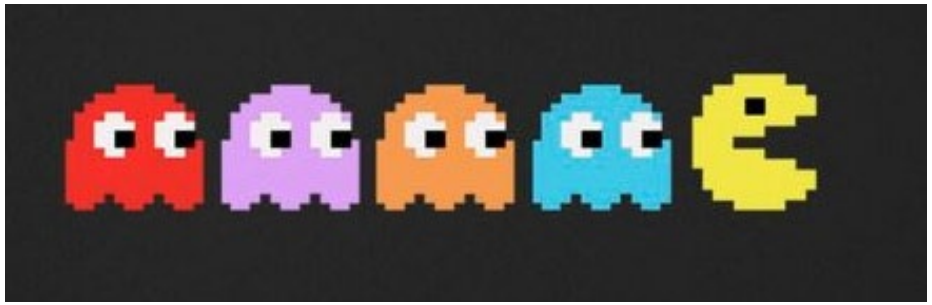
Engaging KS3/4 students with AI and Python Code

Mark Weddell
City of London School for Girls

This presentation is available today at
www.weddell.co.uk

My first AI experiences



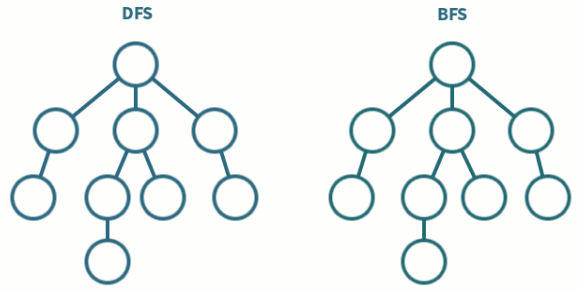
Early Teaching AI experiences

Gather, Train, Use…

Machine Learning for Kids

Teachable Machine

AQA 4.3 - Graph Traversal and Optimisation Algorithms



Inspiration – CAS Conference 2024

```
from nltk import bigrams, trigrams
from collections import Counter, defaultdict

# Create a placeholder for model
model = defaultdict(lambda: defaultdict(lambda: 0))

# Count frequency of co-occurance
for sentence in text:
    for w1, w2, w3 in trigrams(sentence, pad_right=True, pad_left=True):
        model[(w1, w2)][w3] += 1


# Let's transform the counts to probabilities
for w1_w2 in model:
    total_count = float(sum(model[w1_w2].values()))
    for w3 in model[w1_w2]:
        model[w1_w2][w3] /= total_count
```

Miles Berry - 3 July 2024 AI and Computing Education, workshop for Computing At School conference, ADA College.

Why with KS3/4?

Most AI experiences remove the coded complexity of “reasoning” from students.

Algorithms for GCSE are fairly basic and seem far removed from equipping …

“… pupils to use computational thinking and creativity to understand and change the world.”

UK Teacher Standards encourage us to …

“Set high expectations which inspire, motivate and challenge pupils”

Your setting…

My Three Examples

- Algorithmic AI - Rule-based Systems

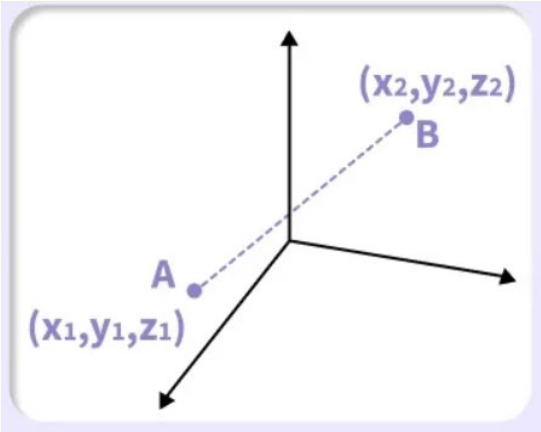
- Data Driven AI – Human in the Loop (HITL)

- Data Driven AI - Automated Playtesting

Colour Matching – Near Neighbour - Algorithmic AI

Distance AB = $\sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2}$

The difference between two colours could be calculated (represented) using the change in R, G and B values.



https://trinket.io/pygame/cace981cbd91

The Code

A ‘minimum’ in list function
combined with 3D Pythagoras calculation

```
def checkColourProximity(red,green,blue):
  #Initial values
  colourProximity = 1000
  closestColourPosition = 0

  #Check against all 16 colours
  for index in range(16):

    # Calculate proximity
    thisProx = ((red-colourPalette[index][0])**2 + (green-colourPalette[index][1])**2 + (blue-colourPalette[index][2])**2) **0.5

    #Is this closer in colour
    if thisProx < colourProximity:
      colourProximity = thisProx
      closestColourPosition = index

  return closestColourPosition
```

Check each colour in the list.
Work out proximity of this colour.
If the proximity is less than current colourProximity…
Set new colourProximity and closestColourPosition

https://trinket.io/pygame/cace981cbd91

Tasks

Ideas…

- As they are lists in Python, you can add a name as a string to each colour list.

  Print the name of the nearest colour.

- Find the nearest two colours, so the user can choose the closest.

- Imagine the numbers represent other values on a scale, such as musical tastes, personality traits, etc.

  How could you adapt the concept? Instead of 3 values, perhaps 4?

Think …

- Are there limitations with this algorithm of near neighbour algorithm?
- When would you use it, and when not?
- Could this develop to K-Near Neighbour and K –Means Clustering ?

https://trinket.io/pygame/cace981cbd91

Multiplication Tables – HITL - Machine Learning AI

The idea that a computer can learn and become more accurate as millions of users attempt to answer questions.


```
What is 12 x 9?: 108

The most common answer so far is 108
You gave the answer most frequently given to this question



2 user(s) have answered this question.
The answer(s) they gave: 108, 108,

The most common answer was given 100.0 % of the time
Confidence in this answer is low,
as few people have answered this question
```

Instead of correct answers, users might receive a reward or gain feedback about their answer...

https://trinket.io/python3/775333af4d6d

The Code

```
## Multiplication Quiz

for i in range (20):
  x = randrange(1, n+1)
  y = randrange(1, n+1)

  question = "What is " + str(x) + " x " + str(y) + ("?:")
  guess = int(input(question))


  ## Add answer given to data
  data[x-1][y-1].append(guess)

  ## Save data to file

  save_data_to_file(data)

  ## Save data to file

  giveFeedback(data, guess, x, y)
```

```
What is 12 x 9?: 108
```

```
The most common answer so far is 108
You gave the answer most frequently given to this question



2 user(s) have answered this question.
The answer(s) they gave: 108, 108,

The most common answer was given 100.0 % of the time
Confidence in this answer is low,
as few people have answered this question
```

Ask Question
Get answer from User
Add to data and save
Feedback about answers to same question

https://trinket.io/python3/775333af4d6d

Tasks

Ideas...

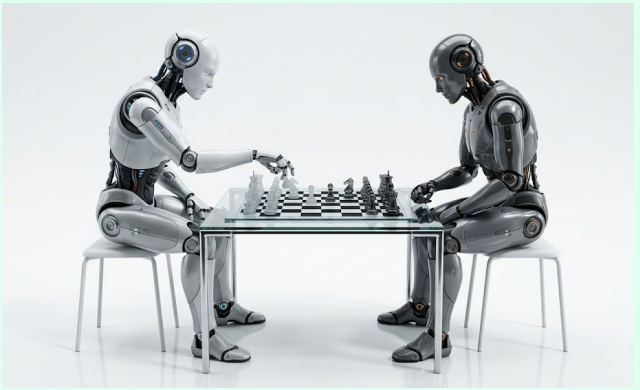
- The data is currently mostly correct.
  Investigate what happens when you get an answer wrong. How does the data change?
- Change the feedback.
- What other questions could you ask people to gather data about?
  Would this work for a country / capital quiz?

Think ...

- How could you ensure that people aren't entering malicious answers?
- What sort of data does this work for? Bias?
  Is there a difference between the data that can and cannot be collected in this way?

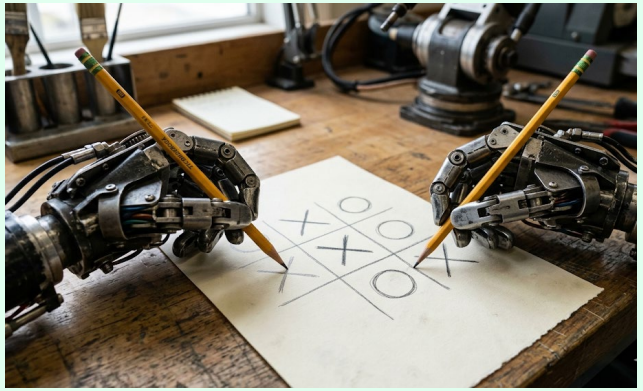
https://trinket.io/python3/775333af4d6d

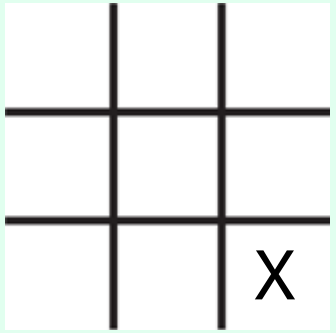
Tic Tac Toe – Computer Generated Data - Machine Learning AI

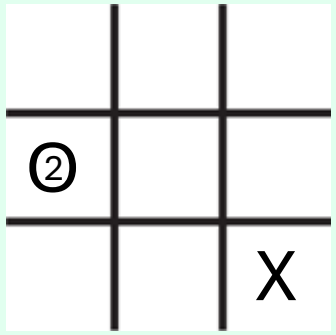


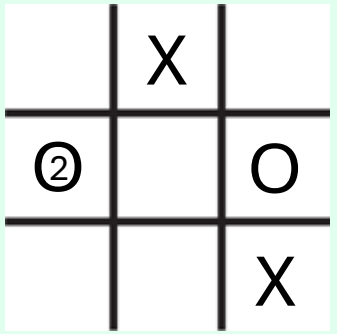
Given a set of rules as to how to play, a computer can play thousands or millions of times.

By analysing moves, can the computer predict good choices?









https://trinket.io/library/trinkets/21ac43e94c79

Tic Tac Toe – Computer Generated Data - Machine Learning AI

```
firstCell = int(input("Enter the number (0 - 8) of the cell where X went: "))

for i in range(100000):              # Play 100,000 random games of Tic Tac Toe

  game = playTicTacToe(firstCell)

  outcome = game[1]                  # outcome is "win", "loss" or "draw"
  gameBoard = game[0]                # gameBoard is a list showing order of moves

  if outcome =="win":                 # Make a list of games won by "O"
    Win.append(gameBoard)

  if outcome =="draw":                 # Make a list of drawn games
    Draw.append(gameBoard)

  if outcome =="lose":                 # Make a list of games lost by "O"
    Loss.append(gameBoard)
```

- Ask for a starting move from your opponent
- Play thousands of games from that point
- For each game…
  - Record whether ‘O’ won, lost or drew
  - Record the order of moves made
  - Add the order of moves to an array of wins, of draws or losses
- Produce statistics of how many games won and lost for each next move.

https://trinket.io/library/trinkets/21ac43e94c79

Tic Tac Toe – Computer Generated Data - Machine Learning AI

Ideas…

- Investigate different starting positions for X
- Produce the analysis of the “Loss” list to see bad places to go.
- Instead of just one first move, keep entering moves until number 9 is entered, to allow analysis from different positions.
- Improve the game playing engine. At present, moves are made completely randomly. If the algorithm could spot winning or losing moves, do the stats change?

Think …

- What sort of situations apart from games could this work for?
- What are the pitfalls of this approach?

https://trinket.io/library/trinkets/21ac43e94c79

Would you use these ideas with students?

How? When?

Why? Why not?

Is this useful?

What would you change?

Thank you

www.weddell.co.uk