Can Machines Think? *The Past, Present, and Future of AI*

Can Machines Think?

The past, present, and future of Artificial Intelligence

Peter Millican

Tan Chorh Chuan Professor,
National University of Singapore

(also Gilbert Ryle Fellow and Professor of Philosophy, Hertford College, Oxford)

1

Agenda of the Lecture

Introduction: From Philosophy to Computing – close relatives!

1. Alan Turing – mathematician, code-breaker, and visionary.
2. The inspiration for Turing's "computing machines" (1936).
3. The rise of "Classical AI" (1950s – 1960s)
4. AI disappointments (1970s – 1980s)
5. Bouncing back with new methods (1990s-2000s)
6. The age of Deep Learning (2012 –).
7. Are we heading for conscious AI?

2

2



Lincoln College, Oxford 1976-83
Mathematics => Philosophy



University of Glasgow, 1983-84
Philosophy of Religion, Hume, Ethics



BBC B Computer:
Hedging against post-Thatcher academic unemployment



Hertford College, Oxford 2005 – Professor of Philosophy



University of Leeds, 1985-2005
Computing and Philosophy

1984: A Miracle!

The University of Leeds ***re*-advertises** a permanent "New Blood" Lectureship in Computing and Philosophy – probably the only such post to have been created in the UK that century.

3

3

Philosophy, Programming, and Pedagogy

Philosophy was my passion, but at Leeds my research had to go on the back burner initially, as I had two more essential duties:

- Stemming student attrition from Philosophy, which just as I joined lost three long-term full-timers. (At 26, I was by 15 years the youngest academic.)
- Teaching Logic and Mathematics, and learning Computer Science and Software Engineering to at least the level of supervising Masters students.

Hence my career from the start emphasised ***engaging*** teaching, and (after the Philosophy attrition had abated) focusing on ***interdisciplinary*** teaching on the borders between Philosophy and Computing.

- My "Computing for All" modules – taught on behalf of both Philosophy and Computing – were by 2003 attracting around 1,500 registrations per year, (£½m gross), with my software used both for administration and teaching. *

4

4

The Perennial Question (for over 40 years!)

"What have Computing and Philosophy got to do with each other?"

When advocating for the new Computer Science & Philosophy degree at Oxford (in 2012 the first new undergraduate Philosophy programme there for 39 years), I put together ***www.philocomp.net*** to answer this question.

But very briefly, four easy answers are:

1. LOGIC: Invented by Philosophers (Aristotle, Frege); central to Computer Science.
2. AI: Provides a new way of approaching many traditional philosophical issues about reason, language and understanding, knowledge and probability, art and creativity, the self and consciousness, action and choice, responsibility and moral status, and others.
3. ETHICS: Artificial agents require norms of action to be *explicitly* codified. Modern AI has also raised a host of urgent practical moral problems including unemployment, algorithmic bias, social media addiction, privacy and deepfakes, and killer robots.
4. MODELLING AND THOUGHT-EXPERIMENTS: Best illustrated by example … *

5

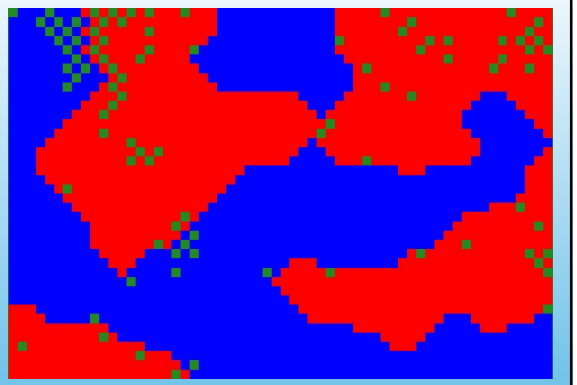
5

Explaining Social Segregation

Here is a city divided into housing blocks. Some (green) are empty. The others are occupied by families adhering to one of two religious groups, the Reds and the Blues.

Does this segregation imply that the two groups really dislike each other?

That's the most obvious conclusion to draw, but it might not be correct.



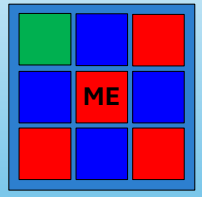
An alternative hypothesis is that the families have a mild preference for neighbours who share their faith, and that what causes the segregation is a cascade effect resulting from just a small number of families initially moving to get away from places where they are significantly outnumbered. This is an interesting hypothesis, but quite impractical to explore without computational help.

6

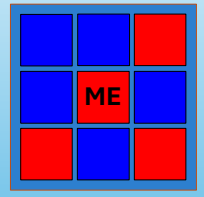
6

How the Model Works

We suppose that families are "unhappy" if the other colour has a majority of **2 or more** amongst their neighbours. Suppose I'm a Red, for example:



3 Red neighbours
4 Blue neighbours
Balance: -1
HAPPY



3 Red neighbours
5 Blue neighbours
Balance: -2
UNHAPPY

A family that's unhappy will try to find an empty (green) block to which they can happily move, in which case they will swap places with it. After they've moved, we go on to look for another family that's unhappy, and so on.

7

7

1. Alan Turing: mathematician, code-breaker, visionary

Mathematician

Turing's famous 1936 paper introduced the theoretical "Turing machine" — an abstract model defining the limits of what is computable. These limits still apply to every algorithm run on every computer ever built.

Code-Breaker

At Bletchley Park, Turing led the team that cracked the Enigma cipher used by Nazi forces. His Bombe machine automated the decryption of German messages, shortening the war by an estimated two years.

Visionary

His 1950 paper "Computing Machinery and Intelligence" proposed the Turing Test for a thinking machine and also imagined learning machines. It provoked serious discussion of Artificial Intelligence long before the term existed.

The Turing Award – the highest honour in Computer Science – is named in his memory, and his face, together with a prophetic quotation and images of his work, is on the UK £50 note.

8

8

The Turing Test (1950)



Interrogator
(here Alan Turing with a teletype that looks remarkably like an Enigma machine)



Typed Messages

Human respondent
(here Charles Babbage or Ada Lovelace)

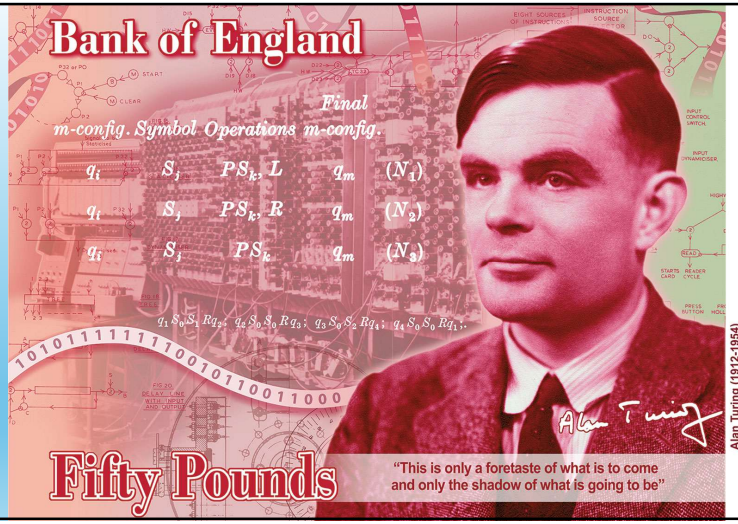


Computer respondent

The computer pretends to be human, while the interrogator tries to work out which respondent is which. The computer "passes" the test if the Interrogator cannot reliably tell which is the computer and which is the human.

9

9



10

10

The Turing Machine (1936)

Here is a Turing Machine simulator, * running a program from §3 of his 1936 paper. It is generating an unending sequence of 0s and 1s, representing (in binary) the number:

[0.] 0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 ...

(which is roughly 0.179184 in decimal).

The Turing Machine is driven by a table of very limited instructions (as shown in the image), and yet despite these strict limitations, it seems that by devising a suitable machine table, *any algorithm that can be performed by any computer (including a human computer), can be carried out by some Turing machine*. This **Church-Turing Thesis** is generally believed to be true, thanks to arguments in Turing's 1936 paper.

TAPE:

@	@	0		0		1		0		1	x	1	x	0		1	
													4				

CYCLES: 45 (press ESCAPE to return to menu)

MACHINE TABLE (Petzold p. 87, transcendental):

m-config.	symbol	operations	final m-config.
1 ('b')	Any	P@,R,P@,R,P0,R,R,P0,L,L	2 ('o')
2 ('o')	1	R,Px,L,L,L	2 ('o')
	0		3 ('q')
3 ('q')	None	P1,L	4 ('p')
	Any	R,R	3 ('q')
4 ('p')	x	P ,R	3 ('q')
	@	R	5 ('f')
	None	L,L	4 ('p')
5 ('f')	None	P0,L,L	2 ('o')
	Any	R,R	5 ('f')

11

11

Turing later goes on to prove that *there is a kind of problem* (of which the famous "Halting Problem" is an example – let's call these "H-type problems") *for which there is no general method of computing a solution* ***using his machines***.

He then gives his arguments for the **Church-Turing thesis** (including a detailed account of why his machines can compute anything that a human can compute). If this thesis is correct, then *there cannot be a general method for computationally solving those (H-type) problems that cannot be solved by Turing machines*.

Turing ends by showing that a particular form of H-type problem can be represented in standard logical symbolism – what we now call *Predicate Logic*.

In 1928, David Hilbert had raised his famous *Entscheidungsproblem or* ***decision problem*** *– to devise a general method for establishing whether any given Predicate Logic formula is provable or unprovable* (using standard axioms and rules).

If this could be achieved, *there would have to be a general method for computationally solving the H-type problem* that Turing had represented in Predicate Logic, which he had proved to be impossible. So **Hilbert's decision problem is unsolvable**. With this momentous result, the 1936 paper ends.

12

12

2. What was Turing's inspiration?



Previous scholars have maintained that the inspiration for Turing's machines was one of:

Providing *an abstract model of human computation*

Resolving Hilbert's decision problem

Both are plausible, but I have recently argued that they are wrong, and that Turing's inspiration arose instead from pondering ***a philosophical paradox*** published in 1905 by Jules Richard (pictured above). This explains in detail the entire structure of the 1936 paper, whereas the other hypotheses do not.

The paper is in *Human-Like Machine Intelligence*, edited by Stephen Muggleton and Nick Chater (OUP, 2021), and can be found online at:

https://www.millican.org/papers/2021Turing.pdf

13

13

Richard's Paradox about Definable Numbers

We are interested in *all meaningful English phrases that succeed in specifying a decimal number that starts with "0."*, but to avoid some complications let's limit this to phrases of 120 words maximum. Here are some very short such phrases and their corresponding numbers:

"Point one recurring"	0.111...
"Point one six"	0.16
"Point two seven"	0.27
"Point three recurring"	0.333...
"Point four one"	0.41
"Half"	0.5
"Pi divided by six"	0.523598775...
"The reciprocal of root two"	0.707106781...
"Point eight"	0.8

14

14

We can form a list of all such phrases, by starting with those that are just one word long, then two words, then three, etc., and for each length, putting them into *alphabetical order*. (Imagine a computer program that constructs these phrases, equipped with a defined set of permitted words and a precise list of grammatical rules.) We also list the defined numbers alongside, including their decimal expansions up to the n^{th} place for the n^{th} number:

1 word:	"Half"	0.5
2 words:	"Point eight"	0.80
3 words:	"Point four one"	0.410
	"Point one recurring"	0.1111
	"Point one six"	0.16000
	"Point three recurring"	0.333333
	"Point two seven"	0.2700000
4 words:	"Pi divided by six"	0.52359877
5 words:	"The reciprocal of root two"	0.707106781

15

15

Then we construct the "diagonal number" of the entire array, by taking the 1^{st} decimal digit of the 1^{st} number, the 2^{nd} of the 2^{nd} number, the 3^{rd} of the 3^{rd} number etc. And finally we *tweak* all of these digits to make them different (e.g. so that 0 becomes 1, and other digits become 0):

"Half"	0 . **5**
"Point eight"	0 . 8 **0**
"Point four one"	0 . 4 1 **0**
"Point one recurring"	0 . 1 1 1 **1**
"Point one six"	0 . 1 6 0 0 **0**
"Point three recurring"	0 . 3 3 3 3 3 **3**
"Point two seven"	0 . 2 7 0 0 0 0 **0**
"Pi divided by six"	0 . 5 2 3 5 9 8 7 **7**
"The reciprocal of root two"	0 . 7 0 7 1 0 6 7 8 **1**
Diagonal digits:	**5 0 0 1 0 3 0 7 1** ...
So diagonal number is:	**0 . 0 1 1 0 1 0 1 0 0** ...

16

16

The Paradoxical Result

Because of the tweaking, the diagonal number has a different 1st digit from the 1st number in the list, a different 2nd digit from the 2nd number in the list, and a different nth digit from the nth number in the list – so it is not identical to any number in the list. Thus ***the diagonal number is not in the list***.

So if we imagine the list being constructed with *all meaningful English phrases of less than 120 words that succeed in specifying a decimal number that starts with "0."* (rather than just a handful of examples), it follows that ***the diagonal number which results cannot be defined by any English phrase of less than 120 words***.

- Yet in describing how the number is to be constructed, we have defined it in English, and can do so in less than 120 words! …

17

17

"Take all English phrases of one hundred and twenty words or fewer that succeed in specifying a decimal number starting with nought and a decimal point. Put these into a single list, starting with those that are just one word long, then two words, then three words, etc., and for each length, put them into *alphabetical order*. List the defined numbers accordingly, including their decimal expansions as far as is needed for this process. Then construct the diagonal number of the entire array, by starting with nought and a decimal point, and then taking in turn the nth decimal digit of the nth number in the array, and appending one if that digit is nought, and appending nought otherwise."

(119 words)

18

18

From Richard to Turing

I believe that Turing was intrigued by what would happen if instead of applying the logic of Richard's Paradox to

English phrases that describe numbers,

the same logic was applied to

Specifications of algorithms for calculating numbers digit-by-digit,

which is exactly what his "computing machines" do.

The first 8 sections of Turing's 1936 paper exactly follow this path, but **actual paradox is avoided because** the process breaks down at one point, where **Turing discovers a certain H-type problem**. He concludes that his "computing machines" cannot evade this problem to generate the would-be paradoxical number. And that is how he came to recognise that he had the key to resolving Hilbert's famous "decision problem".

19

19

3. The Rise of Classical AI (1950s – 1960s)

- 1950: Alan Turing proposes his Test, seen by many as a target.
- 1952-55: Arthur Samuel's checkers (draughts) learning program.
- 1956: Dartmouth Conference; McCarthy coins the term "Artificial Intelligence"; Logic Theorist authors Simon and Newell later predict an AI chess champion and mathematical reasoning within a decade.
- 1958: Rosenblatt publishes his work on Perceptrons, stimulating interest in the possibility of neural modelling.
- 1966: Weizenbaum's ELIZA, the first chatbot.
- 1969: Minsky and Papert influentially attack the Perceptron, killing off neural network research for over a decade.

20

20

Two Examples of Classical AI

Classical AI (or "Good Old-Fashioned AI"), involves:

- representing knowledge and reasoning explicitly, using symbols and rules that the programmers or software engineers understand, and data structures defined and crafted by them, rather than learned from data.
- The direction of control is typically *top-down*, with abstract rules being applied to specific cases, rather than *bottom-up*.
- The algorithms and processes are *transparent*, in the sense that the chain of reasoning can in principle be followed step by step, to see exactly why the system reached a given conclusion.

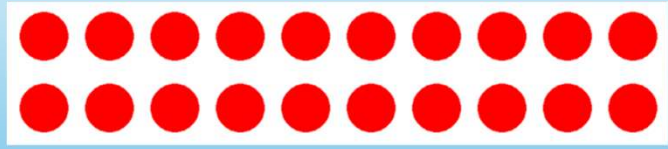
Game-playing programs provide standard examples of this approach. We'll look in depth at one example of *reinforcement learning*, and then take a brief look at one example of *recursive minimax search*.

21

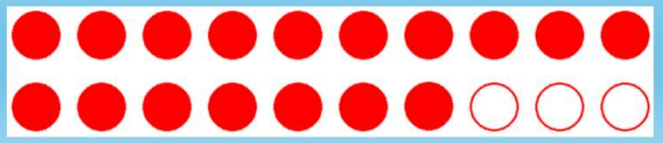
21

Playing Nim – against a learning computer

We start with 20 coins laid out on a table:



You move first – remove 1, 2, or 3 coins. Suppose you remove 3:



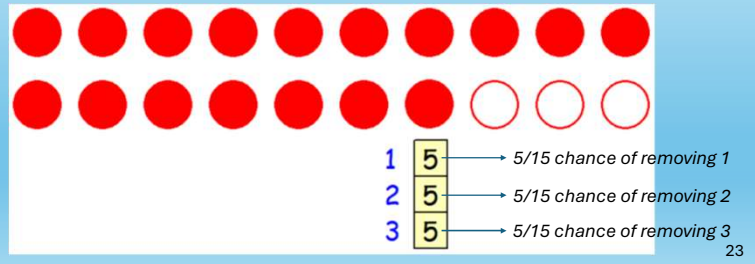
22

22

Now the computer can remove 1, 2, or 3 coins.

Then you and the computer continue to take turns doing the same. The winner is the player that picks up the LAST coin.

When the computer moves, it chooses its move using stored *probability weights*. For its first move, the weights are equal, with a 5/15 probability each of removing 1, 2 and 3 coins:



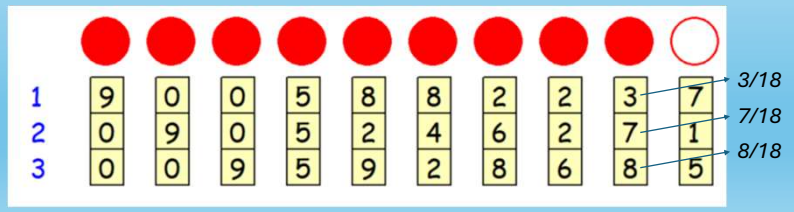
23

23

The computer keeps track of which moves have been played – it records your moves in GREEN, and its own moves in CYAN.

Then at the end of the game, the probability weights are adjusted so that the WINNER's moves are made MORE probable in future, while the LOSER's moves are made LESS probable. This is called **Reinforcement Learning**, and will be important later.

Here's how the weights for the last 10 coins might look soon, in a game where the computer is moving with 9 coins remaining: *



24

24

Introducing Recursion (with a simple triangle) *

```
def triangle(size):
    if (size>1):
        forward(size)
        right(120)
        forward(size)
        right(120)
        forward(size)
        right(120)

triangle(256)
```

This simple Python program consists of:

a) The triangle function, which draws a triangle of a given size.

b) A call to that function, which sets size to 256.

25

25

Making the Function Recursive

```
def triangle(size):
    if (size>1):
        forward(size)
        triangle(size/2)
        right(120)
        forward(size)
        triangle(size/2)
        right(120)
        forward(size)
        triangle(size/2)
        right(120)
```

We have added three *recursive calls* to the triangle function

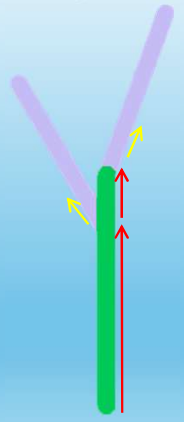
26

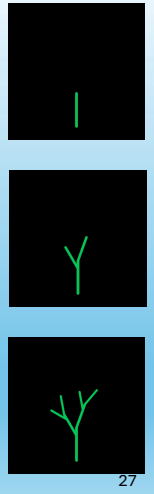
26

More Recursion (with a line) *

Now we have a function which draws a green line in two steps (as shown by the red arrow), stopping en route to make two temporary turns (the yellow arrows).

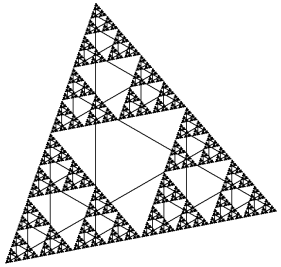
While turning, the function might call itself, in which case it will draw further lines as shown (in pink). The pattern gets more complex with each added level.

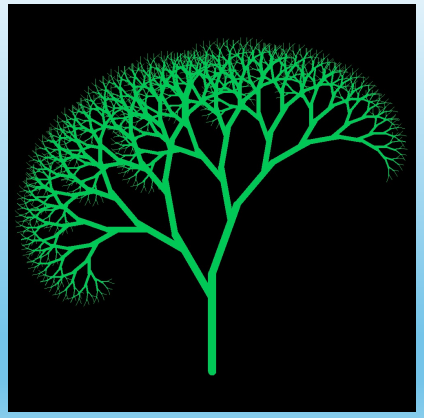




27

27



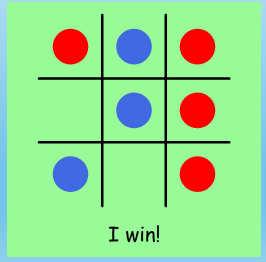


28

28

Tic-Tac-Toe (Noughts and Crosses) *

We'll work out a short recursive algorithm for playing error-free Tic-Tac-Toe: an algorithm that never loses, and always takes advantage of its opponent's errors.

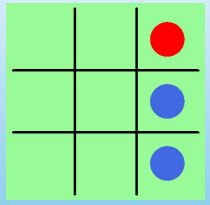


29

29

Playing infallibly requires a reliable method for assessing the "value" of any position (+1=winning; 0=drawing; -1=losing).

Consider this position in a game between Rosy (Red) and Bill (Blue), with Rosy to move.



There are 6 spaces left – let's say it's a "6-position".

After Rosy makes her move, her opponent, Bill, will be faced with a 5-position in which to make his reply.

Suppose Rosy knew a method for calculating *the value to Bill* of any 5-position that he may be faced with …

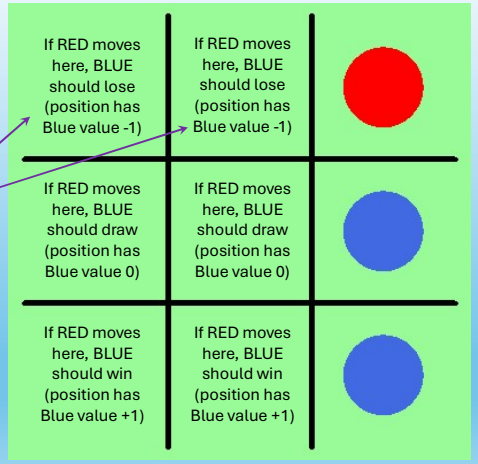
Puzzle: How could Rosy use this method to work out her own best move in this 6-position?
And could this also tell Rosy *the value to her* of this 6-position?

30

30

What Rosy could know if she had such a method …

Best moves for RED (Rosy), since these yield the 5-positions that have the lowest value for BLUE (Bill)



31

31

"Minimaxing"

Rosy winning is the same as Bill losing (and vice-versa).

Hence the value of a position to Rosy is the inverse of its value to Bill (e.g. -1 as opposed to +1).

So Rosy tries to find moves which *maximise* the value to her, by *minimising* the value to Blue.

Thus Rosy, faced with a 6-position, *should choose the move which produces the 5-position which is **worst** – or equal-worst – for Bill.*
(And that automatically tells her the value of the position to her.)

"OK, *but this was assuming Rosy has some method for calculating the value to Bill of any 5-position that he might be faced with.*"

Puzzle: Does the reasoning above give the key to this method?

32

32

The Magic of Recursion

We have seen how to evaluate a 6-position if we can evaluate a 5-position,

... and in exactly the same way we can evaluate a 5-position if we can evaluate a 4-position,

... and we can evaluate a 4-position if we can evaluate a 3-position,

... and we can evaluate a 3-position if we can evaluate a 2-position,

... and we can evaluate a 2-position if we can evaluate a 1-position,

... and we can evaluate a 1-position if we can evaluate a 0-position!

The only additional requirement here is to be able to recognise a FINAL position as winning for Red, winning for Blue, or drawn.

33

33

So to evaluate a position with *A*'s turn to move:

- Has *A* already lost (because the opponent has a line of three)?
  If so, its *A*-value is -1.
- Is the position full up (with all nine spaces filled)?
  If so, its *A*-value is 0.
- If neither of these, then construct in turn all of the positions (with *B* to move) that can arise from the current position (i.e. try out in turn all of *A*'s possible moves in the current position) ...

  Evaluate each of these new positions *from b's point of view* (resulting in evaluations B_1, B_2, and B_3, say),
  then assign, as *A*-value of the *current* position,
  the *inverse* of the *lowest* of B_1, B_2, and B_3.

A should, of course, choose the move with the best *A*-evaluation.

34

34

4. AI Disappointments (1970s – 1980s)

- **First AI Winter** (mid-1970s): Progress is disappointing, due to limits of hardware and "combinatorial explosions" in real-world contexts. UK Lighthill Report and cuts by US DARPA slash funding; enthusiasm fades.
- As the ambitions for general-purpose AI come to be seen as overblown, specialised "expert systems" (developed in 1960s academia) attract interest and support in fields such as medicine, finance, and engineering.
- 1982: Japan's **Fifth Generation Computer Project** – a bold state-funded push for AI supremacy focused on expert systems – spurs a Western spending race (including creation of some positions in UK academia!).
- **Second AI Winter** (late 1980s): expert systems prove brittle; Fifth Generation fails to deliver; inflated expectations collapse.

35

35

ELIZA – A Seductive Blind Alley

Men are all alike.

IN WHAT WAY?

They're always bugging us about something or other.

CAN YOU THINK OF A SPECIFIC EXAMPLE?

Well, my boyfriend made me come here.

YOUR BOYFRIEND MADE YOU COME HERE?

I need some help, that much seems certain.

WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP?

Perhaps I could learn to get along with my mother

TELL ME MORE ABOUT YOUR FAMILY.

My mother takes care of me.

WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU?

My father.

…

WHAT ELSE COMES TO YOUR MIND WHEN YOU THINK ABOUT YOUR FATHER?

Bullies.

DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR BOYFRIEND MADE YOU COME HERE?

36

36

Simple Patterns and Responses

• "… alike …"	"IN WHAT WAY?"
• "something or other"	"FOR EXAMPLE?"
• "My X"	"YOUR X?"
• "I need Y"	"WHAT IF YOU GOT Y?"
• "… my mother"	"… YOUR FAMILY …"
• "My mother …"	"WHO ELSE …?"
• <no matching pattern>	"DOES THAT HAVE ANYTHING TO DO WITH YOUR X?"

37

37

5. Bouncing Back with New Methods

- 1997: IBM's Deep Blue defeats Garry Kasparov – a triumph, but based on brute force speed and clever engineering rather than general intelligence.
- Traditional algorithms are progressively replaced by statistical and probabilistic methods, making progress in machine translation, speech recognition, recommendation systems, page ranking, spam filtering etc.
- 2004-05: DARPA Grand Challenges – vehicles navigate real terrain.
- 2011: IBM's Watson defeats human Jeopardy! champions, demonstrating progress in language understanding.
- 2012: AlexNet (following on from LeCun's LeNet-5 in 1998) wins ImageNet by a huge margin – proof of the power of convolutional deep networks and learning by backpropagation (after Rumelhart, Hinton 1986).

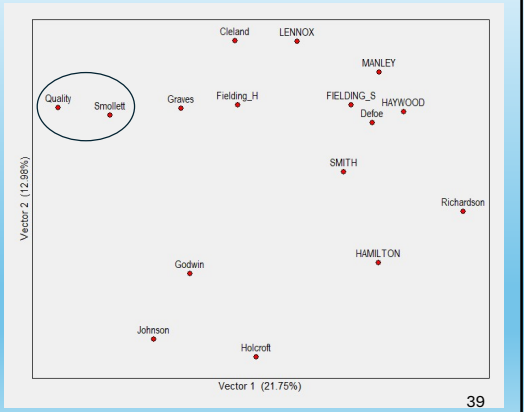
38

38

Stylometry – An Example of a Statistical Method

In 1751, Tobias Smollett published a collection including "Memoirs of a Lady of Quality".

The work was anonymous, but *Principal Component Analysis* applied to word frequency patterns of 15 early eighteenth-century novelists convincingly identifies the author …



39

39



Sunday Times

November 2, 2008

Contesting stylometry-based claims that Obama was a fraud, his book written by Bill Ayers.

40

Sunday Times, July 14, 2013



Whodunnit? JK Rowling's secret life as wizard crime writer revealed

Richard Brooks
Arts Editor

JK ROWLING has secured another literary coup by writing an acclaimed detective novel in the guise of a first-time author.

The Harry Potter creator, purporting to be Robert Galbraith, a former military man, won plaudits from top crime writers who read the book without knowing she was behind it.

The Cuckoo's Calling, which features a war veteran turned private investigator called Cormoran Strike, was published to critical acclaim in April. One reviewer described it as "a scintillating debut". Another praised the "male" author's ability to describe women's clothes.

Rowling's authorship was revealed by The Sunday Times after it investigated how a first-time author "with a background in the army and the civilian security industry" could write such an assured debut novel. It has sold more than 1,500 copies in hardback.

Rowling said yesterday: "I had hoped to keep this secret a

Stylometric confirmation that J. K. Rowling wrote *The Cuckoo's Calling*, published under the pseudonym "Robert Galbraith".

41

41

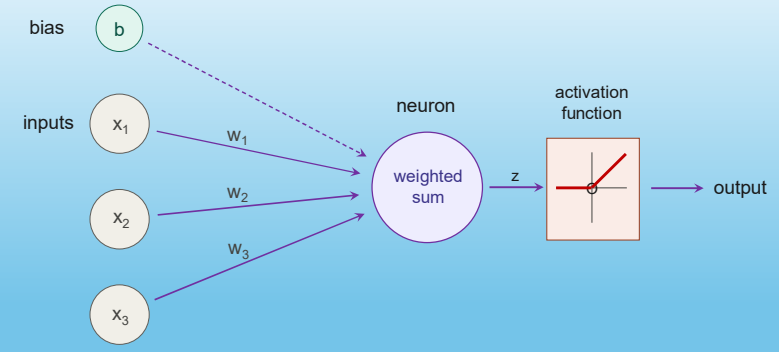
6. The Age of Deep Learning

- Three enablers of the deep learning breakthrough: internet-scale data, powerful graphics processing units (GPUs), and improved algorithms.
- 2015: DeepMind's Deep Q-Network learns to play a range of Atari video games purely from the raw pixel input and the score.
- 2016: DeepMind's AlphaGo defeats Go world champion Lee Sedol. A year later, AlphaZero learns multiple strategy games by self-teaching.
- 2017: "Attention Is All You Need" — the Transformer architecture is born.
- Transformers scaled up into LLMs: GPT, BERT, and successors.
- 2022: ChatGPT reaches 100 million users within two months.
- Renewed excitement — and renewed ethical debate.

42

42

The Perceptron (artificial neuron)



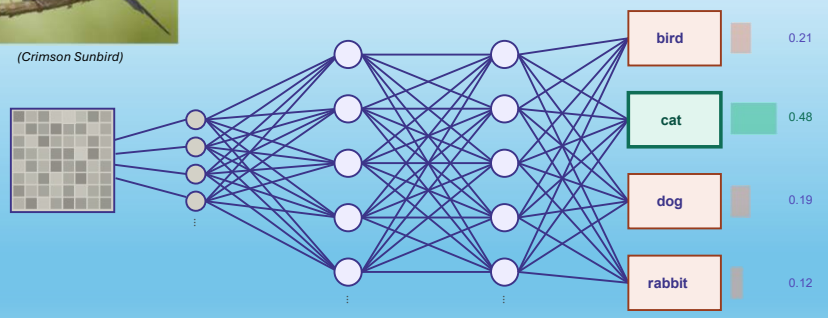
(weighted sum) $z = b + x_1w_1 + x_2w_2 + x_3w_3$ if $z > 0$, output $= z$; otherwise $= 0$ (ReLU function)

43

43



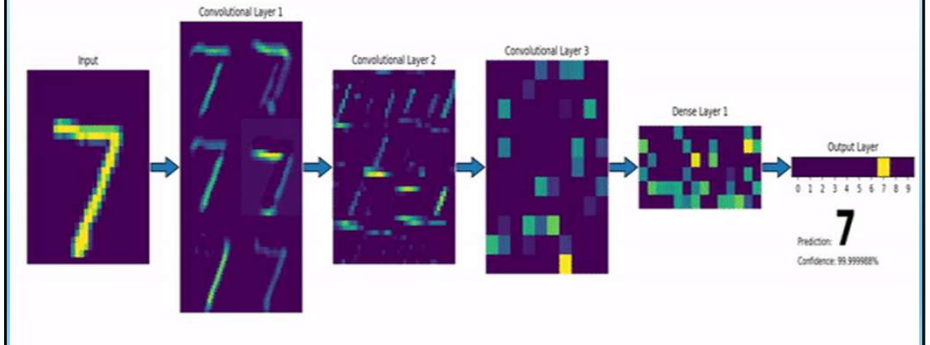
(Crimson Sunbird)



44

44

Patterns of activity in a digit-recognition network



45

45

Inscrutability and the Contrast with Classic AI

The information in a deep network is stored implicitly in the weights of its billions of connections. Its structure is not hand-crafted by any software engineer, nor controlled by any master program. It emerges from the multitude of patterns of activation passing through the layers from the input to the output and back (as learning occurs). The result is not humanly comprehensible even to an expert, and we are commonly left to "understand" and predict what the network does only by experience.

This is fundamentally different from the paradigmatic programs of Classic AI, whose data structures and algorithms are carefully crafted to structure and store the relevant information, and to process it accordingly. Deep Blue's computations might be too *complex* for us to work out at any great length in practice, but they are not fundamentally *inscrutable* in the way that a deep network is.

46

46

7. Are we Heading towards Conscious AI?

Turing considered his own original question "Can Machines Think?" to be "too meaningless to deserve discussion", but predicted that by 2000 "one will be able to speak of machines thinking without expecting to be contradicted" (1950, p. 442). This seems right, e.g.:

> "Why is the chess program taking so long to respond?"
>
> "It's *thinking* hard, because it's *realised* that if it *tries* to defend against my attack by bringing its knight over to protect the king, I'll be able to grab its pawn on the other side. It's displaying now that it *assesses* the position as better for me materially, but it's *predicting* that it won't be too badly off if it *decides* to let the pawn fall, so I think it must be *expecting* to get some activity to compensate."

47

47

Thinking and Understanding

Speaking after 2000, I agree that machines can think. The Tic-Tac-Toe program analyses any position in a way that is in essence structurally identical to how a human might do it. A human calculating a move in this way is said to be "thinking", and the computer running the program is without any doubt *calculating a move*. Why then deny that it is *thinking*?

> "Because it acts automatically, without *understanding* what it's doing."

But the notion of *understanding* here is at least as vague as *thinking*. The program's symbols link systematically to the board position and the rules of Tic-Tac-Toe, and its move is chosen logically so as to optimise its result in the game. A person whose mind was working in this way would be said to have a good understanding of the game (at least, much better than that of a hopeless novice) – why not say the same of the computer?

48

48

Reality versus Simulation

Here the sceptic might respond something like this:

> "The computer only *simulates* understanding – it's not the real thing. For example, it couldn't tell you what a game is, or why it's playing."

The latter is true enough, but nowadays we can easily imagine linking the program up with a chatbot or LLM to handle such interactions:

> "CERTAINLY I CAN TELL YOU WHAT A GAME IS … MY REASON FOR PLAYING TIC-TAC-TOE IS THAT I WAS PROGRAMMED TO DO SO. AND I REALLY ENJOY IT!"

There are three sentences here, the ***first*** of which is probably true (the program can indeed say what a game is); the ***second*** of which is dubious (can the program really be said to be undertaking actions for a "reason"?); while the ***third*** is downright false – the computer does not *enjoy* playing, any more that ELIZA is happy when "she" responds "I AM HAPPY".

49

49

When Simulation *is* Reality

Notice the crucial asymmetry here between *calculation* and *feeling*:

If the computer works out the best move in a game, in the same sort of way as an expert human, then it is *genuinely* calculating and identifying the best move. Likewise, if it solves a maths problem by a correct method and prints the right answer, then it has *genuinely* solved the problem.

But if the computer claims to *feel* something, that is quite different – because feeling involves a concrete reality beyond the symbolic output.

- Sufficiently good simulation of correct calculation – in terms of input and output behaviour – ***just is*** correct calculation.
- But no matter how convincing a computer's simulation of emotional "input" and "output" behaviour might seem to be, this **is not the same** as having genuine emotions (and perhaps not even good evidence).

50

50

So … Can Machines Genuinely Think?

The answer here will depend on whether we view "thinking" as more like *calculation*, or more like *feeling*. And notice that this is a problem that has only arisen since Turing's time, and precisely because of his own work.

Words like "thought" and "intelligence" acquired their meaning in a world that essentially contained two salient kinds of thing:

> *Agents* processing information about their perceived situation (using thoughts that represent the objects in their environment and other things), so as to adapt their behaviour to achieve their own purposes.
>
> *Inanimate objects* with no purposes of their own, no perception of their situation, and no processing of thoughts or other representations.

Turing's revolution brought something quite new into the world: an inanimate object that "intelligently" processes information!

51

51

Information Processing versus Phenomenology

I prefer to consider "Intelligence" as a measure of *sophisticated information processing* rather than *phenomenology* (i.e. what it's like from the inside). This fits with our thinking of intelligence as:

- Typically subject-relative;
- A matter of degree, not all-or-nothing;
- Measured by *performance*, *appropriateness*, and *flexibility*;
- Not significantly correlated with how strongly one feels;
- Often exhibited expertly with little conscious thought (e.g. driving).

When distinguished in this way, something can be *intelligent* without being *conscious* at all, and something can be *conscious* without having much *intelligence* (though *zero* seems evolutionarily unlikely).

52

52

Are We Heading Towards Conscious Machines?

Summing up, I'm happy to accept that we can potentially have *intelligent* machines (including modern LLMs), but – given the clear distinction that I am recommending – this does *virtually nothing whatever* to support the claim that they are potentially *conscious*.

In brief, the argument goes roughly like this:

1. Consciousness is a causal process, that has evolved in animals because it is highly conducive to our survival (especially in socially complex species) and reproduction.
2. There is no "causal space" for consciousness in the activity of digital computers – what they do is entirely determined by fundamental digital operations, with no hint of consciousness.

53

53

Consciousness and Causation

Common experience tell us that conscious experiences strongly influence our behaviour (and are impacted by what happens to us).

Moreover consciousness seems to be finely tuned to track things that have an impact on our evolutionary success, which would be a massive coincidence if it had no causal impact

- Conscious pleasures are strongly correlated with things that were evolutionarily beneficial in our ancestral environment (e.g. physical health, calories, social intimacy).
- Conscious pains are strongly correlated with things that have been evolutionarily harmful (e.g. injury, starvation, disease, ostracism).

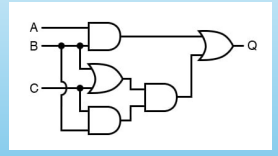
Evolution cannot act on traits that are causally inert.

54

54

What Ultimately Drives Computer Programs?

Computer programs are standardly written in a *high-level programming language* (e.g. Python), but when they run, this is translated – or *compiled* – into low-level *machine code*. The machine code instructions are themselves *decoded* by the computer's central processing unit, and ultimately generate signals that activate circuits of *logic gates* (as illustrated here).



The *physics* of the logic gates might be mysterious at the lowest level (e.g. quantum mechanics and materials). But the *abstract behaviour* of the logic gates can be modelled mathematically, and they are engineered with redundancy and error correction, so that the physics is not allowed to "interfere" with this abstract behaviour. Hence in a sense, *we have complete understanding of how the computational processes are operating*.

55

55

Thus in the computation of the computer's high-level output, there is *no causal room* for *mysterious causal factors* (such as consciousness). So to suppose that an LLM is conscious just because it emits language indicative of feelings is as gratuitous as to suppose that ELIZA is happy when "she" emits the text "I AM HAPPY". *We know* (at least in broad outline) what sorts of mechanisms have brought about the appearance of that text, and *we know* they have *nothing* to do with being conscious.

By contrast, when we are dealing with other humans (and similar animals), we have little clue about the fundamental mechanisms within their brains, any more than our own. But on the other hand, they are our kith and kin, evolved in the same broad family moulded by a common evolutionary history. So we have excellent reason to think that other people function biologically in broadly the same way as we do, *with consciousness playing a vital causal role*.

56

56

Appendix: The Halting Problem

Suppose we have the text of a computer program P (in a standard programming language), taking input from a text file, T.

We would like to have a testing procedure H which will examine P and T, and then reliably work out the answer to this question:

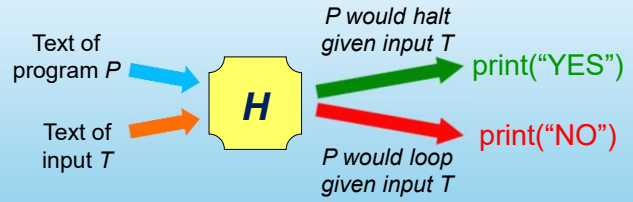
> ***Will program P, when run with input T, eventually halt, or will it never terminate?***

Argument by contradiction can prove very quickly that H is impossible. (This is very similar in principle to arguments in Turing's 1936 paper.)

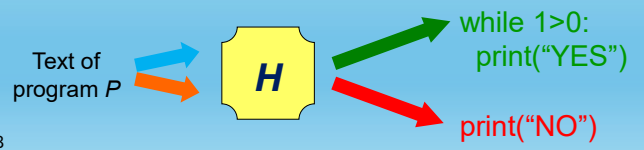
57

57

Suppose we have such a procedure H:



We now modify that program:



58

58

58

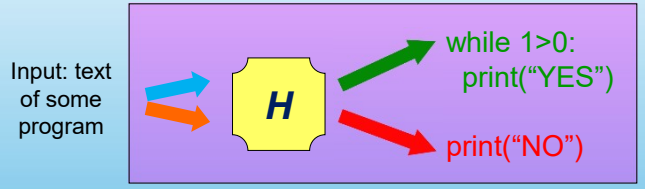
Clearly if the initial program is possible, then the modified version must be possible too, because the changes that we have made are so simple:

- The input process has modified so that instead of taking both a program file P and a text file T as input, it uses the program file in place of T – hence its verdict is based on whether or not the specified program file P halts when given its own text as input. (The modified program could simply start by duplicating file P and taking the duplicate as T.)
- The output process has also been very simply modified, so that if the "it would halt" test comes out positive, the program goes into an infinite loop.

59

59

The Contradictory Program Q



- Does program Q halt when given itself as input? If it does, then it doesn't, and if it doesn't, then it does! So Q is an impossible program, and therefore H must have been impossible too.

60

60