- Never deployed
  - Some ethical/legal issues
  - Biggest problem was lack of general electronic medical systems into which it could be integrated
    - Standalone system that required patient details to be entered via in response to long list of questions – single session could take ~30mins – not feasible for use by busy clinicians
- Revealed issues surrounding acquiring necessary rules from experts: knowledge acquisition bottleneck



Schank – Scripts

- In NLP successes and limitations of programs like SHRDLU showed how understanding NL also relied on large amounts of general knowledge about the world
- Yale linguist/AI research Roger Schank + students pursued a research program on language understanding through the 1970s in which
  - Emphasis was less on language itself (e.g. on syntax)
  - Concentrated on representing and reasoning with knowledge required for language understanding
- Argued people have scripts – representations of stereotypical situations – which are used to interpret stories about such situations



Schank – Scripts

- Famous example is the restaurant script. Describes
  - typical roles – customer, waitress, chef, cashier
  - Typical sequence of actions:

    entering; ordering; eating; exiting

  and the actions sub-actions and role players

  - Given a restaurant script and a NL story like:

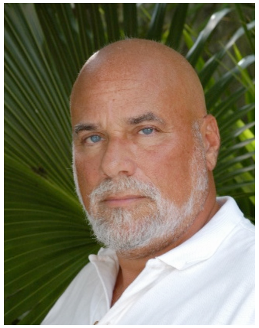
    John went to a restaurant. The hostess seated John. The hostess gave John a menu. The waiter came to the table. John ordered lobster. John was served quickly. John left a large tip. John left the restaurant.

  Schank's "Script Applier Mechanism" (SAM) program could answer questions like

  - What did John eat? (lobster)
  - Who gave John the lobster? (probably the waiter)
  - Why did John leave a large tip? (because he was served quickly)

  where the answer is not in the text, but requires inference based on world knowledge (supplied here by the script)

Schank – Scripts



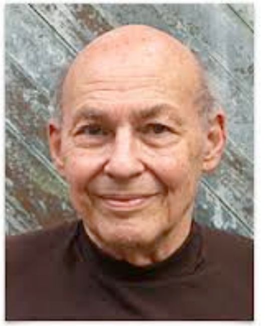
- Compelling approach but:
  - How do we/computers acquire scripts?
    - Don't want to manually code them – how can they be learned?
  - How many scripts are there?
    - Are there scripts for getting out bed? Putting on socks? Etc.?
  - How fine-grained are they?
    - Is there a script for an Indian restaurant as well as for restaurant?
- Schank tried to answer these questions in later work but the approach has not been pursued
  - However, influenced an important sub-area of applied natural language processing called information extraction

Minsky – Frames

- Like scripts, frames were an approach to representing and organizing the large amounts of world knowledge need for AI applications
- Assembled facts about object and entity types and arranged them into large taxonomic hierarchies
- Collection of facts, procedures and default values for an object type is called a frame. For example:

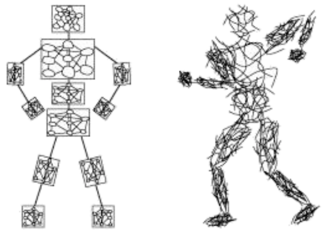
Slot	Value	Type
Boy	-	(this frame)
ISA	Person	(parent frame)
SEX	Male	(instance value)
AGE	Under 12 yrs.	(procedural attachment - sets constraint)
HOME	A Place	(frame)
NUM_LEGS	Default = 2	(default, inherited from Person frame)

Minsky – Frames



- Knowledge representation languages (like KL-One) emerged to support organization of hierarchies of frames and inference (e.g. inheritance) over them.
- These were the antecedents of:
  - Object orientation/inheritance in programming languages like Java
  - Description logics, ontologies and the semantic web

Neats vs Scruffies



- Tension arose between
  - those who favoured formal logical + rule-based approaches (McCarthy, Feigenbaum, Kowalski + the Prolog community in Europe)
  - those who tried to capture and use world/commonsense knowledge in more intuitive structure like scripts, frames and semantic networks (Schank, Minsky and his students)

  Became characterised as dispute between Neats and Scruffies
- See

  http://thedailyomnivore.net/2013/01/23/neats-vs-scruffies/

  https://en.wikipedia.org/wiki/Neats_vs._scruffies

Knowledge-based Systems: Summary

- Knowledge-based systems comprise two sub-systems
  - Knowledge base:
    - Stores facts about the world in some knowledge representation formalism both about
      - Classes: dogs are mammals, and
      - Instances: Snoopy is a beagle
  - Inference engine:
    - Answers queries about what is true or false according to KB (may need to reason do this – Is Snoopy a mammal?
    - Asserts new knowledge into the KB
    - May provide explanations for answers
- Inference engine work over IF-THEN rules like

  R1: man(x) => mortal(x)

  and may use
  - Forward Chaining: reason from, e.g. man(socrates) to mortal(socrates) OR
  - Backward Chaining: when seeking to determine whether, e.g. mortal(socrates) try to prove man(socrates)
- KBS typically model a narrow, specialised domain

Knowledge-based Systems: Summary Strengths and Weaknesses

- Strengths
  - By separating knowledge (KB) from process (reasoning), knowledge can be stated explicitly in a form that domain experts can create and modify
    - Users can maintain systems and computer scientists do not need to be involved to modify code every time domain rules/facts change
- Weaknesses
  - Knowledge acquisition bottleneck
    - Experts either not available or cannot formalise what they know
  - Performance – AI systems emphasis on development rather then performance
  - Integration with legacy or other corporate databases

Rise of Knowledge-based and Expert Systems (1969-1989)

- Need for knowledge
- DENDRAL
- MYCIN
- Schank – Scripts
- Minsky – Frames
- Commercial Expert Systems
- Return of Research Funding

Commercial Expert Systems

- During the 1980s "expert systems" – what knowledge-based systems were called in the business world – found their way into the market
- First commercially successful system: R1 (later XCON)
  - developed by Digital Equipment Corp (DEC) to configure newly ordered computers
  - eventually had about 2500 rules
  - by 1986 had processed 80K orders and was 95-98% accurate
  - estimated to save DEC $25M a year
  - Helped ensure correct peripherals, cables, connections, software (e.g. device drivers), etc. included in order
  - Why R1? creator McDermott at CMU: "Three years ago I wanted to be a knowledge engineer, and today I are one."

(http://en.wikipedia.org/wiki/XCON)

Commercial Expert Systems

- By 1988: DEC had 40 expert systems; Dupont had 100 saving $10M/year; most major US corps using or developing them
- Application categories and examples

(from https://en.wikipedia.org/wiki/Expert_system)

Category	Problem Addressed	Examples
Interpretation	Inferring situation descriptions from sensor data	Hearsay (Speech Recognition), PROSPECTOR
Prediction	Inferring likely consequences of given situations	Preterm Birth Risk Assessment[34]
Diagnosis	Inferring system malfunctions from observables	CADUCEUS, MYCIN, PUFF, Mistral,[35] Eydenet,[36] Kaleidos[37]
Design	Configuring objects under constraints	Dendral, Mortgage Loan Advisor, R1 (Dec Vax Configuration)
Planning	Designing actions	Mission Planning for Autonomous Underwater Vehicle[38]
Monitoring	Comparing observations to plan vulnerabilities	REACTOR[39]
Debugging	Providing incremental solutions for complex problems	SAINT, MATHLAB, MACSYMA
Repair	Executing a plan to administer a prescribed remedy	Toxic Spill Crisis Management
Instruction	Diagnosing, assessing, and repairing student behavior	SMH.PAL,[40] Intelligent Clinical Training,[41] STEAMER[42]
Control	Interpreting, predicting, repairing, and monitoring system behaviors	Real Time Process Control,[43] Space Shuttle Mission Control[44]

Return of Research Funding

- In 1981 the Japanese government announced $850M funding of the Fifth generation computer project
  - objectives were to write programs and build machines that could carry on conversations, translate languages, interpret pictures, and reason like human beings
  - Prolog chosen as primary computer language for the project
- The UK funded (1983-87) the £350 million Alvey project on IT which included AI and NPL as areas of focus
- In the US
  - a consortium of American companies formed the Microelectronics and Computer Technology Corporation (MCC) to fund large scale projects in AI and IT
  - DARPA founded the Strategic Computing Initiative and tripled its investment in AI between 1984 and 1988.

References

Russell, Stuart and Norvig, Peter (2010) Artificial Intelligence: A Modern Introduction (3rd ed). Pearson.

Schank, R. C. and Abelson, R.P. Scripts, Plans, and Knowledge.

Wikipedia: ALPAC. http://en.wikipedia.org/wiki/ALPAC (visited 26/11/13).

Wikipedia: Alvey Programme. http://en.wikipedia.org/wiki/Alvey_Programme (visited 27/11/13).

Wikipedia: Dendral. http://en.wikipedia.org/wiki/Dendral (visited 27/11/13).

Wikipedia: Expert System. http://en.wikipedia.org/wiki/Expert_system (visited 10/10/15).

Wikipedia: Fifth Generation Computer. http://en.wikipedia.org/wiki/Fifth_generation_computer (visited 27/11/13).

Wikipedia: Frame (artificial intelligence)http://en.wikipedia.org/wiki/Frame_(Artificial_intelligence) (visited 27/11/13).

Wikipedia: History of Artificial Intelligence. http://en.wikipedia.org/wiki History_of_artificial_intelligence_ (visited 12/11/13).

Wikipedia. Mycin. http://en.wikipedia.org/wiki/Mycin (visited 27/11/13).

Wikipedia. Xcon. http://en.wikipedia.org/wiki/Xcon (visited 27/11/13).