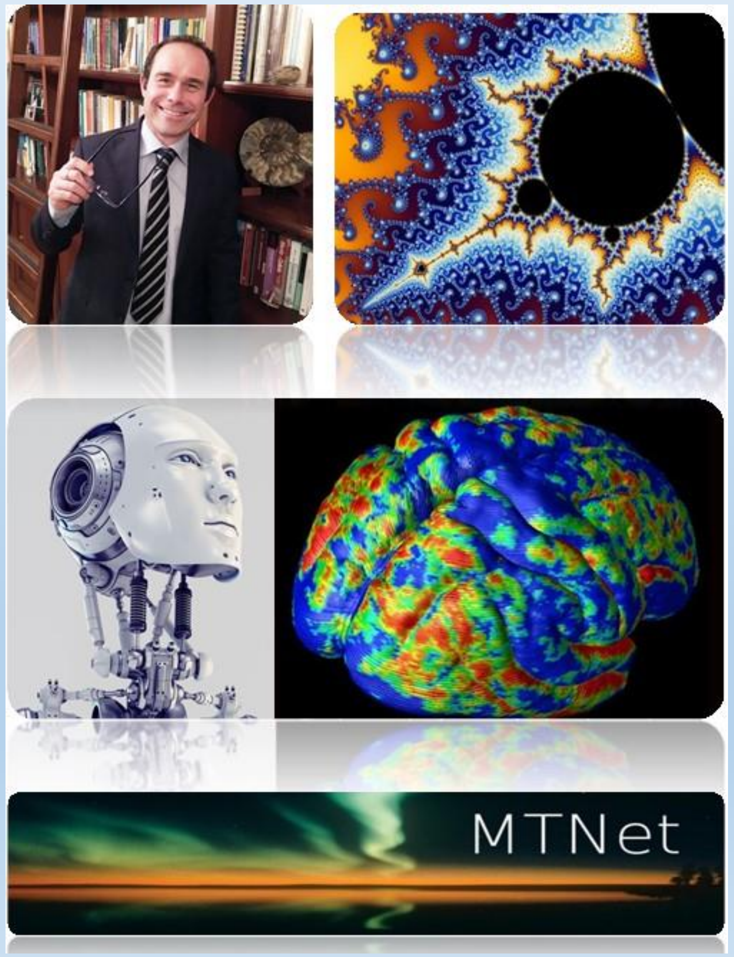
Testing GPT-3 for the Geosciences

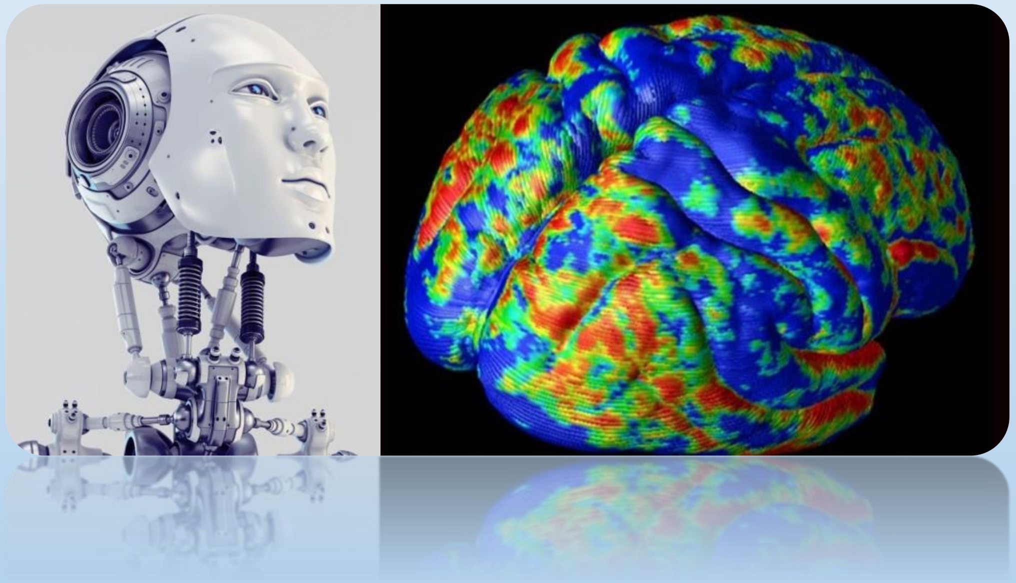
Benefits and limitations

Paolo Dell'Aversana



Reference paper

GPT-3: a new cooperation scenario between humans and machines. Benefits and limitations of GPT-3 as a coding virtual assistant



Paolo Dell'Aversana

OUTLINE

OUTLINE

- **Introduction**
- Examples, examples, examples …
- Applications in Geosciences
- A critical discussion: benefits and limitations
- Creative use of GPT-3
- Conclusions
- Questions and discussion

Introduction

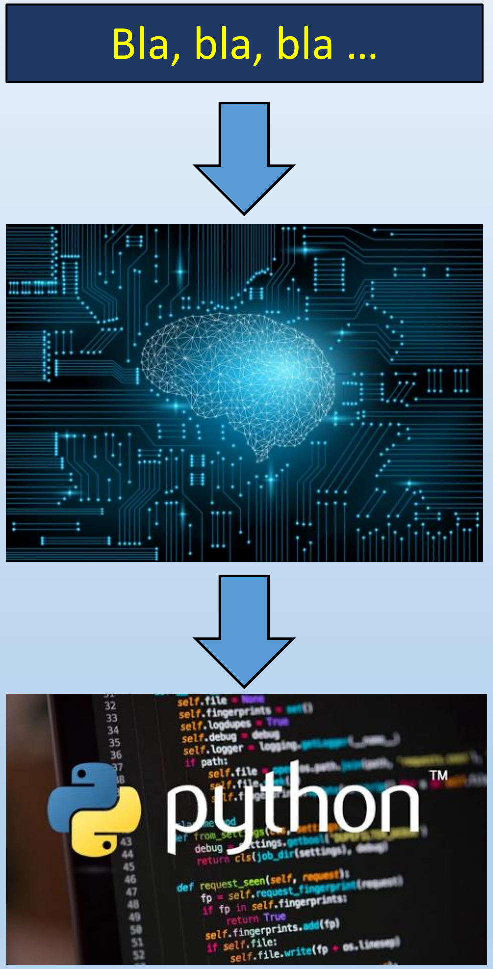
- GPT-3 (Generative Pre-trained Transformer) is a language generation model.
- It uses a ‘Transformer architecture’, which is a type of neural network designed for natural language processing tasks (Vaswani et al., 2017).
- GPT-3 uses ‘self-attention’.
- This allows the model to analyse various parts of the input and determine dependencies between them.

Self attention

- Self-attention allows calculating the representation of a **sequence of input vectors** (e.g. words in a sentence).
- It works by computing "**attention weights**" between every pair of elements in the sequence, indicating the importance of each element with respect to the others.
- These attention weights are then used to compute a weighted sum of the elements, representing the **sequence as a single vector**.
- This allows the Transformer to capture **long-range dependencies** in the input sequence, since it considers all elements jointly rather than one at a time.

GPT-3 can do …

- Text generation
- Language translation
- Text summarization
- Question answering
- Text classification
- Sentiment analysis
- …
- Text-to-code generation

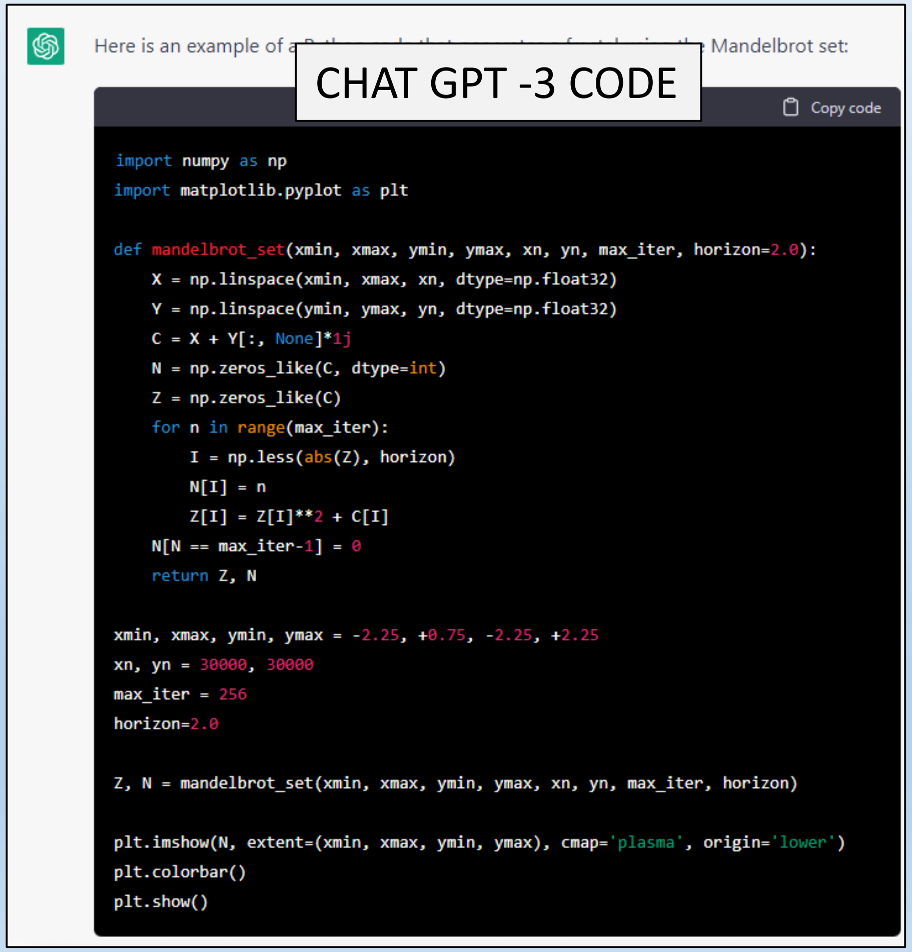


OUTLINE

- Introduction
- **Examples, examples, examples …**
- Applications in Geosciences
- A critical discussion: benefits and limitations
- Creative use of GPT-3
- Conclusions
- Questions and discussion

Example

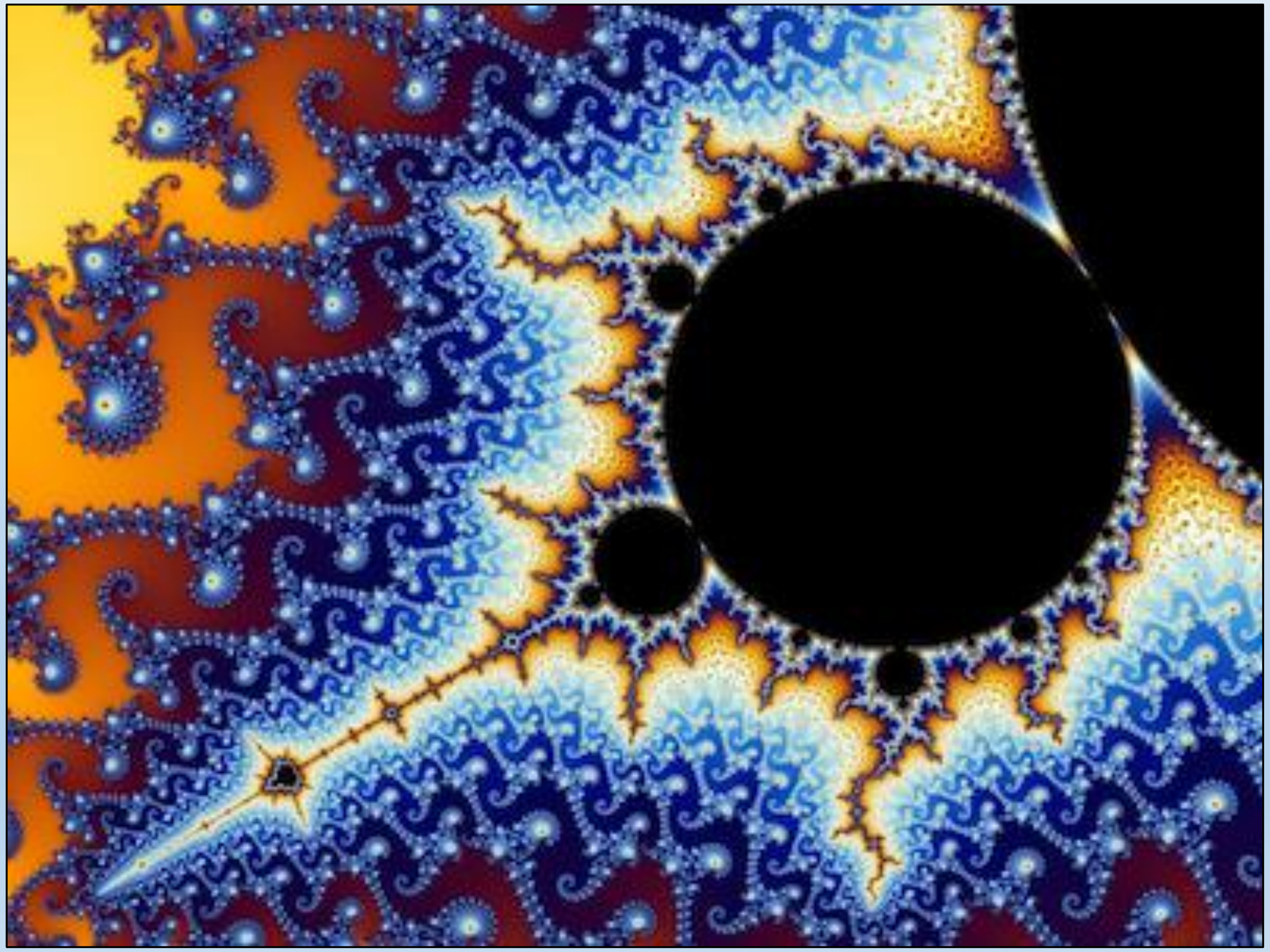
Python code for creating fractals

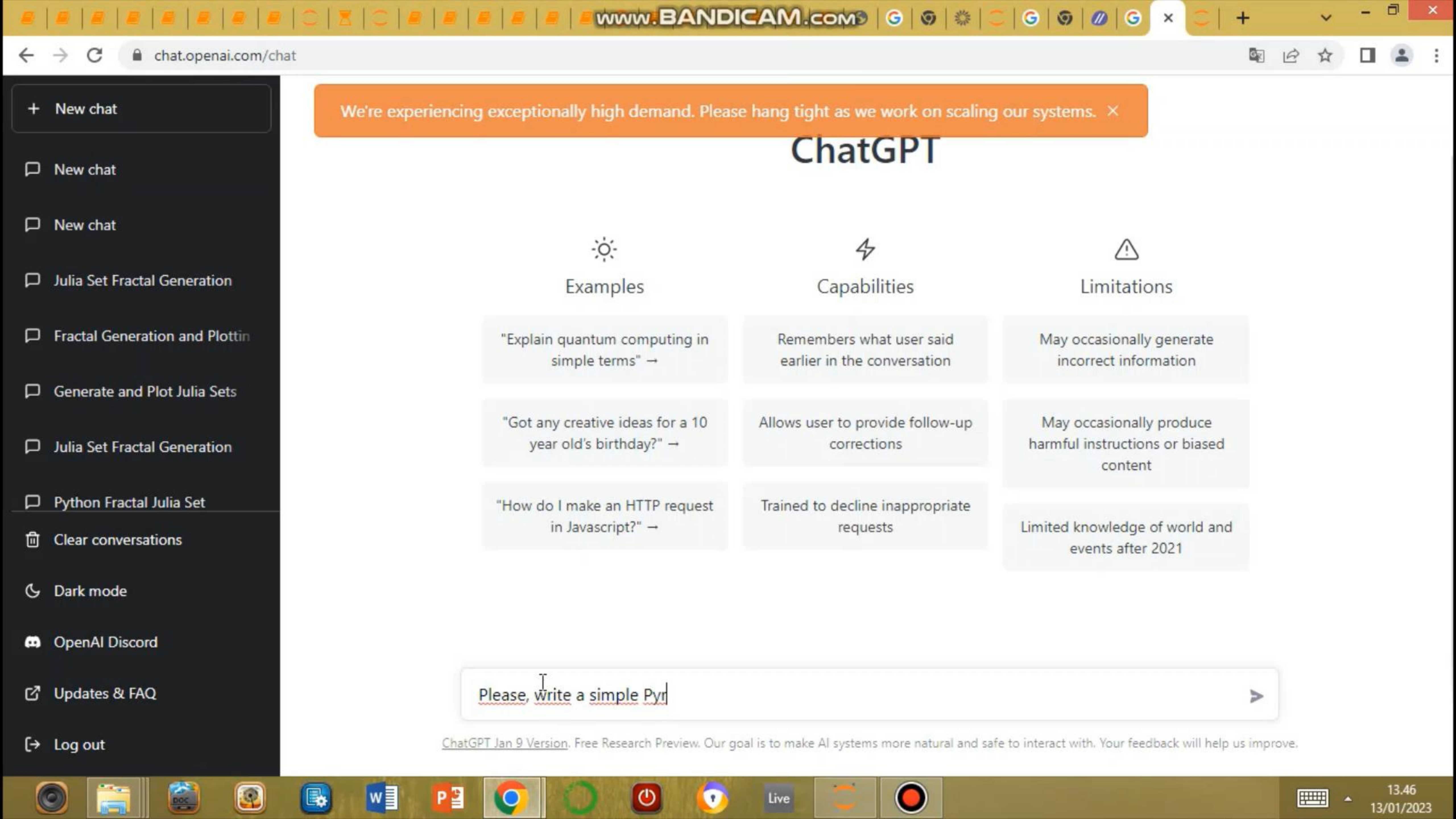


Here is an example of a Python code that generates a fractal using the Mandelbrot set:

CHAT GPT -3 CODE

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

def mandelbrot_set(xmin, xmax, ymin, ymax, xn, yn, max_iter, horizon=2.0):
    X = np.linspace(xmin, xmax, xn, dtype=np.float32)
    Y = np.linspace(ymin, ymax, yn, dtype=np.float32)
    C = X + Y[:, None]*1j
    N = np.zeros_like(C, dtype=int)
    Z = np.zeros_like(C)
    for n in range(max_iter):
        I = np.less(abs(Z), horizon)
        N[I] = n
        Z[I] = Z[I]**2 + C[I]
    N[N == max_iter-1] = 0
    return Z, N

xmin, xmax, ymin, ymax = -2.25, +0.75, -2.25, +2.25
xn, yn = 30000, 30000
max_iter = 256
horizon=2.0

Z, N = mandelbrot_set(xmin, xmax, ymin, ymax, xn, yn, max_iter, horizon)

plt.imshow(N, extent=(xmin, xmax, ymin, ymax), cmap='plasma', origin='lower')
plt.colorbar()
plt.show()
```





www.BANDICAM.com

chat.openai.com/chat

+ New chat
New chat
New chat
Julia Set Fractal Generation
Fractal Generation and Plottin
Generate and Plot Julia Sets
Julia Set Fractal Generation
Python Fractal Julia Set
Clear conversations
Dark mode
OpenAI Discord
Updates & FAQ
Log out

We're experiencing exceptionally high demand. Please hang tight as we work on scaling our systems.

ChatGPT

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

Please, write a simple Pyr

ChatGPT Jan 9 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

This opens new doors for creativity

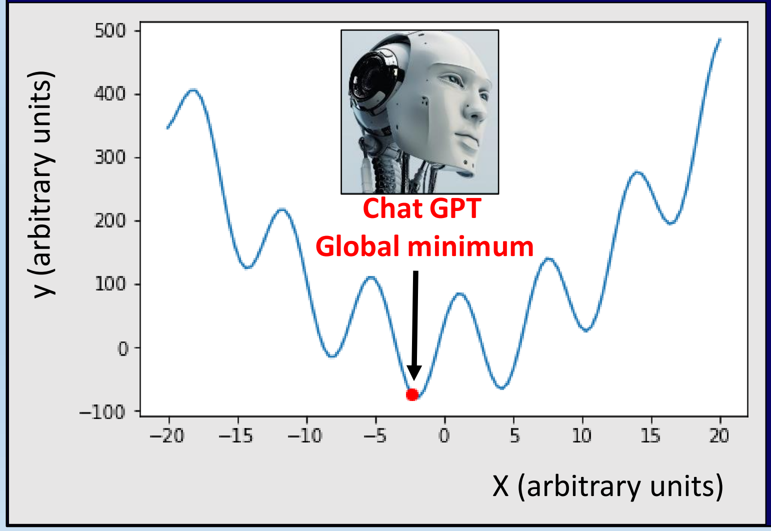
- GPT-3 is a virtual assistant for transforming raw ideas into prototype codes and for testing them.

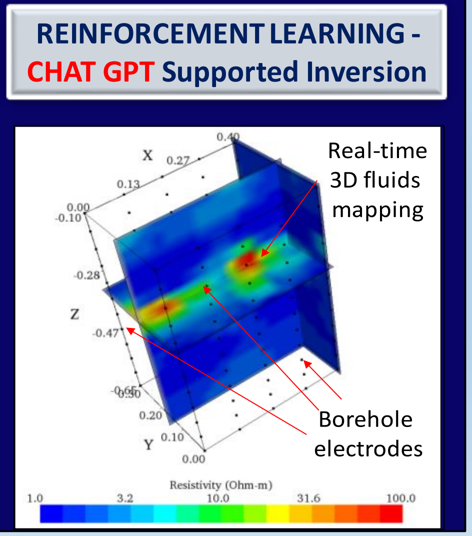
- Examples:
  - Processing algorithms
  - Inversion
  - Optimization
  - Deep Neural Networks
  - Image recognition
  - Sound analysis
  - Advanced statistics
  - …

GPT-3 code examples

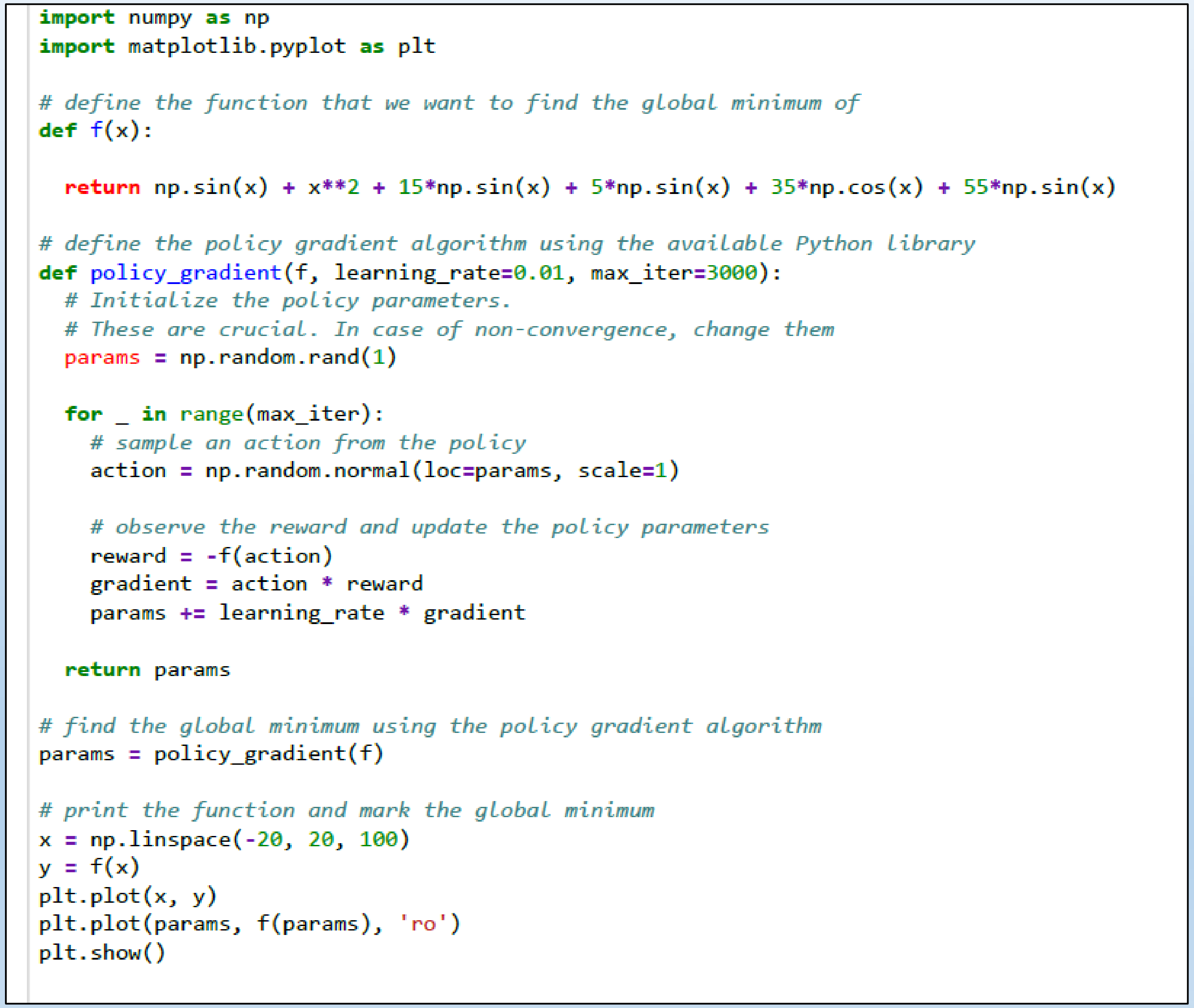
Some preliminary applications

- New global optimization approaches
- Novel algorithms for joint inversion of seismic and electromagnetic data
- New methods for predictive/prescriptive reservoir management
- New approaches for fluid displacement prediction (Oil, gas, CO2 …)
- Time-lapse resistivity inversion for aquifer monitoring
- …

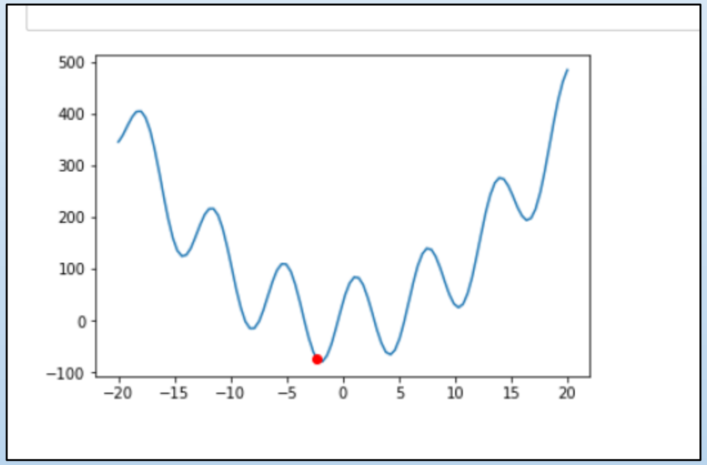




Finding the global minimum of a complicate function



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

# define the function that we want to find the global minimum of
def f(x):

  return np.sin(x) + x**2 + 15*np.sin(x) + 5*np.sin(x) + 35*np.cos(x) + 55*np.sin(x)

# define the policy gradient algorithm using the available Python library
def policy_gradient(f, learning_rate=0.01, max_iter=3000):
  # Initialize the policy parameters.
  # These are crucial. In case of non-convergence, change them
  params = np.random.rand(1)

  for _ in range(max_iter):
    # sample an action from the policy
    action = np.random.normal(loc=params, scale=1)

    # observe the reward and update the policy parameters
    reward = -f(action)
    gradient = action * reward
    params += learning_rate * gradient

  return params

# find the global minimum using the policy gradient algorithm
params = policy_gradient(f)

# print the function and mark the global minimum
x = np.linspace(-20, 20, 100)
y = f(x)
plt.plot(x, y)
plt.plot(params, f(params), 'ro')
plt.show()
```

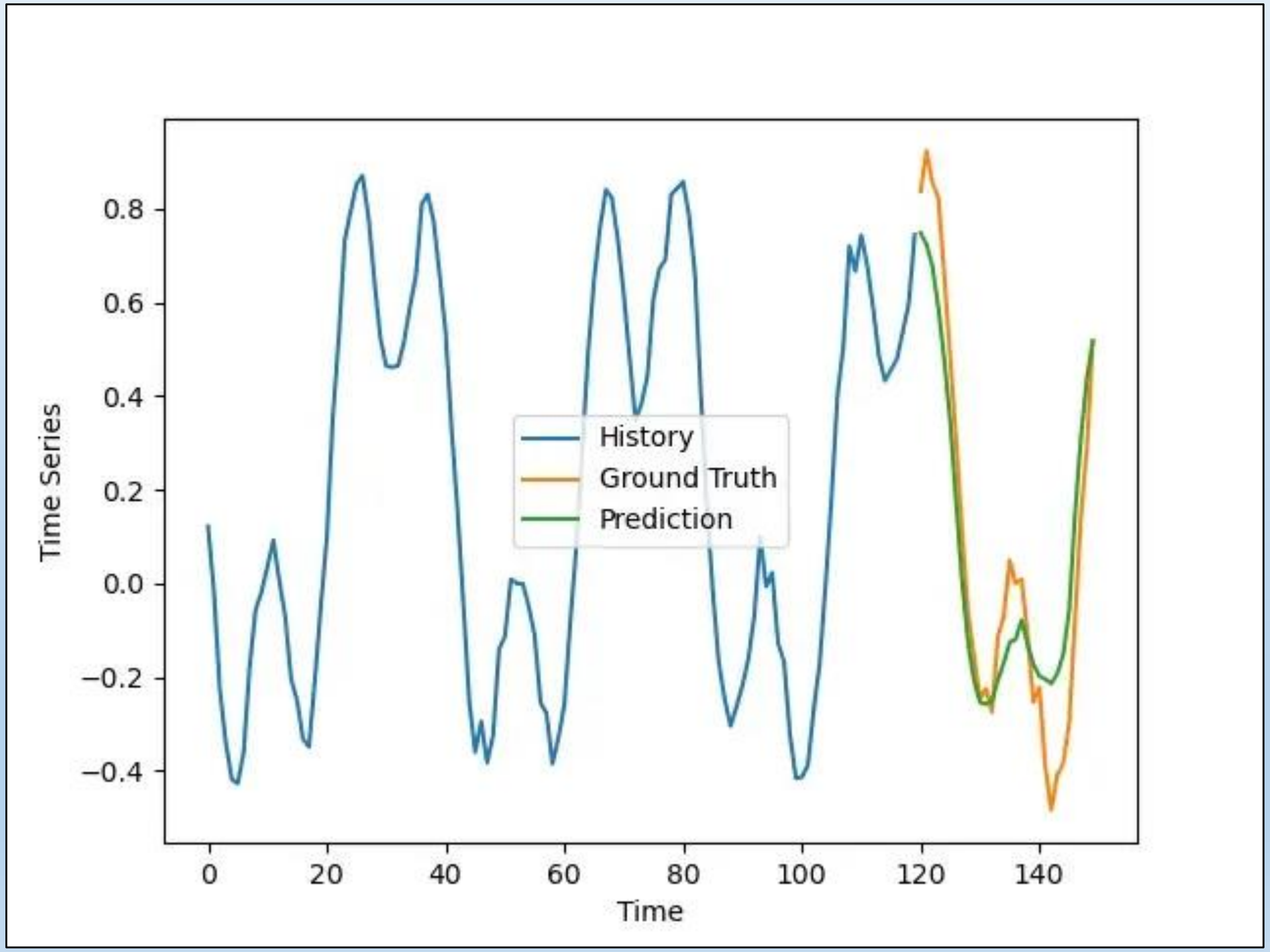


OUTLINE

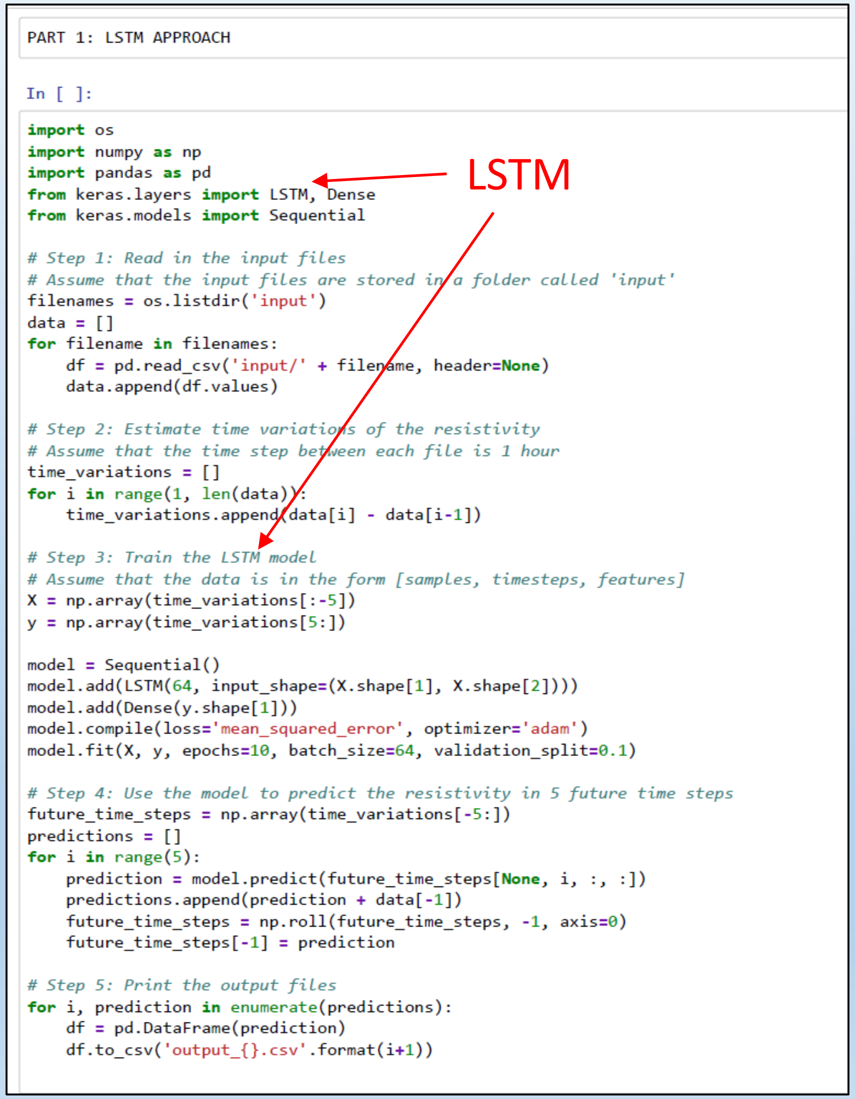
Transformers and forecasting models

- Transformers are useful for predicting future data in a sequence because they are specifically designed to process sequential data.

- Their architecture allows the model to take into account the entire context of the sequence, rather than just the previous few elements.



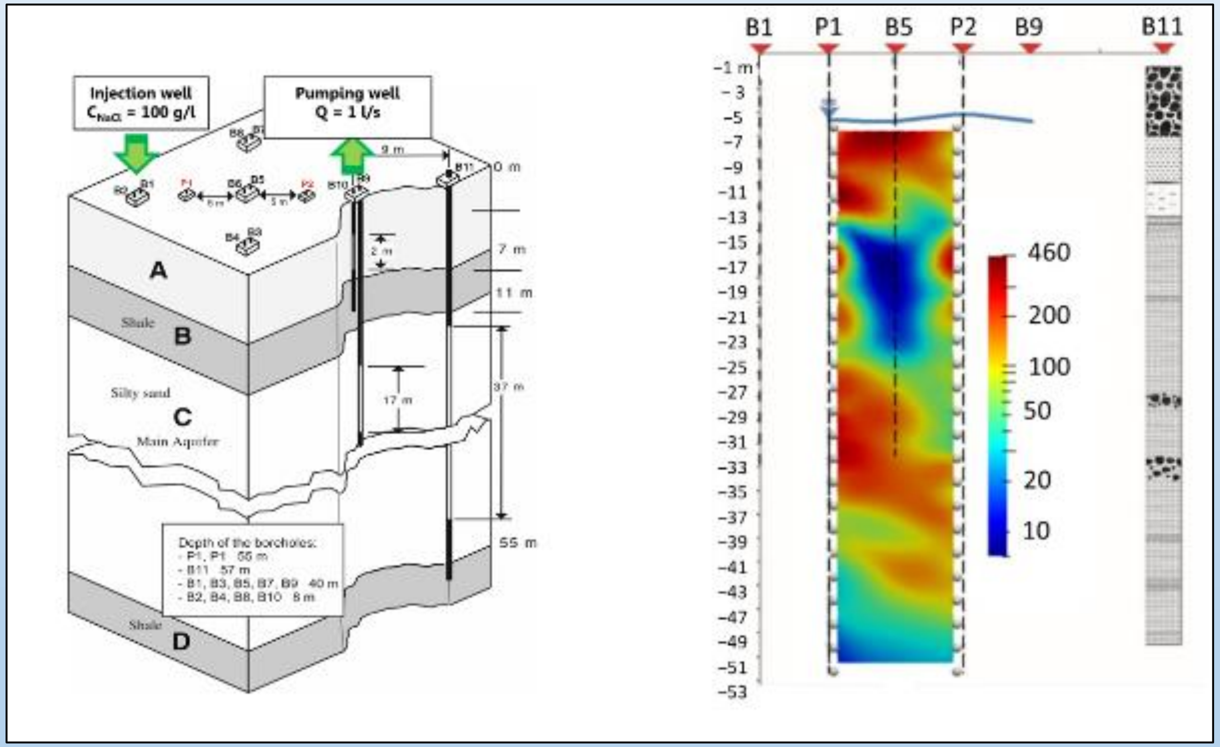
RECURRENT NEURAL NETWORKS AND TRANSFORMERS FOR PREDICTING RESISTIVITY VARIATIONS OVER TIME



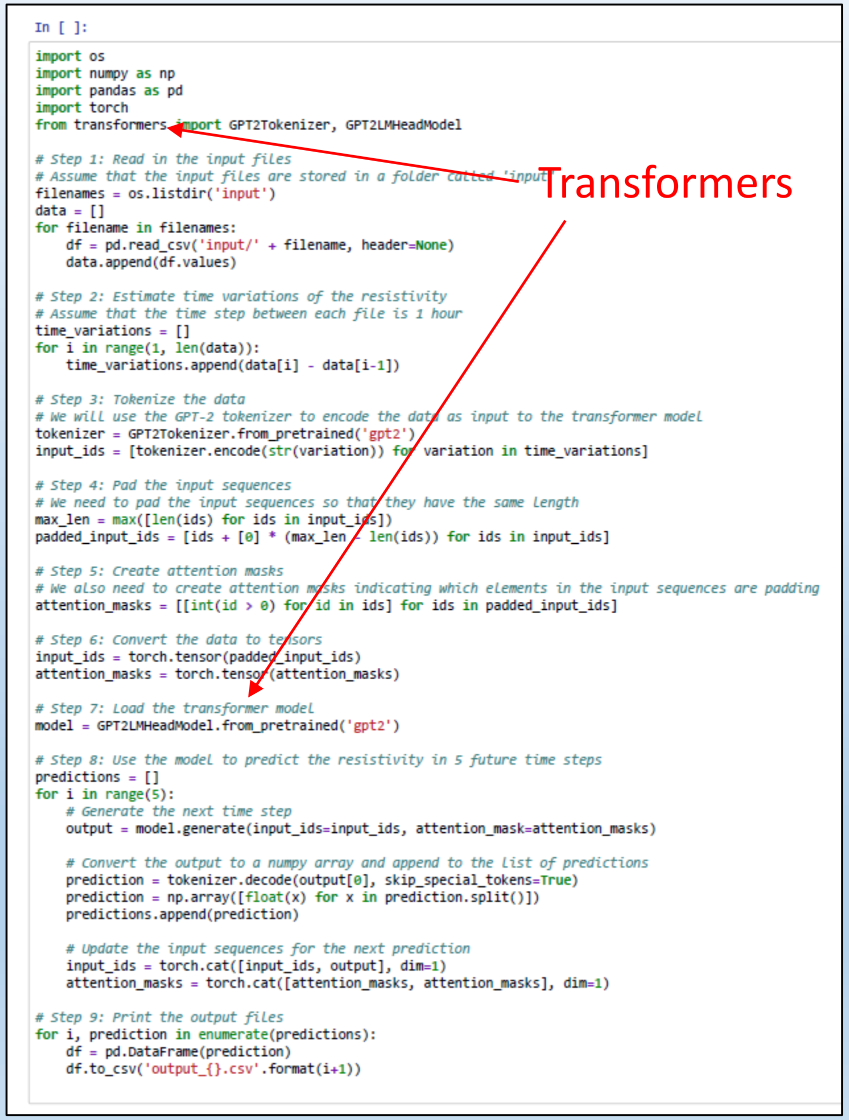
PART 1: LSTM APPROACH

In [ ]:

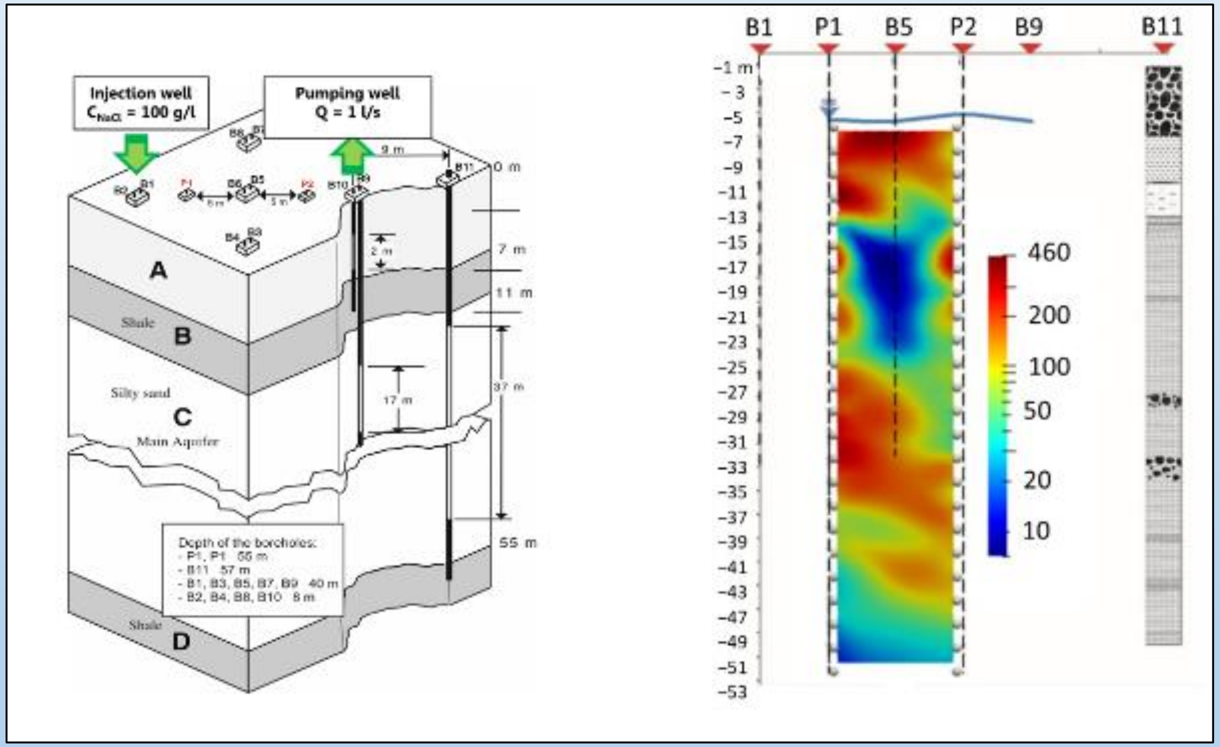
```python
import os
import numpy as np
import pandas as pd
from keras.layers import LSTM, Dense
from keras.models import Sequential

# Step 1: Read in the input files
# Assume that the input files are stored in a folder called 'input'
filenames = os.listdir('input')
data = []
for filename in filenames:
    df = pd.read_csv('input/' + filename, header=None)
    data.append(df.values)

# Step 2: Estimate time variations of the resistivity
# Assume that the time step between each file is 1 hour
time_variations = []
for i in range(1, len(data)):
    time_variations.append(data[i] - data[i-1])

# Step 3: Train the LSTM model
# Assume that the data is in the form [samples, timesteps, features]
X = np.array(time_variations[:-5])
y = np.array(time_variations[5:])

model = Sequential()
model.add(LSTM(64, input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(y.shape[1]))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X, y, epochs=10, batch_size=64, validation_split=0.1)

# Step 4: Use the model to predict the resistivity in 5 future time steps
future_time_steps = np.array(time_variations[-5:])
predictions = []
for i in range(5):
    prediction = model.predict(future_time_steps[None, i, :, :])
    predictions.append(prediction + data[-1])
    future_time_steps = np.roll(future_time_steps, -1, axis=0)
    future_time_steps[-1] = prediction

# Step 5: Print the output files
for i, prediction in enumerate(predictions):
    df = pd.DataFrame(prediction)
    df.to_csv('output_{}.csv'.format(i+1))
```



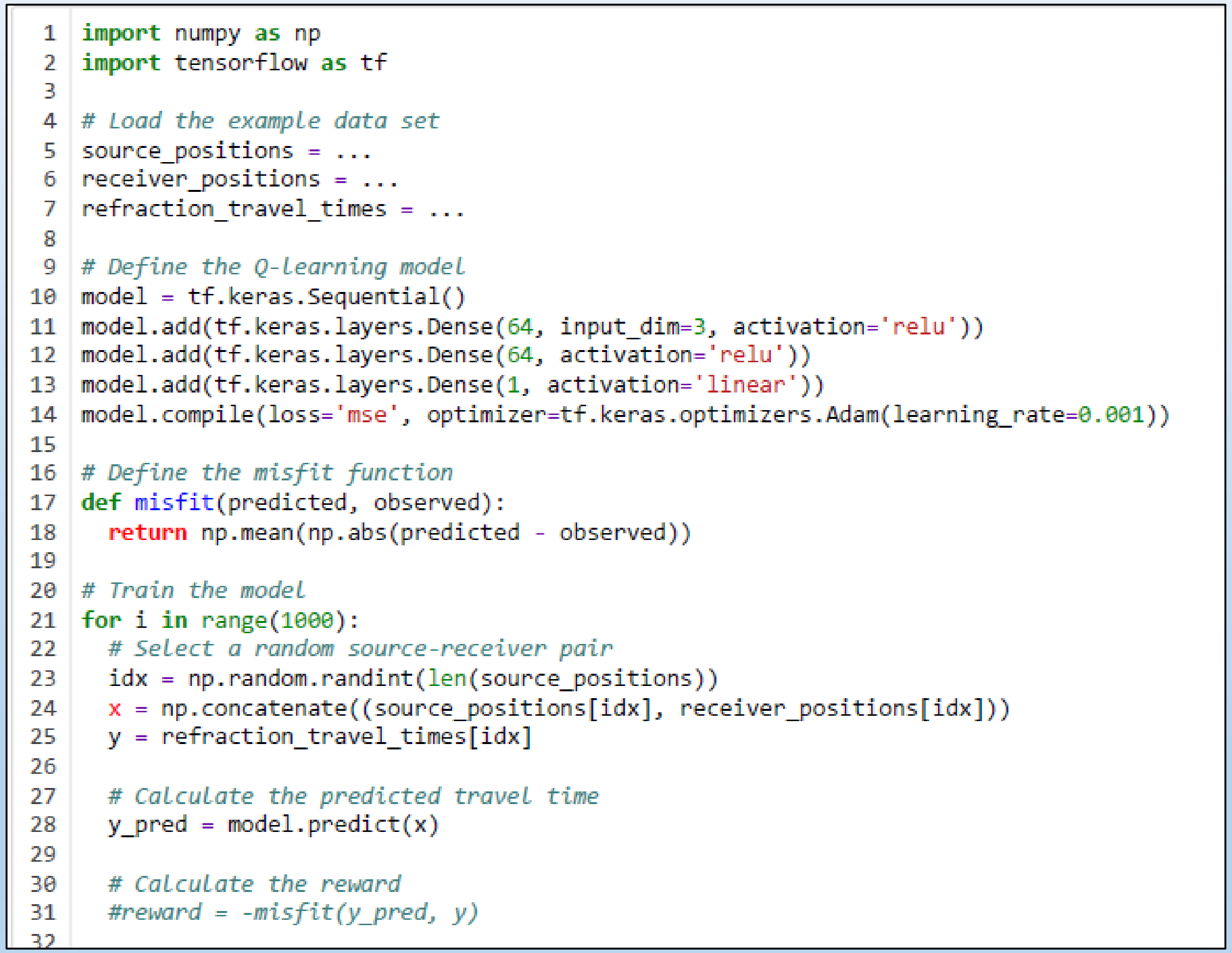
RECURRENT NEURAL NETWORKS AND TRANSFORMERS FOR PREDICTING RESISTIVITY VARIATIONS OVER TIME



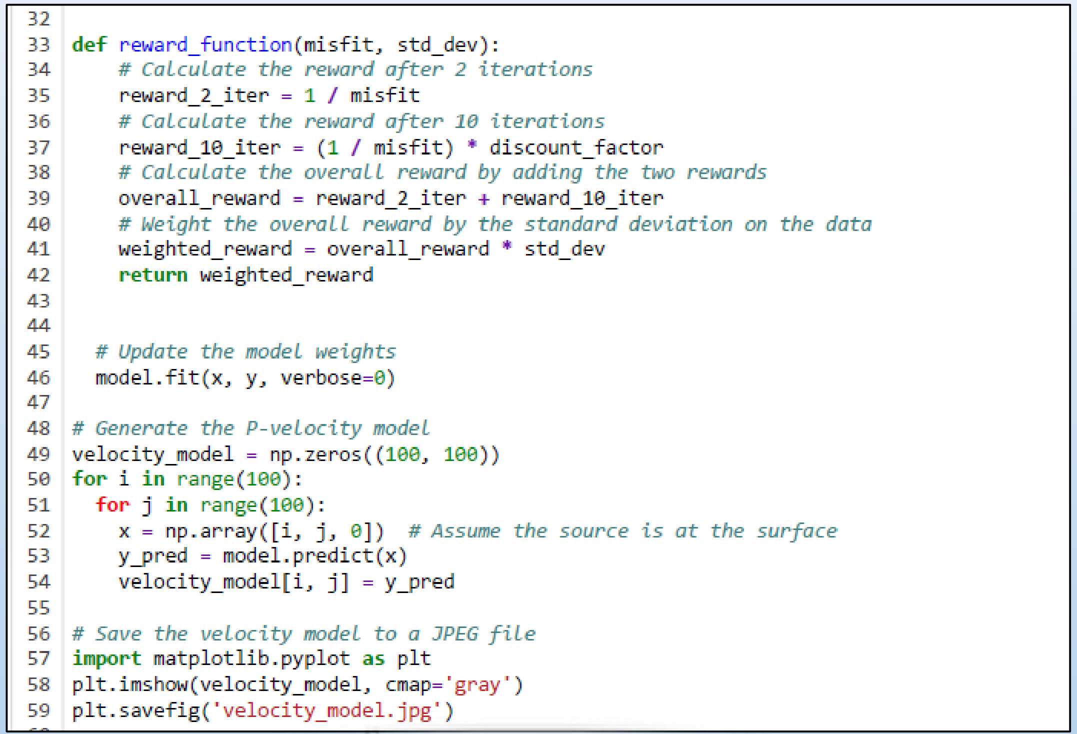
```
In [ ]:
import os
import numpy as np
import pandas as pd
import torch
from transformers import GPT2Tokenizer, GPT2LMHeadModel

# Step 1: Read in the input files
# Assume that the input files are stored in a folder called 'input'
filenames = os.listdir('input')
data = []
for filename in filenames:
    df = pd.read_csv('input/' + filename, header=None)
    data.append(df.values)

# Step 2: Estimate time variations of the resistivity
# Assume that the time step between each file is 1 hour
time_variations = []
for i in range(1, len(data)):
    time_variations.append(data[i] - data[i-1])

# Step 3: Tokenize the data
# We will use the GPT-2 tokenizer to encode the data as input to the transformer model
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
input_ids = [tokenizer.encode(str(variation)) for variation in time_variations]

# Step 4: Pad the input sequences
# We need to pad the input sequences so that they have the same length
max_len = max([len(ids) for ids in input_ids])
padded_input_ids = [ids + [0] * (max_len - len(ids)) for ids in input_ids]

# Step 5: Create attention masks
# We also need to create attention masks indicating which elements in the input sequences are padding
attention_masks = [[int(id > 0) for id in ids] for ids in padded_input_ids]

# Step 6: Convert the data to tensors
input_ids = torch.tensor(padded_input_ids)
attention_masks = torch.tensor(attention_masks)

# Step 7: Load the transformer model
model = GPT2LMHeadModel.from_pretrained('gpt2')

# Step 8: Use the model to predict the resistivity in 5 future time steps
predictions = []
for i in range(5):
    # Generate the next time step
    output = model.generate(input_ids=input_ids, attention_mask=attention_masks)

    # Convert the output to a numpy array and append to the list of predictions
    prediction = tokenizer.decode(output[0], skip_special_tokens=True)
    prediction = np.array([float(x) for x in prediction.split()])
    predictions.append(prediction)

    # Update the input sequences for the next prediction
    input_ids = torch.cat([input_ids, output], dim=1)
    attention_masks = torch.cat([attention_masks, attention_masks], dim=1)

# Step 9: Print the output files
for i, prediction in enumerate(predictions):
    df = pd.DataFrame(prediction)
    df.to_csv('output_{}.csv'.format(i+1))
```



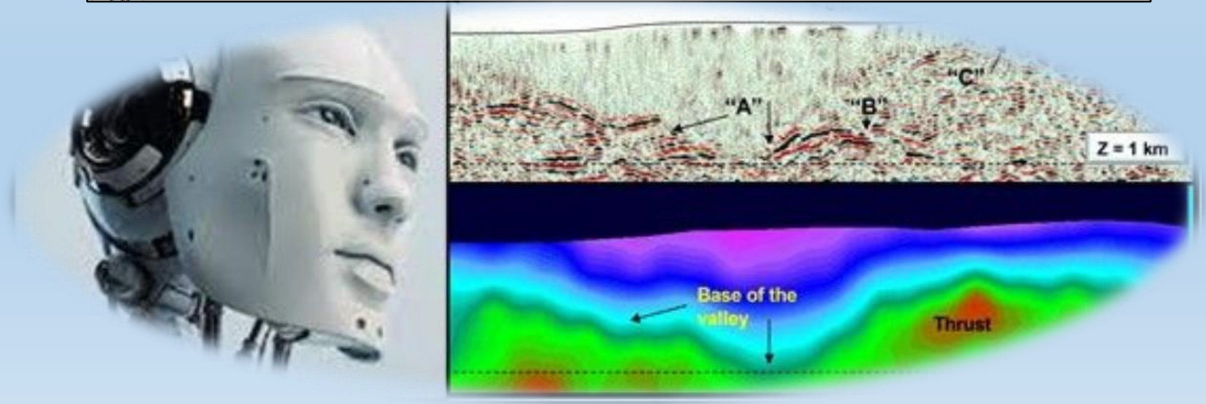
REFRACTION TOMOGRAPHY SUPPORTED BY Q-LEARNING



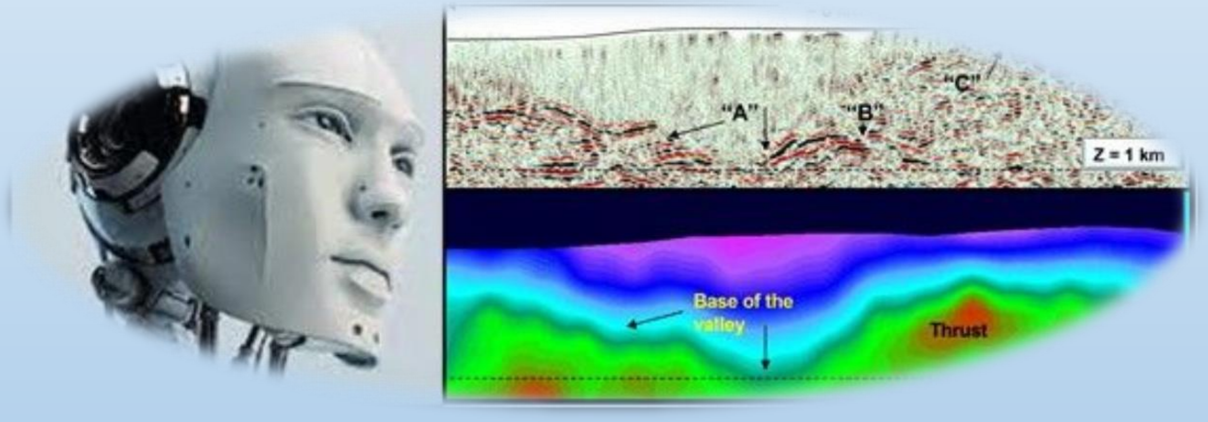
```python
import numpy as np
import tensorflow as tf

# Load the example data set
source_positions = ...
receiver_positions = ...
refraction_travel_times = ...

# Define the Q-learning model
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(64, input_dim=3, activation='relu'))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='linear'))
model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(learning_rate=0.001))

# Define the misfit function
def misfit(predicted, observed):
  return np.mean(np.abs(predicted - observed))

# Train the model
for i in range(1000):
  # Select a random source-receiver pair
  idx = np.random.randint(len(source_positions))
  x = np.concatenate((source_positions[idx], receiver_positions[idx]))
  y = refraction_travel_times[idx]

  # Calculate the predicted travel time
  y_pred = model.predict(x)

  # Calculate the reward
  #reward = -misfit(y_pred, y)

```



```python

def reward_function(misfit, std_dev):
    # Calculate the reward after 2 iterations
    reward_2_iter = 1 / misfit
    # Calculate the reward after 10 iterations
    reward_10_iter = (1 / misfit) * discount_factor
    # Calculate the overall reward by adding the two rewards
    overall_reward = reward_2_iter + reward_10_iter
    # Weight the overall reward by the standard deviation on the data
    weighted_reward = overall_reward * std_dev
    return weighted_reward


  # Update the model weights
  model.fit(x, y, verbose=0)

# Generate the P-velocity model
velocity_model = np.zeros((100, 100))
for i in range(100):
  for j in range(100):
    x = np.array([i, j, 0])  # Assume the source is at the surface
    y_pred = model.predict(x)
    velocity_model[i, j] = y_pred

# Save the velocity model to a JPEG file
import matplotlib.pyplot as plt
plt.imshow(velocity_model, cmap='gray')
plt.savefig('velocity_model.jpg')
```



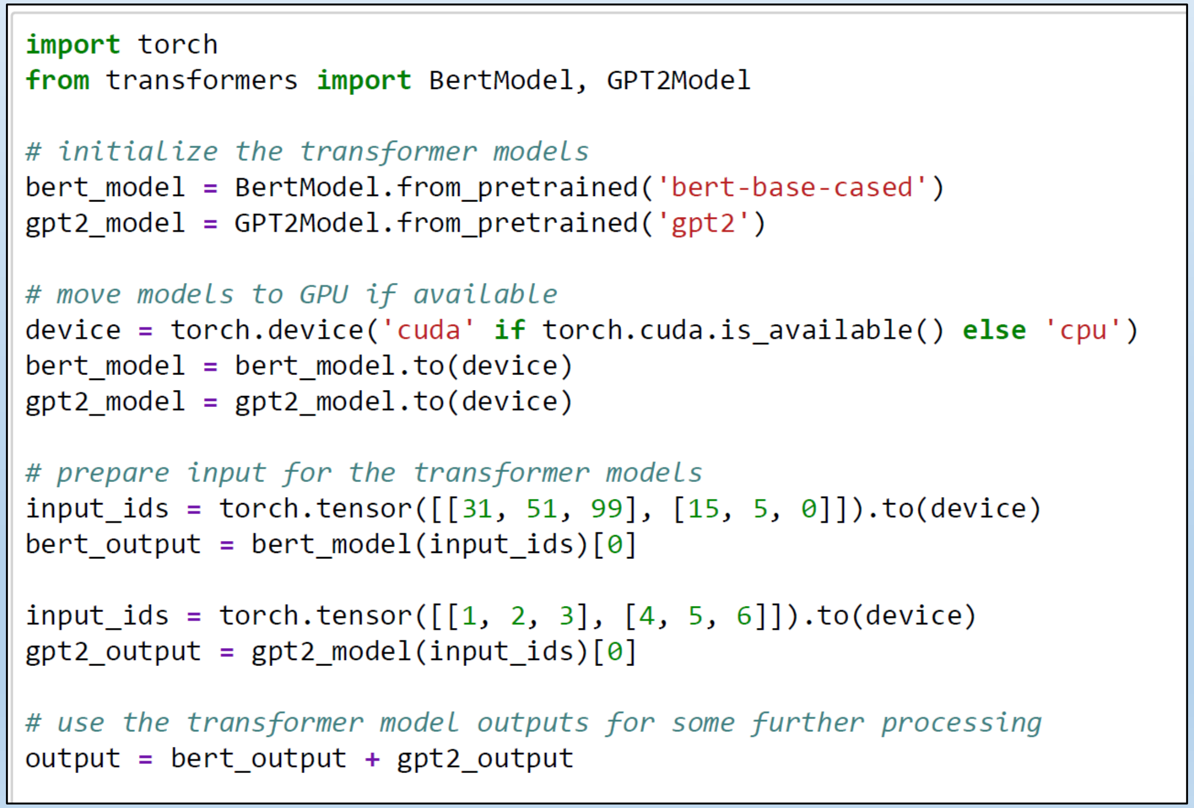
MANY IS BETTER THAN ONE



Combining multiple Transformers

MANY IS BETTER THAN ONE

Using the ‘BERT’ and ‘GPT’ transformer models in parallel

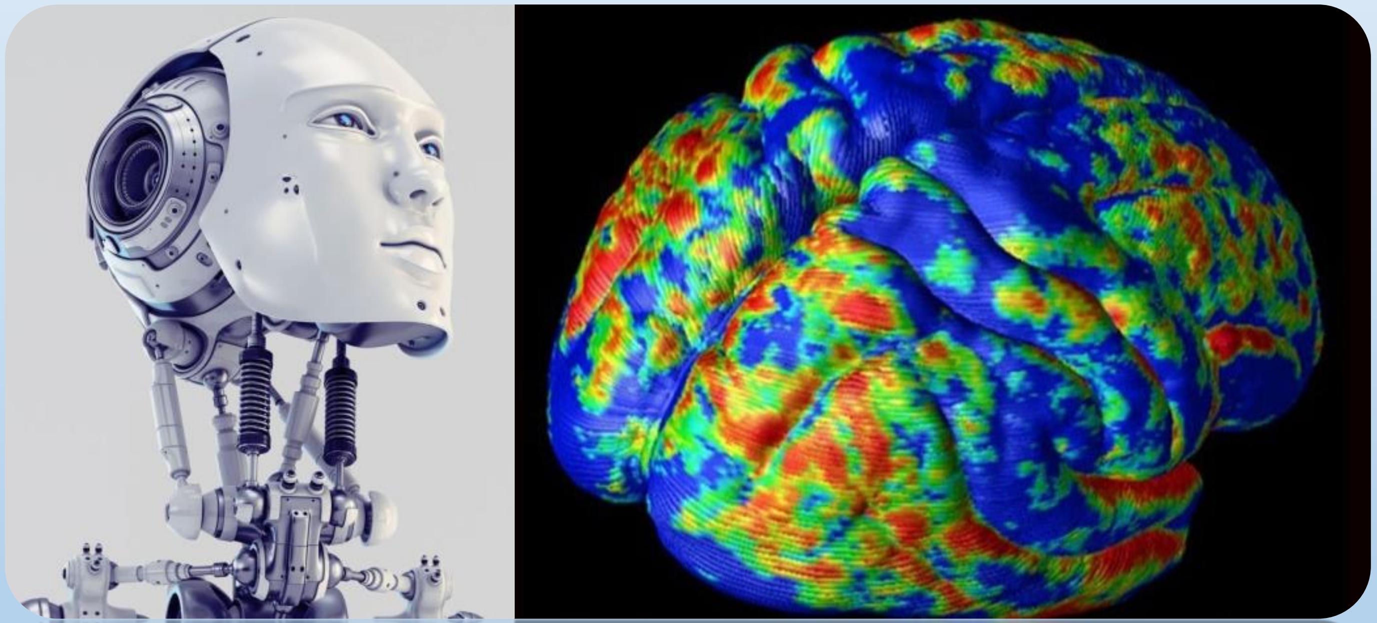


```python
import torch
from transformers import BertModel, GPT2Model

# initialize the transformer models
bert_model = BertModel.from_pretrained('bert-base-cased')
gpt2_model = GPT2Model.from_pretrained('gpt2')

# move models to GPU if available
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
bert_model = bert_model.to(device)
gpt2_model = gpt2_model.to(device)

# prepare input for the transformer models
input_ids = torch.tensor([[31, 51, 99], [15, 5, 0]]).to(device)
bert_output = bert_model(input_ids)[0]

input_ids = torch.tensor([[1, 2, 3], [4, 5, 6]]).to(device)
gpt2_output = gpt2_model(input_ids)[0]

# use the transformer model outputs for some further processing
output = bert_output + gpt2_output
```

OUTLINE

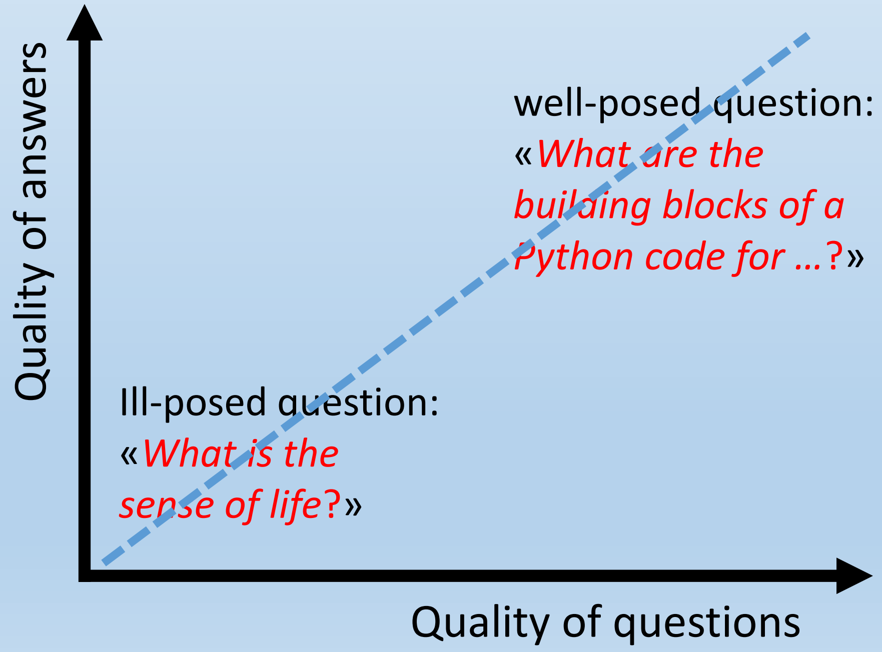
- Introduction
- Examples, examples, examples …
- Applications in Geosciences
- **A critical discussion: benefits and limitations**
- Creative use of GPT-3
- Conclusions
- Questions and discussion

BENEFITS AND LIMITATIONS



FIRST PRINCIPLE OF CHAT BOT

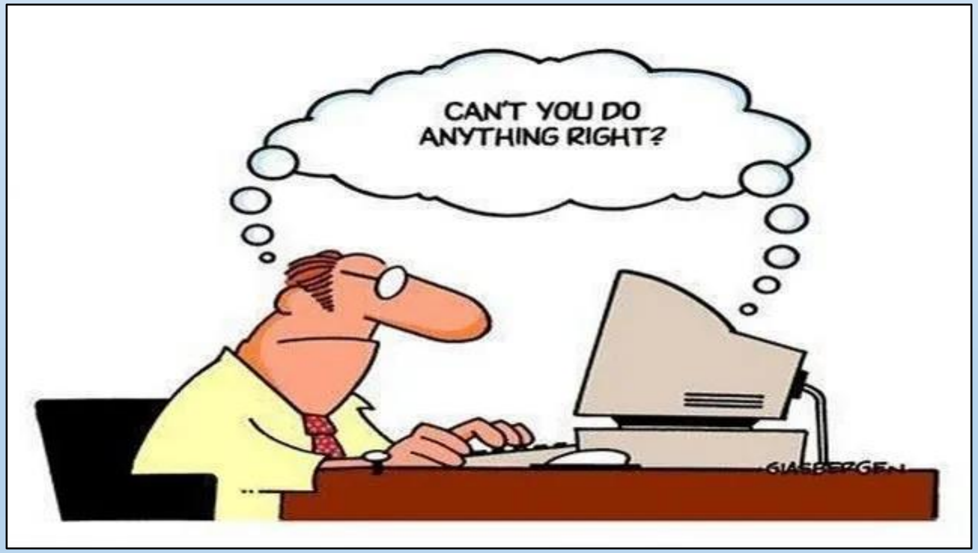
"The quality of the answers is directly proportional to the quality of the questions."

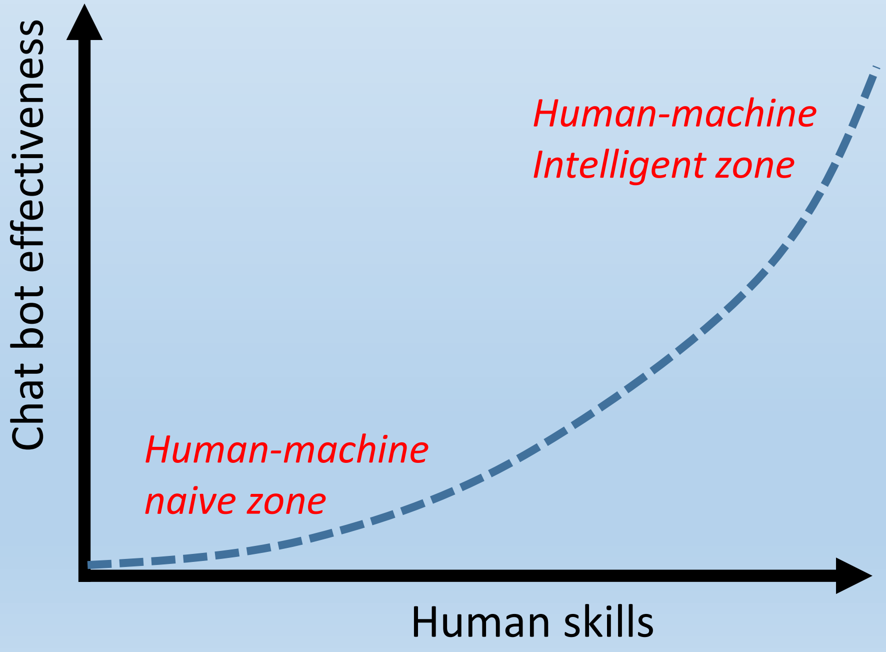


Quality of answers
Quality of questions
well-posed question:
«*What are the building blocks of a Python code for …?*»
Ill-posed question:
«*What is the sense of life?*»

A NON-OBVIOUS DERIVATION

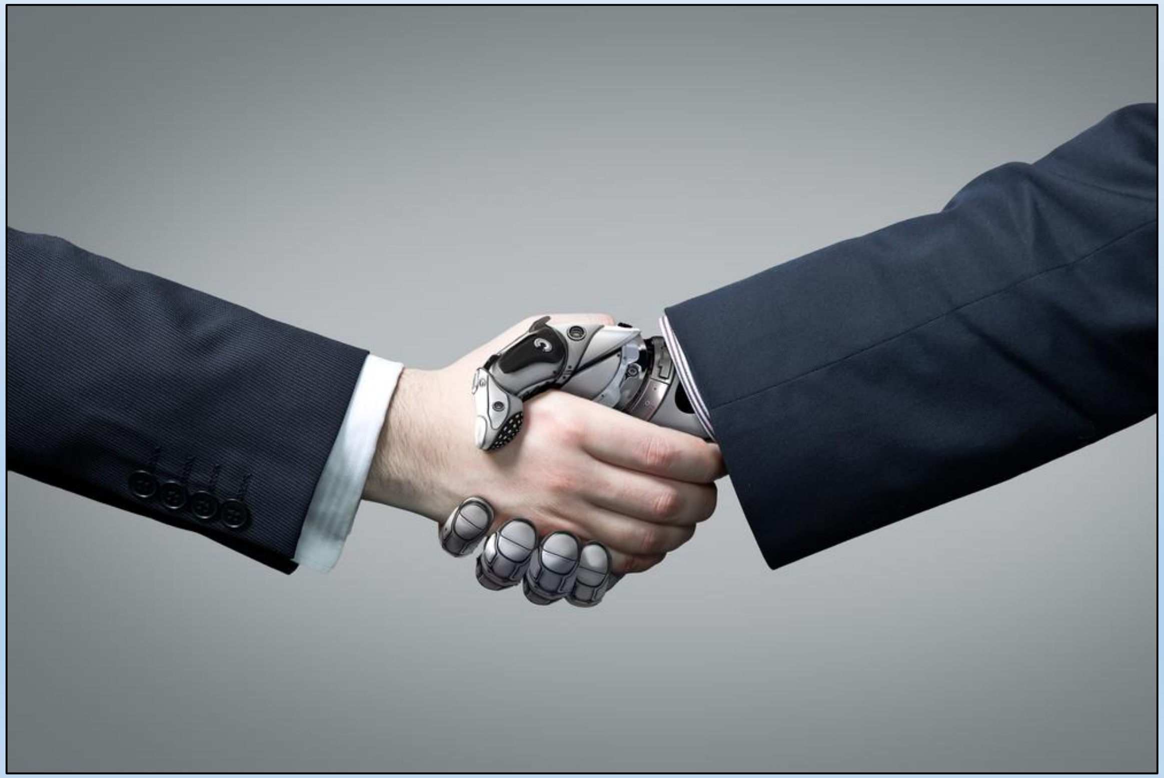
"The effectiveness of a chat bot depends non-linearly on the skills of the user."





AN IMPORTANT REMARK

The focus is on the cooperation between artificial and biological intelligence

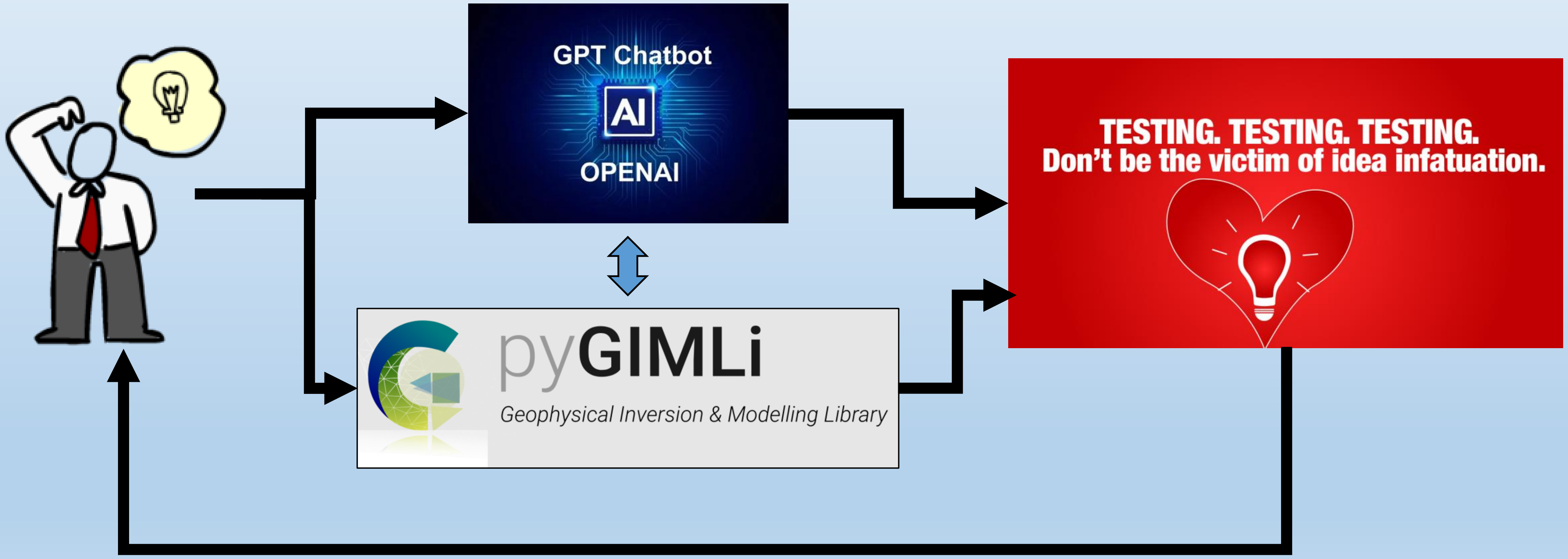


OUTLINE

- Introduction
- Examples, examples, examples …
- **Applications in Geosciences … and in other fields**
- A critical discussion: benefits and limitations
- Creative use of GPT-3
- Conclusions
- Questions and discussion

An example of cooperation loop:

testing new ideas by combination of GPT with existing libraries

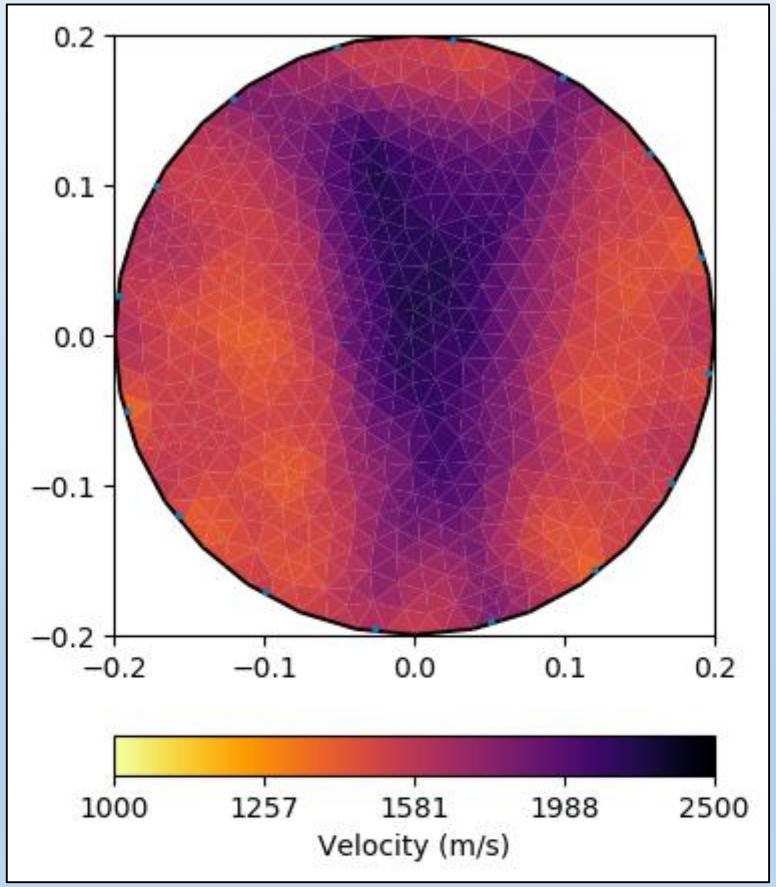

GPT Chatbot
AI
OPENAI
pyGIMLi
Geophysical Inversion & Modelling Library
TESTING. TESTING. TESTING.
Don't be the victim of idea infatuation.

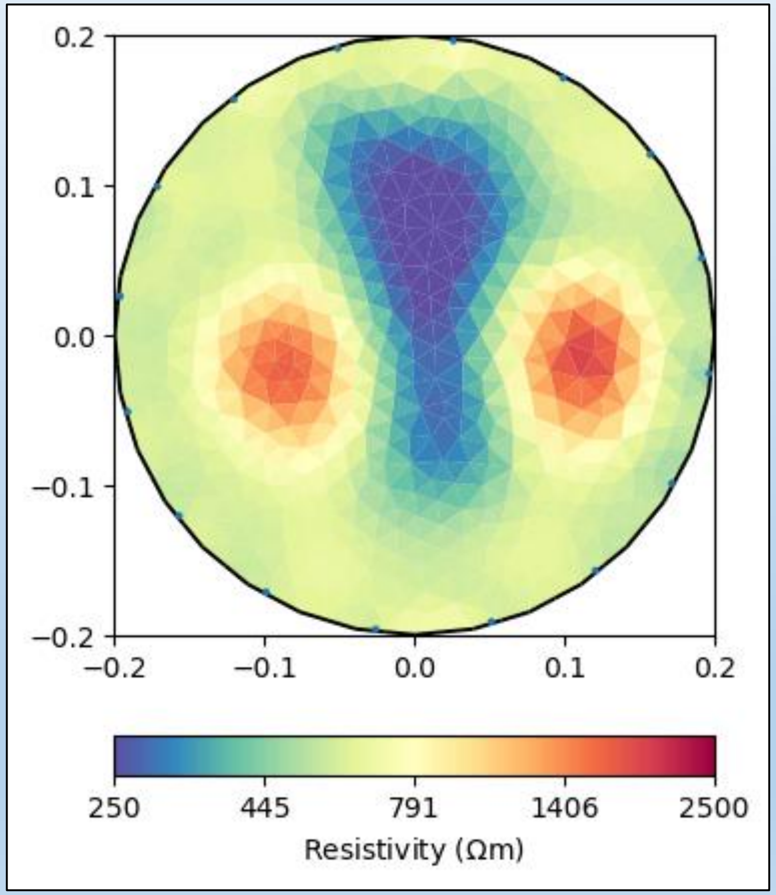
Joint inversion of refraction seismic and geo-electric data

pyGIMLi

Geophysical Inversion & Modelling Library

Credit: https://www.pygimli.org/_examples_auto/5_misc/plot_petro_joint_inv.html





Joint inversion of refraction seismic and geo-electric data

The code imports the input data files using the pg.load() function.

It creates a 2D mesh for the inv. using pg.createGrid() function.

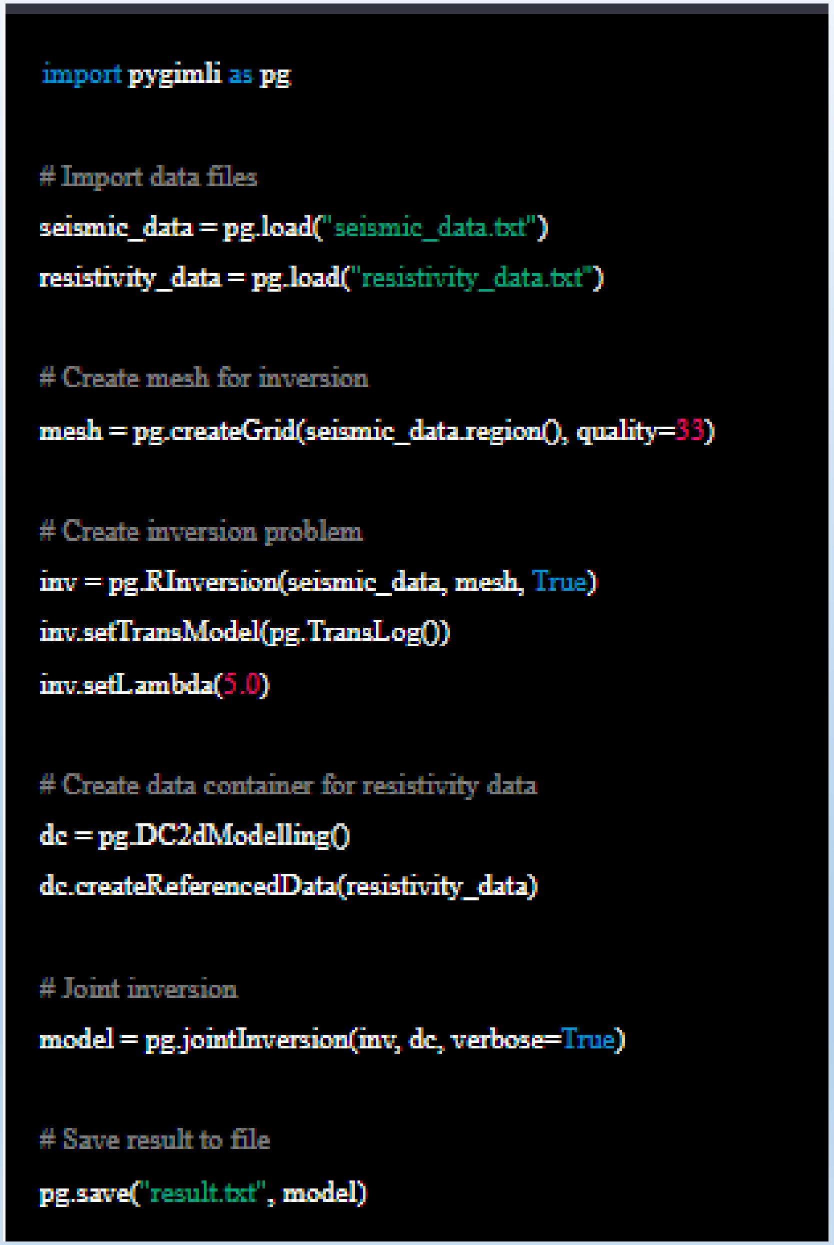
The inversion problem is created using the pg.RInversion() function, with the seismic travel time data and the mesh. The transmissivity model is set using the setTransModel() function.

The regularization parameter is set using the setLambda() function.

The code also creates a data container for the resistivity data using the pg.DC2dModelling() function.

The joint invers. is performed using pg.jointInversion() function.

The result is saved to a file using the pg.save() function.



```
import pygimli as pg

# Import data files
seismic_data = pg.load("seismic_data.txt")
resistivity_data = pg.load("resistivity_data.txt")

# Create mesh for inversion
mesh = pg.createGrid(seismic_data.region(), quality=33)

# Create inversion problem
inv = pg.RInversion(seismic_data, mesh, True)
inv.setTransModel(pg.TransLog())
inv.setLambda(5.0)

# Create data container for resistivity data
dc = pg.DC2dModelling()
dc.createReferencedData(resistivity_data)

# Joint inversion
model = pg.jointInversion(inv, dc, verbose=True)

# Save result to file
pg.save("result.txt", model)
```

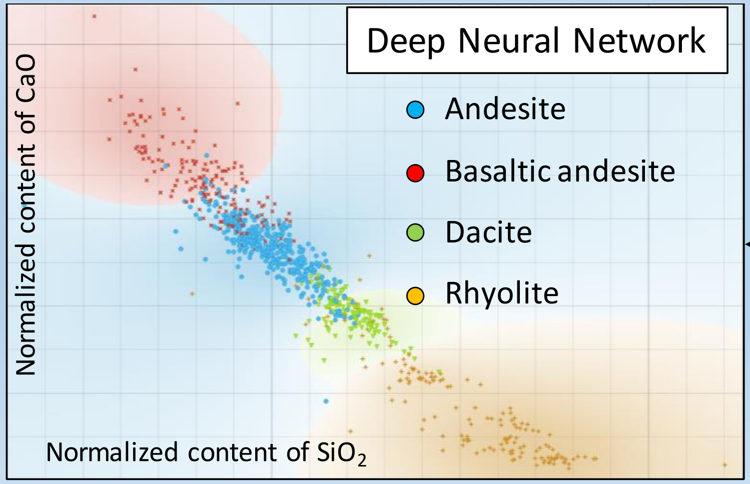
Additional applications in Geo-sciences using Machine Learning supported by GPT-3

Machine Learning for rock sample classification based on chemical composition

- SiO_2 — 72.04% (silica)
- Al_2O_3 — 14.42% (alumina)
- K_2O— 4.12%
- Na_2O — 3.69%
- CaO — 1.82%
- FeO — 1.68%
- Fe_2O_3 — 1.22%
- MgO — 0.71%
- TiO_2 — 0.30%
- P_2O_5 — 0.12%
- MnO — 0.05%

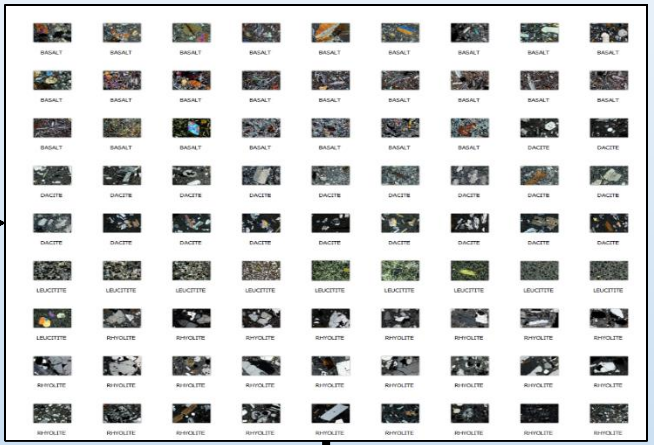
ML &

GPT-3



Machine Learning + GPT-3 for mineralogical recognition

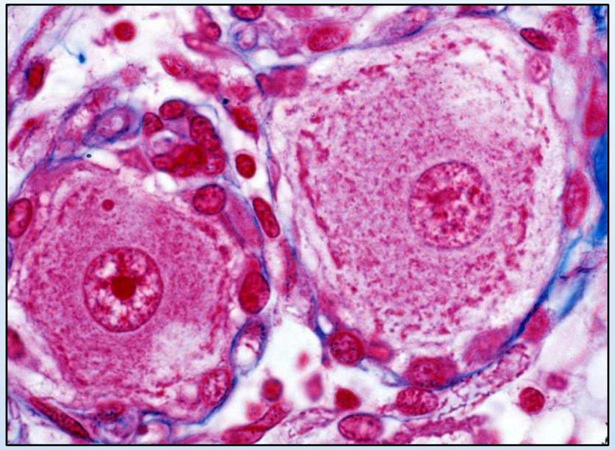
Machine training



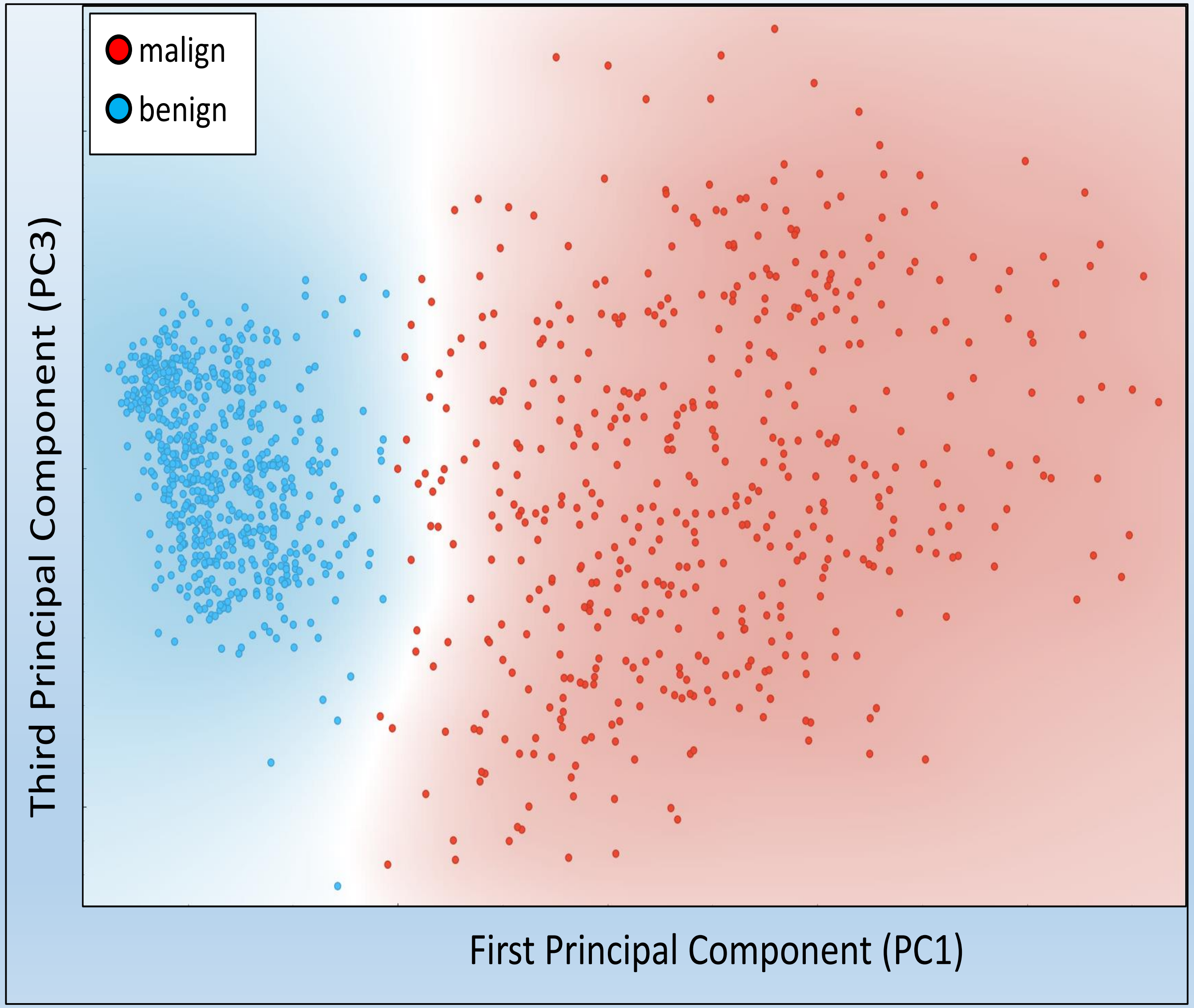
Automatic classification

Applications in Medical Sciences using Machine Learning supported by GPT-3

Medical diagnosis



ML & GPT-3

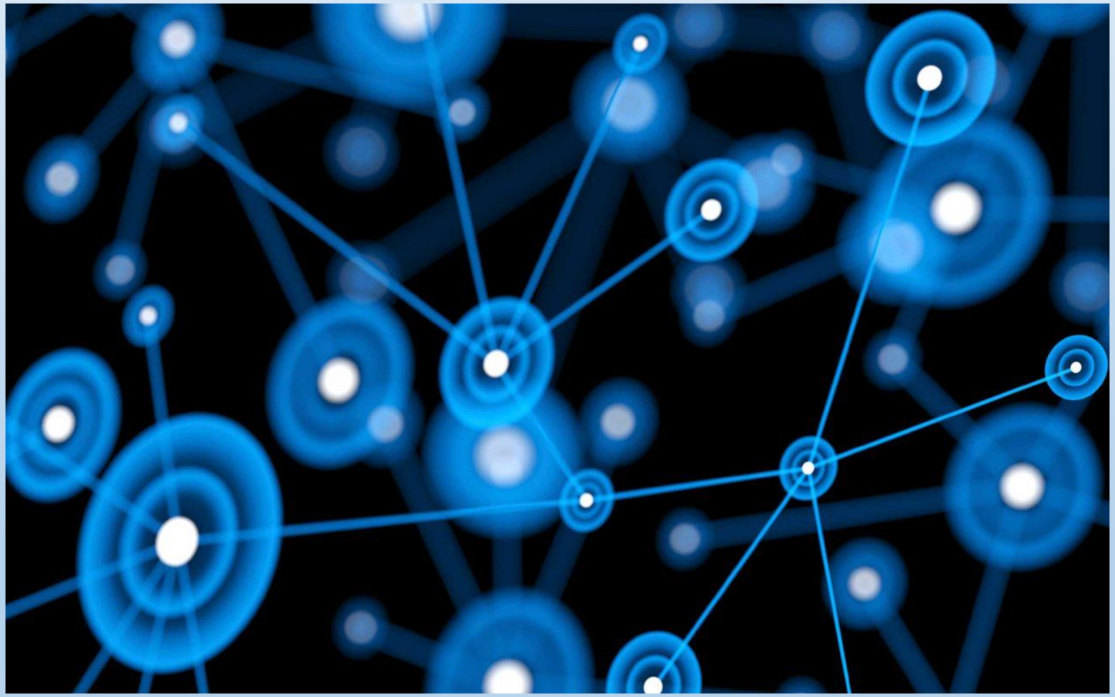


OUTLINE

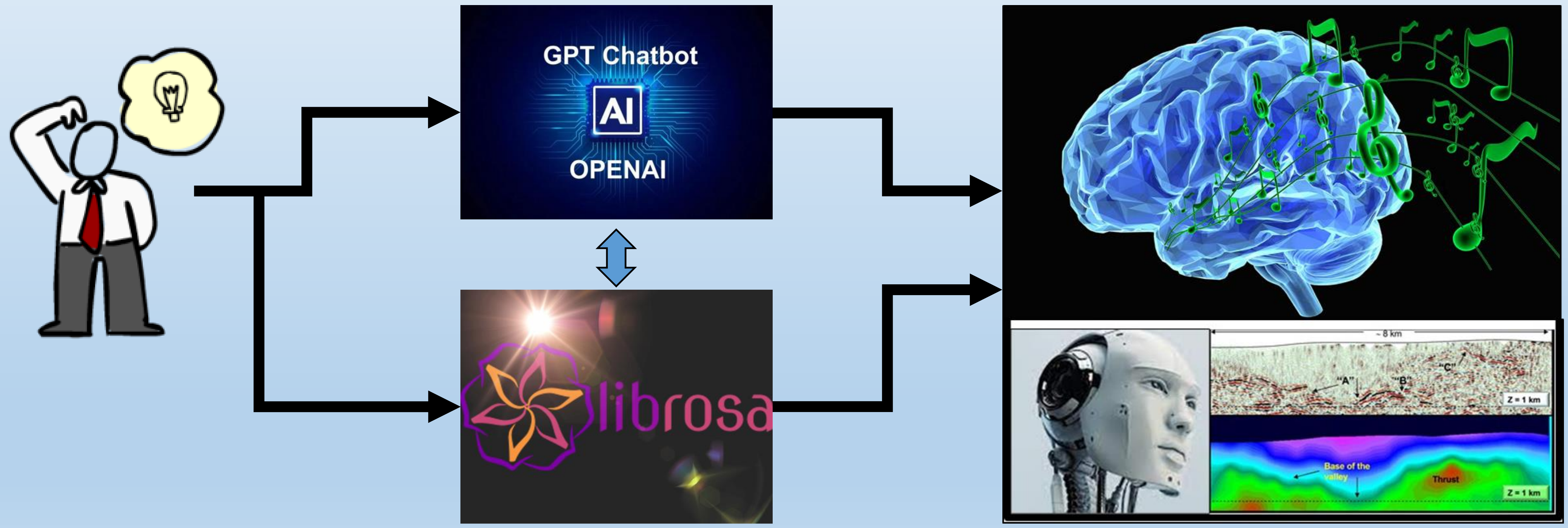
- Introduction
- Examples, examples, examples …
- Applications in Geosciences
- A critical discussion: benefits and limitations
- **Creative use of GPT-3**
- Conclusions
- Questions and discussion

GPT-3 SUPPORTING HYBRID TECHNOLOGIES

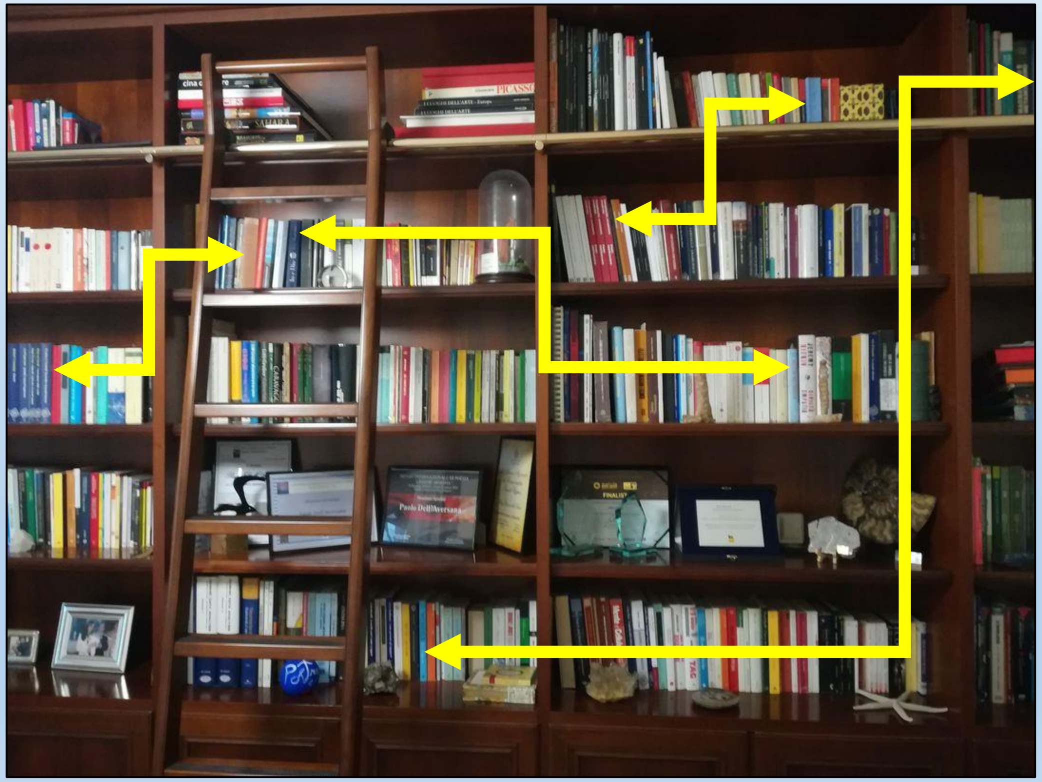
Using GPT-3 for linking different scientific domains



Linking Geophysics, Machine Learning, Sound Engineering and Cognition using “Librosa” libraries



GPT-3


Connecting book content through GPT-3

COMBINATION AND SELECTION IS ALL YOU NEED.

I am testing GPT-3 and Transformer technology as tools for inspiring/triggering/developing combinatorial creativity and innovation.

I created a complete workflow and a methodology for doing, at moment, two types of applications (see conceptual cartoon in the figure):

1) Correlating the semantic content of books (included in the GPT-3 knowledge base) for generating linked concepts, for inspiring new ideas and intuitions.

2) Exploring/combining patents databases, scientific literature, code-libraries (included in the GPT-3 knowledge base) and so forth for generating new algorithms, computer and math codes, new potential methods, inventions, patents from their semantic combination.

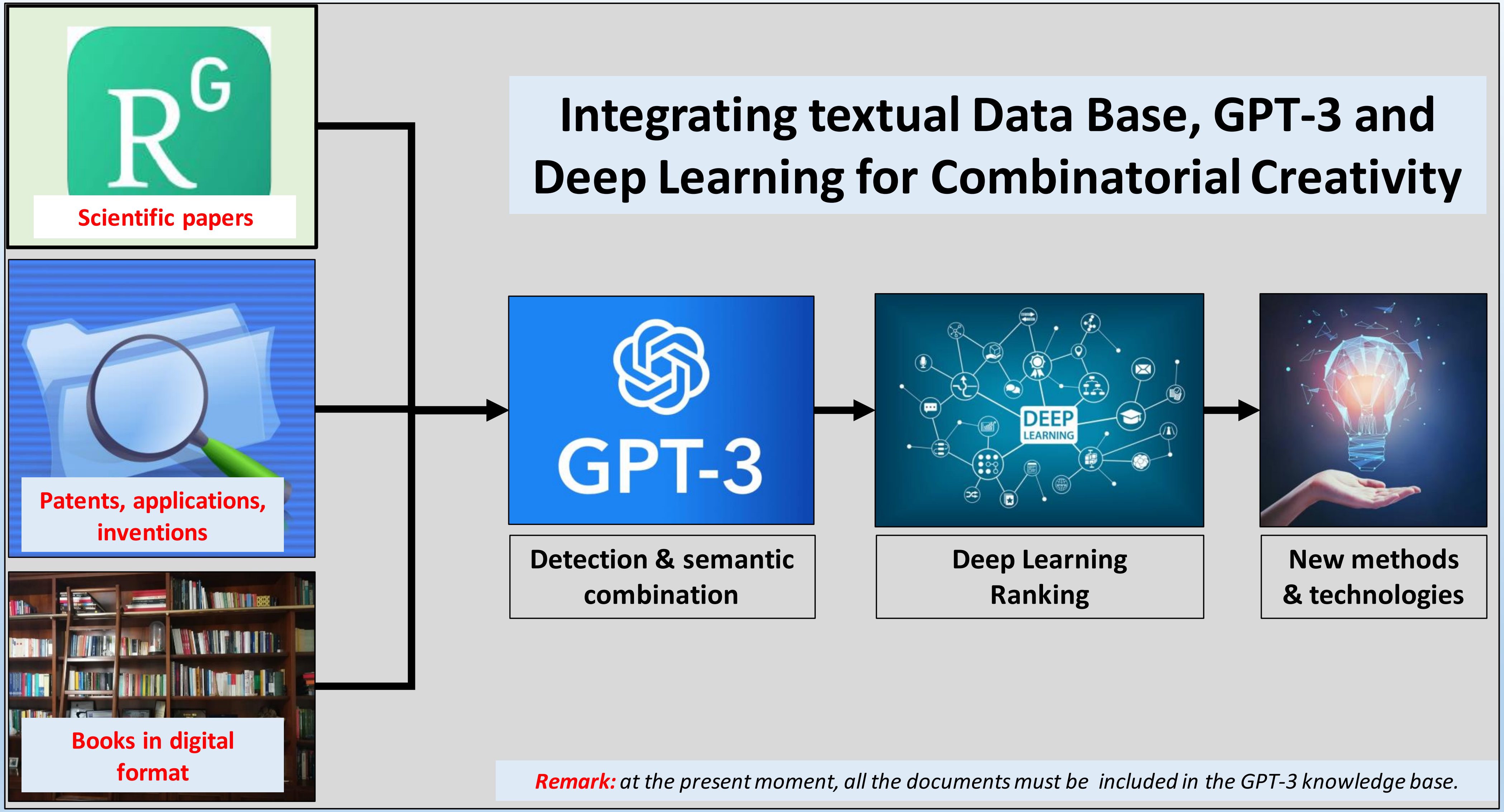
In about two months of trials, I have produced many new intriguing ideas, new algorithms and Python codes, several innovative workflows for signal analysis, image/sound classification, geophysical data interpretation and much more …

Next, all of these “products” have been selected through an automatic ranking process based on Deep Learning (depending on pre-defined criteria, tasks and purposes) and using well-known techniques (such as CNNs, RNNs, Transformers, Deep Boltzmann Machines, and various NLP algorithms).

In my tests, about 20% of the “products” survived to the selection process. The remaining 80% has been rejected or saved for future check because labelled as “obvious”, “not new” and/or “too challenging”.

The result is a significant number of new effective ideas, approaches, methodologies, technologies ready for verification tests in multi-disciplinary fields (geophysics, operation geology, drilling geo-steering, sound engineering, digital music, math, art, medical diagnosis, project management, financial analysis, sentiment analysis ...).

Integrating textual Data Base, GPT-3 and Deep Learning for Combinatorial Creativity


Scientific papers
Patents, applications, inventions
Books in digital format
GPT-3
Detection & semantic combination
DEEP LEARNING
Deep Learning Ranking
New methods & technologies

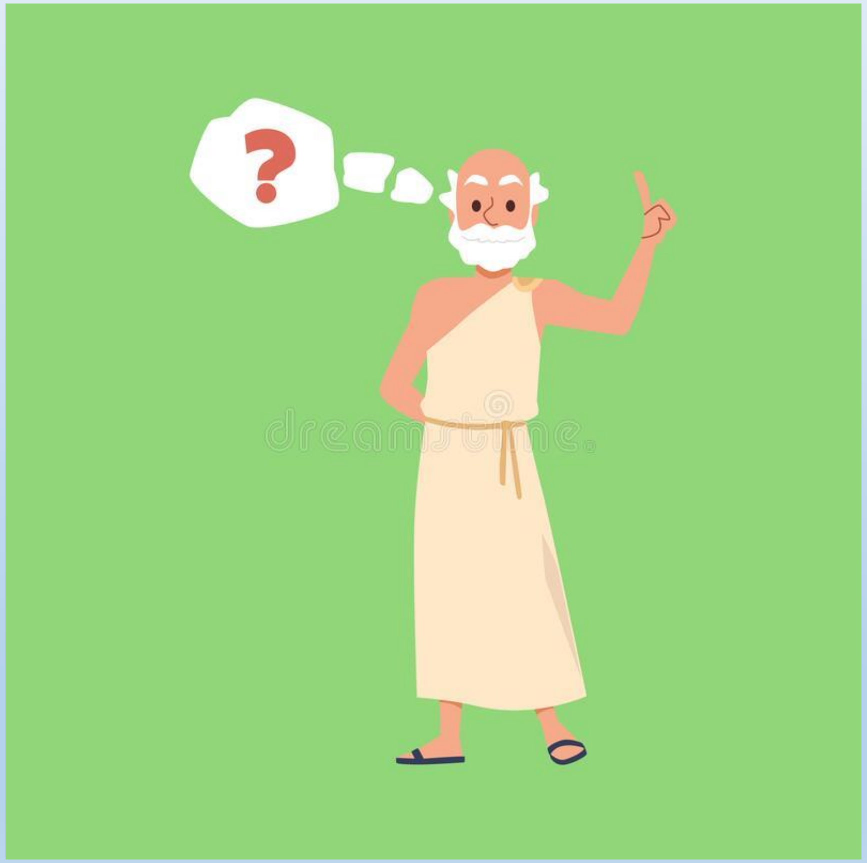
Remark: *at the present moment, all the documents must be included in the GPT-3 knowledge base.*

OUTLINE

- Introduction
- Examples, examples, examples …
- Applications in Geosciences
- A critical discussion: benefits and limitations
- Creative use of GPT-3
- **Conclusions**
- Questions and discussion

SOME OPEN QUESTIONS AND REMARKS

1) Where are we going?
2) Will AI replace humans?
3) Or has a new form of collaboration opened up between artificial and biological intelligence?
4) Has the era of artificial creativity begun?
5) Any other open question ?
6) …



Suggested readings

“**Attention Is All You Need**” by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, (2017). Advances in Neural Information Processing Systems, 30, 5998-6008. This is the reference paper for an introduction about Transformers architecture.

“**Language Models are Unsupervised Multitask Learners**” by Brown et al. (2020): This paper describes the original GPT model and its successors, GPT-2 and GPT-3. It discusses the design and training of the model and presents results on a variety of language tasks.

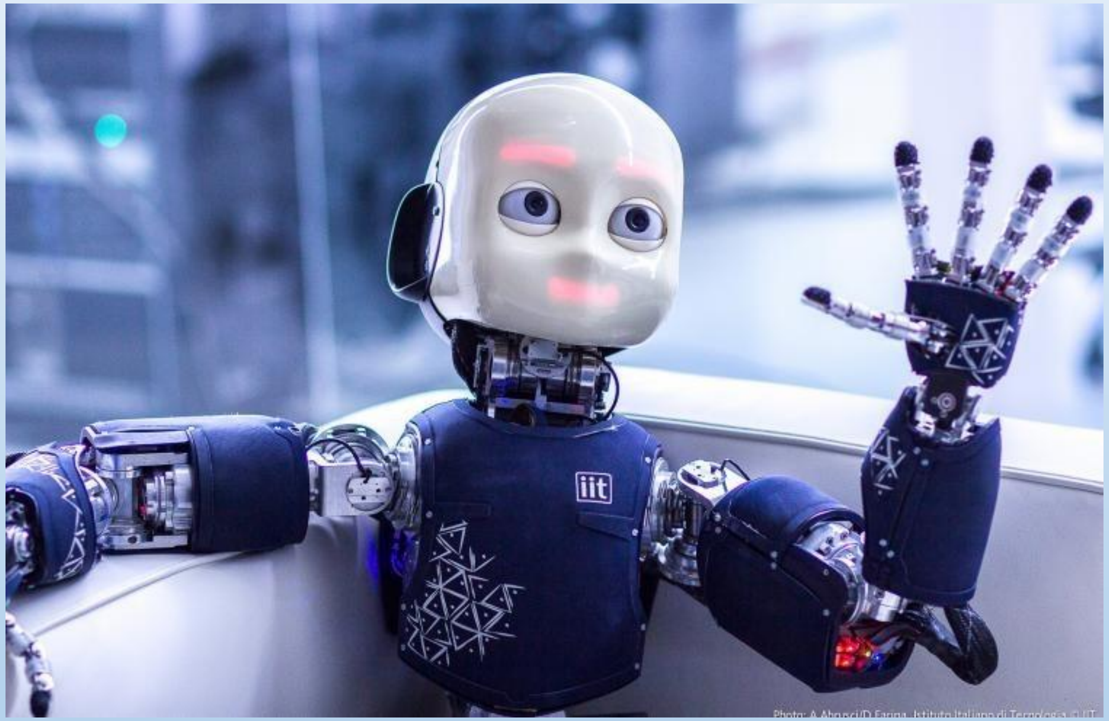
“**Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**” by Radford et al. (2019): This paper presents the results of fine-tuning GPT-2 on a variety of natural language processing tasks and shows that it can achieve state-of-the-art results on many of them.

“**Better Language Models and Their Implications**” by Brown et al. (2020): This paper presents the development and evaluation of GPT-3 and discusses its potential impact on various language tasks and applications.

“Inversion of geophysical data supported by Reinforcement Learning”, by Paolo Dell’Aversana, (2023). BGTA (in press) DOI DOI: 10.4430/bgo00411.

Dell’ Aversana. Paolo. https://www.researchgate.net/publication/368292736_GPT-3_a_new_cooperation_scenario_between_humans_and_machines_Benefits_and_limitations_of_GPT-3_as_a_coding_virtual_assistant

THANK YOU!


iit
Photo: A. Abrusci/D. Farina, Istituto Italiano di Tecnologia - IIT

OUTLINE

- Introduction
- Examples, examples, examples …
- Applications in Geosciences
- A critical discussion: benefits and limitations
- Creative use of GPT-3
- Conclusions
- **Questions and discussion**