Introduction

www.scs.s.tcd.ie/Tim.Fernando/KRA

Key Phrases:

Can machines think?
- Turing test & ELIZA effect
- AI-complete (contra low hanging fruit)

Agent & environment
- Cognitive Revolution & Big Data

Knowledge Representation & Automata
Can machines think? (Turing 1950)

**Turing test:** can C tell A from B?

From Wikipedia, (Juan Alberto Sánchez Margallo)

Intelligence operationalized: subject to testing
Can machines think? (Turing 1950)

**Turing test:** can C tell A from B?

From Wikipedia, (Juan Alberto Sánchez Margallo)

Intelligence operationalized: subject to testing ... cheating?
ELIZA (Weizenbaum, 1964-66) & artful deception
- use pattern matching and substitution to fake it (e.g. Prolog)

Caution: Programs may appear to work better than they do

Siri rage (Urban dictionary):
When you get enraged because Siri just doesn't get it.
ELIZA (Weizenbaum, 1964-66) & artful deception
- use pattern matching and substitution to fake it (e.g. Prolog)

ELIZA effect: humans anthropomorphise computers

e.g. when ATM says “thank you”
ELIZA (Weizenbaum, 1964-66) & artful deception
- use pattern matching and substitution to fake it (e.g. Prolog)

**ELIZA effect:** humans anthropomorphise computers

  e.g. when ATM says “thank you”

An AI problem is **AI-complete** if any AI problem is mechanically reducible to it (i.e., it is at least as hard as any other).

E.g. Natural Language Understanding

  *The town councilors refused to give the demonstrators a permit because they feared violence.*

  *Who feared violence?*  
  
  T. Winograd
ELIZA (Weizenbaum, 1964-66) & artful deception
- use pattern matching and substitution to fake it (e.g. Prolog)

**ELIZA effect**: humans anthropomorphise computers

e.g. when ATM says “thank you”

An AI problem is **AI-complete** if any AI problem is mechanically reducible to it (i.e., it is at least as hard as any other).

E.g. Natural Language Understanding

*The town councilors refused to give the demonstrators a permit because they advocated violence.*

*Who advocated violence?*

T. Winograd
ELIZA (Weizenbaum, 1964-66) & artful deception
- use pattern matching and substitution to fake it (e.g. Prolog)

**ELIZA effect**: humans anthropomorphise computers

e.g. when ATM says “thank you”

An AI problem is **AI-complete** if any AI problem is mechanically reducible to it (i.e., it is at least as hard as any other).

E.g. Natural Language Understanding

*The town councilors refused to give the demonstrators a permit because they advocated violence.*

*Who advocated violence?*  

T. Winograd

**Caution**: Programs may appear to work better than they do

**Siri rage** (Urban dictionary):

*When you get enraged because Siri just doesn’t get it.*
Chinese room argument (Searle’s thought experiment)

- a clerk can follow instructions for communicating in Chinese without understanding Chinese

Strong vs Weak AI

Chinese room argument (Searle’s thought experiment)

- a clerk can follow instructions for communicating in Chinese without understanding Chinese

  mindless obedience ≠ mind from biological processes

Strong vs Weak AI

**Chinese room argument** (Searle’s thought experiment)

- a clerk can follow instructions for communicating in Chinese without understanding Chinese
  
  mindless obedience \(\neq\) mind from biological processes

**Just do it**  
\(\rightarrow\) Black Box, judged by its actions
Locating intelligence (black box)

Intelligence: (abilities, goals, ..., experience) $\mapsto$ action

Poole & Mackworth
Locating intelligence (black box)

Intelligence: (abilities, goals, ..., experience) $\leftrightarrow$ action

Turing test: what to say $\sim\rightarrow$ what to do

Poole & Mackworth
Between agent and environment

<table>
<thead>
<tr>
<th>agent</th>
<th>environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>program</td>
<td>data</td>
</tr>
<tr>
<td>Cognitive Revolution</td>
<td>Big Data</td>
</tr>
<tr>
<td>hard-wired</td>
<td>experienced</td>
</tr>
<tr>
<td>rationalist</td>
<td>empiricist</td>
</tr>
<tr>
<td>nativist</td>
<td>behaviorist</td>
</tr>
<tr>
<td>innate</td>
<td>tabula rasa</td>
</tr>
<tr>
<td>nature</td>
<td>nurture</td>
</tr>
</tbody>
</table>

Turing machine &
specialized automaton
## Between agent and environment

<table>
<thead>
<tr>
<th>agent</th>
<th>environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>program</td>
<td>data</td>
</tr>
<tr>
<td>Cognitive Revolution</td>
<td>Big Data</td>
</tr>
<tr>
<td>hard-wired</td>
<td>experienced</td>
</tr>
<tr>
<td>rationalist</td>
<td>empiricist</td>
</tr>
<tr>
<td>nativist</td>
<td>behaviorist</td>
</tr>
<tr>
<td>innate</td>
<td>tabula rasa</td>
</tr>
<tr>
<td>nature</td>
<td>nurture</td>
</tr>
</tbody>
</table>

Turing machine & specialized automaton

Learning (from environment)
trial & error: “data as oil”
**Between agent and environment**

<table>
<thead>
<tr>
<th>agent</th>
<th>environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>program</td>
<td>data</td>
</tr>
<tr>
<td>Cognitive Revolution</td>
<td>Big Data</td>
</tr>
<tr>
<td>hard-wired</td>
<td>experienced</td>
</tr>
<tr>
<td>rationalist</td>
<td>empiricist</td>
</tr>
<tr>
<td>nativist</td>
<td>behaviorist</td>
</tr>
<tr>
<td>innate</td>
<td>tabula rasa</td>
</tr>
<tr>
<td>nature</td>
<td>nurture</td>
</tr>
</tbody>
</table>

Turing machine & specialized automaton

Learning (from environment)

trial & error: “data as oil”

Moving target: changing agent & environment
So having stated the motivation for working on AI and the challenges, how should we actually make progress?

Given a complex real-world task, at the end of the day, we need to write some code (and possibly build some hardware too). But there is a huge chasm between the real-world task and code.

A useful paradigm for solving complex tasks is to break them up into two stages. The first stage is **modeling**, whereby messy real-world tasks are converted into clean formal tasks called models. The second stage is **algorithms**, where we find efficient ways to solve these formal tasks.

### Algorithms (example)

**Formal task**:
- **Input**: list \( L = [x_1, \ldots, x_n] \) and a function \( f: X \to \mathbb{R} \)
- **Output**: \( k \) highest-scoring elements

**Example** (\( k = 2 \)):
- \( L \): A B C D
- \( f \): [3 2 7 1]

Two algorithms:
- Scan through to find largest, scan through again to find the second largest, etc.
- Sort \( L \) based on \( f \), return first \( k \) elements

Let's start with something that you're probably familiar with: algorithms. When you study algorithms, you are generally given a well-defined formal task, something specified with mathematical precision, and your goal is to solve the task. A solution either solves the formal task or it doesn't, and in general, there are many possible solutions with different computational trade-offs.

As an example, suppose you wanted to find the \( k \) largest elements in a list of \( L = [x_1, \ldots, x_n] \) according to given a scoring function \( f \) that maps each element into a real-valued score. Solving a formal task involves coming up with increasingly more efficient algorithms for solving the task.
From Narrow to General AI

• So having stated the motivation for working on AI and the challenges, how should we actually make progress?

• Given a complex real-world task, at the end of the day, we need to write some code (and possibly build some hardware too). But there is a huge chasm between the real-world task and code.

Paradigm

Real-world task

Modeling

Formal task (model)

Algorithms

Program

unstructured information $\leadsto$ actionable knowledge

Demis Hassabis

Autonomous = perform tasks in complex environments without constant user guidance

Adaptive = improve performance by learning from experience

www.theguardian.com/technology/2016/feb/16/demis-hassabis-artificial-intelligence-deepmind-alphago
Traditional approach

A spell checker:

input "hte" → complex program → output "the"

Problem: complexity becomes unwieldy

Machine learning approach

Training examples

- hte ⇒ the
- jeopardy ⇒ jeopardy
- affedave ⇒ affidavit
- misilous ⇒ miscellaneous

Learning algorithm

input → simple program

parameters = [3.2, 1.2, ...]

output
We now embark on our tour of the topics in this course. The topics correspond to types of models that we can use to represent real-world tasks. The topics will in a sense advance from low-level intelligence to high-level intelligence, evolving from models that simply make a reflex decision to models that are based on logical reasoning.

A spell checker:

- **Traditional approach**
  - A spell checker:
  - **input**: "hte"
  - **complex program**
  - **output**: "the"

  **Problem**: complexity becomes unwieldy

- **Machine learning approach**
  - **Training examples**
    - hte $\Rightarrow$ the
    - jeopardy $\Rightarrow$ jeopardy
    - affedave $\Rightarrow$ affidavit
    - misilous $\Rightarrow$ miscellaneous

  - **Learning algorithm**
  - **input**: simple program
  - **output**: parameters $= [3.2, 1.2, ...]$

  - **Problem**: complexity becomes unwieldy

Search problems
Markov decision processes
Adversarial games
Constraint satisfaction problems
Bayesian networks

---

**Reflex**

"Low-level intelligence"

**States**

**Variables**

"High-level intelligence"

**Logic**

**Machine learning**
Back in Trinity

4th year ML modules

- CSU44061 Machine Learning
  Semester 1 (5 ECTS)

- CSU44062 Advanced Computational Linguistics
  Semester 1 (5 ECTS)
  unsupervised ML for natural language processing
Back in Trinity

4th year ML modules

▶ CSU44061 Machine Learning
   Semester 1 (5 ECTS)

▶ CSU44062 Advanced Computational Linguistics
   Semester 1 (5 ECTS)
   unsupervised ML for natural language processing

CSU44060: backing up a bit

▶ Logic & Agents as Turing machines
▶ Description Logics
▶ Finite-state methods
▶ Action, change & time
▶ Trakhtenbrot’s & Fagin’s theorems