

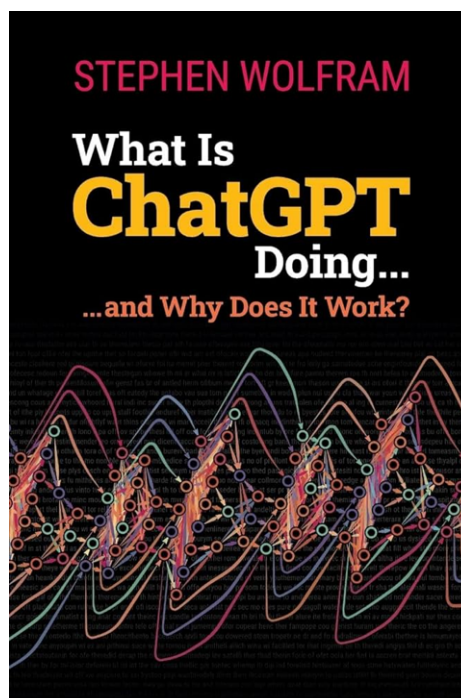
What is behind the ChatGPT?



Talk Outline

- Inspiration
- Explaining the name ChatGPT
- Synthesize Data
- Generative AI Model - virtual guest speaker Prof. Mirella Lapata-Uni. Edinburgh
- Statistics of the different GPT-i versions
- Amazing and Amusing Examples
- Final Remarks

1. Inspiration



2.Name

ChatGPT → Chat Generative Pretrained Transformer

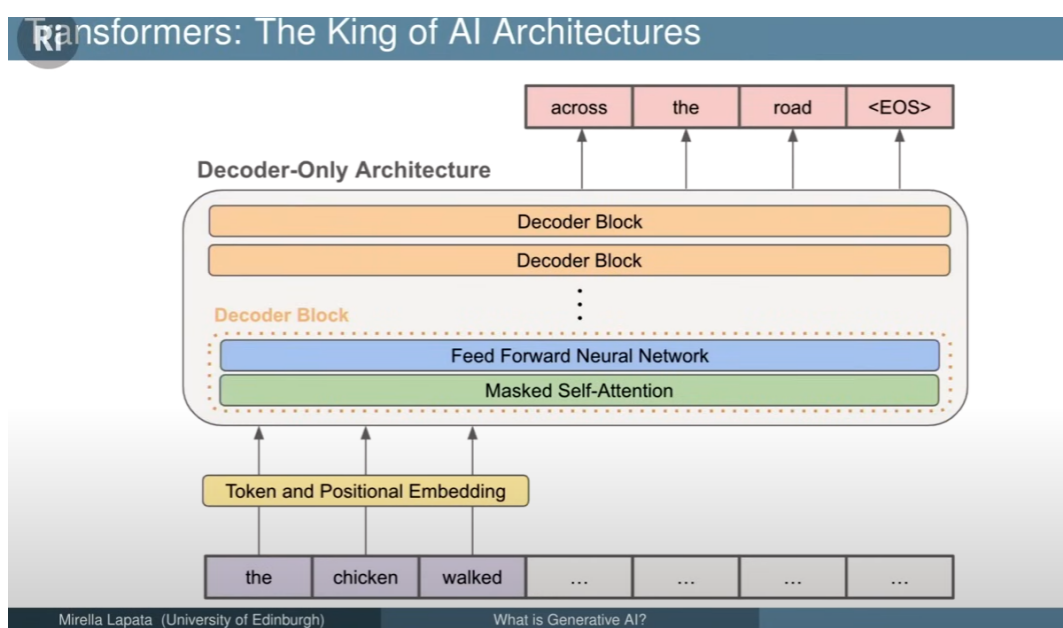
3.Synthesize Data

Synthetic data is information that is not generated by real-world occurrences but is artificially generated.

4.Generative AI Models - Large Language Model - Transformer

What is generative AI and how does it work? – The Turing Lectures with Prof. Mirella Lapata (Uni. Edinburgh)

```
In[*]:= SystemOpen["https://www.youtube.com/watch?v=_6R7Ym6Vy_I"]
```



5.Statistics of the different GPT-i versions

OpenAI's "GPT-n" series

Model	Architecture	Parameter count	Training data	Release date	Training cost
GPT-1	12-level, 12-headed Transformer decoder (no encoder), followed by linear-softmax.	117 million	BookCorpus: ^[27] 4.5 GB of text, from 7000 unpublished books of various genres.	June 11, 2018 ^[8]	30 days on 8 P600 GPUs, or 1 petaFLOP/s-day. ^[8]
GPT-2	GPT-1, but with modified normalization	1.5 billion	WebText: 40 GB of text, 8 million documents, from 45 million webpages upvoted on Reddit.	February 14, 2019 (initial/limited version) and November 5, 2019 (full version) ^[28]	"tens of petaflop/s-day", ^[29] or 1.5e21 FLOP. ^[30]
GPT-3	GPT-2, but with modification to allow larger scaling	175 billion ^[31]	499 billion tokens consisting of CommonCrawl (570 GB), WebText, English Wikipedia, and two books corpora (Books1 and Books2).	May 28, 2020 ^[29]	3640 petaflop/s-day (Table D.1 ^[29]), or 3.1e23 FLOP. ^[30]
GPT-3.5	Undisclosed	175 billion ^[31]	Undisclosed	March 15, 2022	Undisclosed
GPT-4	Also trained with both text prediction and RLHF; accepts both text and images as input. Further details are not public. ^[26]	Undisclosed. Estimated 1.7 trillion ^[32]	Undisclosed	March 14, 2023	Undisclosed. Estimated 2.1e25 FLOP. ^[30]

6. Amazing and Amusing Examples

Verses

```
In[*]:= LLMSynthesize["Can you write a verse about Budapest and write it in Hungarian"]
```

```
Out[*]=
```

Budapest, gyönyörű város,
Hol lángol a látnivalók sora.
A Duna folyónak ölében terül el,
Szívdobbanásomra megfoghatatlan belső nyugalommal.

A város színében úszó hidak,
Elvarázsolnak, mint a varázsszók.
A Szent István Bazilika méltósága,
Belém égetve marad, örökre hálával.

A Hősök tere büszkeségével,
Magyar történelem szobra üzen.
A Várnegyed rejtett történelmi kincsei,
A múltba vezetnek, álmodozó lelkemben.

Budapest, a Károlyi-palota meséje,
Egykor nemes urak otthona volt régebben.
A Széchenyi fürdő pihentető vize,
Testem-lelkem megmelengeti minden szikrája.

Az Erzsébet híd lengő pillérei között,
Sétálva érzem, milyen végtelen boldogságot hoz.
A Parlament igézően ragyog a folyón,
Magammal viszem haza emlékeim csodáját.

Budapest, te lélegzetelállító város,
Szívembe zártalak, örökre hálával és vágyakozással.
Ha messzeségben járok, szívem mindig hozzád vonz,
Hazatérve újra megszólal a dallam, a Budapest dalának hangján.

```
In[*]:= LLMSynthesize["Write a haiku about Lionel Messi"]
```

```
Out[*]=
```

Graceful on the field
Lionel, a football king
Goals and dreams he weaves

Joke

```
LLMSynthesize["Tell me a joke about a drunkard man and explain me why it is a joke"]
```

Out[]:=

Why did the drunkard man bring a ladder to the bar?

Because he heard the drinks were on the house!

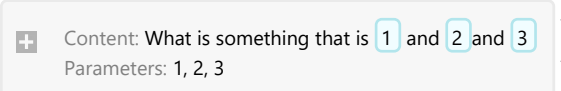
Explanation:

This joke relies on the play on words. In this context, the phrase "on the house" usually means that the drinks are free of charge. However, the drunkard man misunderstands it literally and assumes that the drinks were located on top of the building, so he brings a ladder to reach them. The humor lies in the absurdity of his interpretation and his action.

Riddle

```
In[ ]:= FindbyProperty = LLMFunction["What is something that is `1` and `2` and `3`"]
```

Out[]:=

```
LLMFunction[]
```

Content: What is something that is 1 and 2 and 3
Parameters: 1, 2, 3

```
In[ ]:= FindbyProperty["Can not walk", "crying a lot", "sleeping a lot"]
```

Out[]:=

A baby.

Translation

```
In[ ]:= LLMSynthesize["Translate this sentece to Hungarian: The  
weather could have been even better though if it was not raining"]
```

Out[]:=

Az időjárás még jobb is lehetett volna, ha nem esett volna az eső.

```
In[ ]:= LLMSynthesize["Translate this sentece to English: Az  
időjárás még jobb is lehetett volna, ha nem esett volna az eső."]
```

Out[]:=

The weather could have been even better if it hadn't rained.

Computer Code

Mathematica

```
In[ ]:= LLMSynthesize[  
"Write a Mathematica code which minimizes the size of a square which contains  
3 circles having radius 2, 3, and 4. The code should compute  
the coordinates of the centers of these cicrles and the side  
length of the square and plot the circles as well as the square "]
```

Out[]:=

Here is a Mathematica code that solves the problem:

```

```mathematica
(* Define the radii of the circles *)
r1 = 2;
r2 = 3;
r3 = 4;

(* Define the variables for the
 center coordinates and the side length of the square *)
{x1, y1} = {x[1], y[1]};
{x2, y2} = {x[2], y[2]};
{x3, y3} = {x[3], y[3]};
s = s;

(* Define the objective function to be minimized *)
objective = s;

(* Define the constraints *)
constraints = {
 (* The circles are contained inside the square *)
 x1 - r1 >= -s/2, x1 + r1 <= s/2, y1 - r1 >= -s/2, y1 + r1 <= s/2,
 x2 - r2 >= -s/2, x2 + r2 <= s/2, y2 - r2 >= -s/2, y2 + r2 <= s/2,
 x3 - r3 >= -s/2, x3 + r3 <= s/2, y3 - r3 >= -s/2, y3 + r3 <= s/2,
 (* The circles do not overlap *)
 (x1 - x2)^2 + (y1 - y2)^2 >= (r1 + r2)^2,
 (x1 - x3)^2 + (y1 - y3)^2 >= (r1 + r3)^2,
 (x2 - x3)^2 + (y2 - y3)^2 >= (r2 + r3)^2
};

(* Solve the optimization problem *)
sol = NMinimize[{objective, constraints}, {x1, x2, x3, y1, y2, y3, s}];

(* Extract the solution *)
{x1sol, x2sol, x3sol, y1sol, y2sol,
 y3sol, ssol} = {x1, x2, x3, y1, y2, y3, s} /. sol[[2]];

(* Print the solution *)
Print["Center of circle 1: (", x1sol, ",", y1sol, ")"];
Print["Center of circle 2: (", x2sol, ",", y2sol, ")"];
Print["Center of circle 3: (", x3sol, ",", y3sol, ")"];
Print["Side length of the square: ", ssol];

(* Plot the circles and the square *)
circle1 = Graphics[{Circle[{x1sol, y1sol}, r1], Red}];
circle2 = Graphics[{Circle[{x2sol, y2sol}, r2], Green}];
circle3 = Graphics[{Circle[{x3sol, y3sol}, r3], Blue}];
square = Graphics[{EdgeForm[Black],
 FaceForm[None], Rectangle[{-ssol/2, -ssol/2}, {ssol/2, ssol/2}]}];
Show[{circle1, circle2, circle3, square}, Axes -> True, PlotRange -> All]
```

```

This code defines the radii of the circles as ``r1``, ``r2``, and ``r3``. It then defines the variables for the center coordinates (``x1``, ``x2``, ``x3``, ``y1``, ``y2``, ``y3``) and the side length of the square (``s``).

The objective function to be minimized is the side length of the square. The constraints ensure that the circles are contained inside the square and do not overlap.

The code uses ``NMinimize`` to solve the optimization problem, and extracts the solution. It then prints the coordinates of the centers of the circles and the side length of the square.

Finally, the code uses ``Graphics`` to plot the circles and the square, with different colors for each circle. The ``Show`` function combines the individual graphics objects and displays the plot with axes and a suitable plot range.

```

In[*]:= (* Define the radii of the circles *)
r1 = 2;
r2 = 3;
r3 = 4;

(* Define the variables for the center
coordinates and the side length of the square *)
{x1, y1} = {x[1], y[1]};
{x2, y2} = {x[2], y[2]};
{x3, y3} = {x[3], y[3]};
s = s;

(* Define the objective function to be minimized *)
objective = s;

(* Define the constraints *)
constraints = {
  (* The circles are contained inside the square *)
  x1 - r1 ≥ -s / 2, x1 + r1 ≤ s / 2, y1 - r1 ≥ -s / 2, y1 + r1 ≤ s / 2,
  x2 - r2 ≥ -s / 2, x2 + r2 ≤ s / 2, y2 - r2 ≥ -s / 2, y2 + r2 ≤ s / 2,
  x3 - r3 ≥ -s / 2, x3 + r3 ≤ s / 2, y3 - r3 ≥ -s / 2, y3 + r3 ≤ s / 2,
  (* The circles do not overlap *)
  (x1 - x2)^2 + (y1 - y2)^2 ≥ (r1 + r2)^2,
  (x1 - x3)^2 + (y1 - y3)^2 ≥ (r1 + r3)^2,
  (x2 - x3)^2 + (y2 - y3)^2 ≥ (r2 + r3)^2
};

(* Solve the optimization problem *)
sol = NMinimize[{objective, constraints}, {x1, x2, x3, y1, y2, y3, s}];

(* Extract the solution *)
{x1sol, x2sol, x3sol, y1sol, y2sol, y3sol, ssol} =
  {x1, x2, x3, y1, y2, y3, s} /. sol[[2]];

(* Print the solution *)
Print["Center of circle 1: (", x1sol, ",", y1sol, ")"];
Print["Center of circle 2: (", x2sol, ",", y2sol, ")"];
Print["Center of circle 3: (", x3sol, ",", y3sol, ")"];
Print["Side length of the square: ", ssol];

(* Plot the circles and the square *)
circle1 = Graphics[{Circle[{x1sol, y1sol}, r1], Red}];
circle2 = Graphics[{Circle[{x2sol, y2sol}, r2], Green}];
circle3 = Graphics[{Circle[{x3sol, y3sol}, r3], Blue}];
square = Graphics[{EdgeForm[Black], FaceForm[None],
  Rectangle[{-ssol / 2, -ssol / 2}, {ssol / 2, ssol / 2}]}];
Show[{circle1, circle2, circle3, square}, Axes → True, PlotRange → All]

```

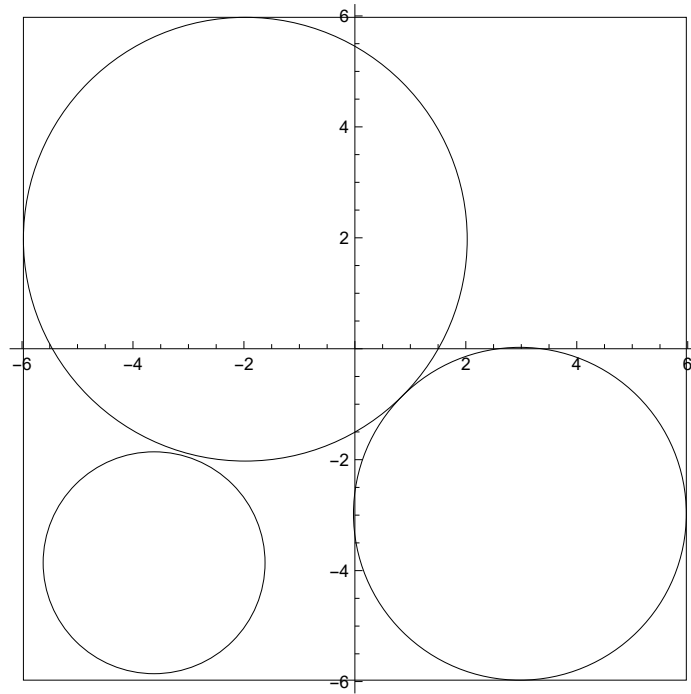

Center of circle 1: $(-3.61931, -3.85845)$

Center of circle 2: $(2.97487, -2.97487)$

Center of circle 3: $(-1.97487, 1.97487)$

Side length of the square: 11.9497

Out[]=



Python

In[*]:= **LLMSynthesize**[

"Generate a neural network in Python, which has input set X=[1,2,3,4,5,6,7,8] and the target set y=[1,4,9,16,25,36,49,64]. Plot the result. Employ sklearn."]

Out[*]=

To generate a neural network using scikit-learn in Python and plot the result, we can use the MLPRegressor class from the neural_network module. Here's the code:

```
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor

Input set X and target set y
X = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)
y = np.array([1, 4, 9, 16, 25, 36, 49, 64])

Create a MLPRegressor with one hidden layer of 100 neurons
model = MLPRegressor(hidden_layer_sizes=(100,), random_state=1, max_iter=2000)

Fit the model to the data
model.fit(X, y)

Predict using the trained model
y_pred = model.predict(X)

Plotting the result
plt.scatter(X, y, color='b', label='Actual')
plt.plot(X, y_pred, color='r', label='Predicted')
plt.legend()
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```
```

This code will generate a neural network model with one hidden layer of 100 neurons.


It will fit the model to the input set X and target set y. Finally, it will predict the target values using the trained model and plot the actual and predicted values.

Make sure to have scikit-learn and matplotlib

installed in your Python environment before running this code.

In[*]:= **session = StartExternalSession**["Python"]

Out[*]=

ExternalSessionObject [ System: Python Version: 3.7.4
UUID: e4633b62-5bae-4f0e-b514-68e243e311be]

In[*]:=

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.neural_network import MLPRegressor

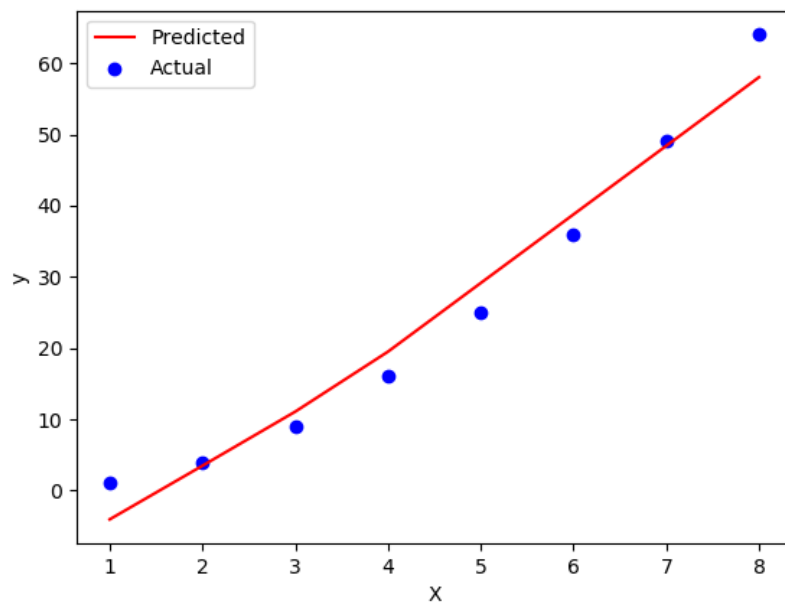
# Input set X and target set y
X = np.array([1, 2, 3, 4, 5, 6, 7, 8]).reshape(-1, 1)
y = np.array([1, 4, 9, 16, 25, 36, 49, 64])

# Create a MLPRegressor with one hidden layer of 100 neurons
model = MLPRegressor(hidden_layer_sizes=(100,), random_state=1, max_iter=2000)

# Fit the model to the data
model.fit(X, y)

# Predict using the trained model
y_pred = model.predict(X)

# Plotting the result
plt.scatter(X, y, color='b', label='Actual')
plt.plot(X, y_pred, color='r', label='Predicted')
plt.legend()
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```



Vision

```
In[*]:= ImageSynthesize["Show a green mouse wich jumping with a parachute " ]  
Out[*]=
```



```
In[*]:= ImageSynthesize["A painting of a cat in the style of Dali"]  
Out[*]=
```



```
ImageSynthesize["Brain sitting on a rocket flying to the moon"]
```

```
Out[ ]:=
```



```
In[ ]:= LLMSynthesize["A haiku about the swallow"]
```

```
Out[ ]:=
```

```
Swift in the blue sky,  
Swallow dances with grace,  
Whispers of spring's embrace.
```

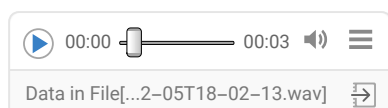
```
In[ ]:= SpeechSynthesize[  
    "Swift in the blue sky, Swallow dances with grace, Whispers of spring's embrace."]
```

```
Out[ ]:=
```



```
In[ ]:= SpeechSynthesize["5+12/4+Sin[1]", "Zira"]
```

```
Out[ ]:=
```



Outlook and Final remarks

A generative model is a type of machine learning model that aims to learn the underlying *patterns or distributions of data in order to generate new, similar data* . In essence, it's like teaching a computer to dream up its own data based on what it has seen before . The significance of this model lies in its ability to create, which has vast implications in various fields, from art to science .

Out of LLM (*Large Language Model*) as GPT there are different models for the Generative AI, too.

Variational Autoencoders (VAEs). VAEs are a type of autoencoder that produces a compressed representation of input data, then decodes it to generate new data.

They're often used in tasks like image denoising or generating new images that share characteristics with the input data.

Generative Adversarial Networks (GANs). GANs consist of two neural networks, the generator and the discriminator, that are trained together. The generator tries to produce data, while the discriminator attempts to distinguish between real and generated data. Over time, the generator becomes so good that the discriminator can't tell the difference. GANs are popular in image generation tasks, such as creating realistic human faces or artworks.

MIT Introduction to Deep Learning 6.S191: Lecture 4

Deep Generative Modeling

Lecturer: Ava Amini

```
In[*]:= SystemOpen["https://www.youtube.com/watch?v=3G5hWM6jqPk"]
```

- Classical computers get overwhelmed by exponential calculations when it comes to these enormous amounts of data...plus AI and machine learning algorithms need parallel computations therefore *quantum processing* are the perfect candidates

- AI is the nuclear power of our age – *Henry Kissinger*

```
In[*]:= SystemOpen["https://www.youtube.com/watch?v=182PxsKHxYc"]
```

It is amazing and amusing but a bit frightening too!