

**Table S6.** The Python code used to generate descriptions of potential geodiversity sites mapped in Qgis.

Script for generation of descriptions of points of interest detected in GIS
<pre> # The code is adapted from the python-simplechat script (from the official repository of Ollama). # Original source code is available here: https://github.com/ollama/ollama/blob/main/examples/python- simplechat/client.py  import csv import json import requests  # Creates dictionary and replace id number by names of lithological classes  import pandas as pd dictionary = pd.read_csv(r'D:\\Dictionary.csv', sep='\t', encoding='utf-8') my_dict = dictionary.set_index("number").to_dict()  data = pd.read_csv(r'D:\\Selected-geodiversity-sites.csv', sep='\ t', encoding='utf-8')  data_en=data.replace({"type_1": my_dict['name_en']}) data_en=data_en.replace({"type_2": my_dict['name_en']}) data_pl=data.replace({"type_1": my_dict['name_pl']}) data_pl=data_pl.replace({"type_2": my_dict['name_pl']})  # Opening files to write generated descriptions out_en = open(r'D:\\Geodiversity-sites-generate-en.csv', 'w', newline='', encoding='utf-8') writer_en = csv.writer(out_en)  out_pl = open(r'D:\\Geodiversity-sites-generate-pl.csv', 'w', newline='', encoding='utf-8') writer_pl = csv.writer(out_pl)  # Ollama must be installed and running. model = "llama3:latest"  def chat(messages):     r = requests.post(         "http://127.0.0.1:11434/api/chat",         json={"model": model, "messages": messages, "stream": True},         stream=True     )     r.raise_for_status()     output = "" </pre>

```

    for line in r.iter_lines():
        body = json.loads(line)
        if "error" in body:
            raise Exception(body["error"])
        if body.get("done") is False:
            message = body.get("message", "")
            content = message.get("content", "")
            output += content
            # the response streams one token at a time, print that
            print(content, end="", flush=True)

        if body.get("done", False):
            message["content"] = output
            return message

def main():
    messages = []
    id = 0
    # Generating content in Polish
    for index, row in data_pl.iterrows():
        if id == 0:
            user_input = f"Napisz jedno zdanie opisujące miejsce,
gdzie jesteśmy, tak jak w przykładach poniżej. Nie dodawaj żadnego
komentarza, zwróć tylko jedno zdanie opisu. Oto przykłady:\nDane:
Jednostka 1: wysoczyzna morenowa; Jednostka 2: kem lub terasa
kemowa \nOpis: Znajdujemy się na granicy między wysoczyzną
morenową i kemem lub terasą kemową.\nDane: Jednostka 1: piaski
wodnolodowcowe; Jednostka 2: dno doliny\nOpis: Znajdujemy się w
miejscu, gdzie piaski wodnolodowcowe znajdują się obok dna
doliny.\nDane: Jednostka 1: dawny tunel podlodowcowy; Jednostka 2:
oz\nOpis: W tym miejscu oz sąsiaduje z dawnym tunelem
podlodowcowym.\nDane: Jednostka 1: wzgórze moreny czołowej;
Jednostka 2: wysoczyzna morenowa\nOpis: Jesteśmy pomiędzy
wzgórzami moreny czołowej i wysoczyzną morenową.\nTeraz Twoja
kolej:\nDane: Jednostka 1: " + row['type_1'] + '; Jednostka 2: ' +
row['type_2']
            else:
                user_input = 'Dane: Jednostka 1: ' + row['type_1'] +
'; Jednostka 2: ' + row['type_2']
                messages.append({"role": "user", "content": user_input})
                print (str(id) + "\n")
                message = chat(messages)
                messages.append(message)
                print("\n")
                writer_pl.writerow([row['X'], row['Y'], row['fid'],
row['type_1'], row['type_2'], message])
                id = id + 1

    # Generating content in English
    messages = []

```

```

id = 0
for index, row in data_en.iterrows():
    if id == 0:
        user_input = f"We are standing in a place located at
the boundary of two geological units. Return 1-2 sentences that
describe a place, following the examples below. Do not add any
comments, just write the descriptions. Try to avoid repeating
sentences. Here are some samples:\nInput: Geological unit 1:
moraine plateau; Geological unit 2: kame or kame terrace \
\nDescription: We are standing at the edge of a moraine plateau and
a kame or kame terrace.\n\nInput: Geological unit 1: moraine
plateau; Geological unit 2: former subglacial tunnel \
\nDescription: We are situated at the border of a moraine plateau
and a former subglacial tunnel.\nInput: Geological unit 1:
glaciofluvial sands; Geological unit 2: valley bottom\
\nDescription: In this area, the bottom of a valley borders the
glaciofluvial sands.\nInput: Geological unit 1: former tunnel
valley; Geological unit 2: esker\nDescription: At this place, an
esker neighbours the post-glacial, former tunnel valley.\nInput:
Geological unit 1: outwash plain; Geological unit 2: alluvial
terrace\nDescription: This is the spot where an outwash plain
meets an alluvial terrace.\nNow it is your turn:\nInput:
'Geological unit 1: " + row['type_1'] + ' Geological unit 2: ' +
row['type_2']
        else:
            user_input = 'Input: Geological unit 1: ' +
row['type_1'] + '; Geological unit 2: ' + row['type_2']
            messages.append({"role": "user", "content": user_input})
            print (str(id) + "\n")
            message = chat(messages)
            messages.append(message)
            print("\n")
            writer_en.writerow([row['X'], row['Y'], row['fid'],
row['type_1'], row['type_2'], message])
            id = id + 1
        out_en.close()
        out_pl.close()

if __name__ == "__main__":
    main()

```

### Script for generation of descriptions of viewpoints detected in GIS

```

# The code is adapted from the python-simplechat script (from the
official repository of Ollama).
# Original source code is available here:
https://github.com/ollama/ollama/blob/main/examples/python-
simplechat/client.py

import csv
import json
import requests

```

```

# Creates dictionary and replace id number by names of
lithological classes

import pandas as pd
dictionary = pd.read_csv(r'D:\\Dictionary.csv', sep='\t',
encoding='utf-8')
my_dict = dictionary.set_index("number").to_dict()

data = pd.read_csv(r'D:\\Selected-viewpoints.csv', sep='\t',
encoding='utf-8')

data_en=data.replace({"type": my_dict['name_en']})
data_pl=data.replace({"type": my_dict['name_pl']})

# Opening files to write generated descriptions
out_en = open(r'D:\\Viewpoints-generate-en.csv', 'w', newline='',
encoding='utf-8')
writer_en = csv.writer(out_en)

out_pl = open(r'D:\\Viewpoints-generate-pl.csv', 'w', newline='',
encoding='utf-8')
writer_pl = csv.writer(out_pl)

# Ollama must be installed and running.
model = "llama3:latest"

def chat(messages):
    r = requests.post(
        "http://127.0.0.1:11434/api/chat",
        json={"model": model, "messages": messages, "stream":
True},
        stream=True
    )
    r.raise_for_status()
    output = ""

    for line in r.iter_lines():
        body = json.loads(line)
        if "error" in body:
            raise Exception(body["error"])
        if body.get("done") is False:
            message = body.get("message", "")
            content = message.get("content", "")
            output += content
            # the response streams one token at a time, print that
as we receive it
            print(content, end="", flush=True)

        if body.get("done", False):
            message["content"] = output
            return message

```

```

def main():
    messages = []
    id = 0
    # Generating content in Polish
    for index, row in data_pl.iterrows():
        if id == 0:
            user_input = f"Znajdujemy się na wzniesieniu. Napisz
1-2 zdania o tym miejscu, tak jak w przykładach poniżej. Nie
dodawaj własnych komentarzy. Oto przykłady:\nDane: Wysokość:
130,20; Forma terenu: wysoczyzna morenowa \nOpis: Wzniesienie, na
którym jesteśmy, jest częścią wysoczyzny morenowej i wznosi się na
wysokość około 130 m n.p.m.\nDane: Wysokość: 91,80; Forma terenu:
oz\nOpis: Stoimy na ozie. Ta polodowcowa forma terenu sięga tu na
wysokość ponad 90 m n.p.m.\nDane: Wysokość: 115,20; Forma terenu:
kem lub terasa kemowa\nOpis: To wzniesienie (mające około 115 m
n.p.m.) znajduje się na kemie lub terasie kemowej.\nDane:
Wysokość: 94,10; Forma terenu: wzgórza moreny czołowej \nOpis:
Jesteśmy na wzgórzu moreny czołowej. Wznosi się ono na wysokość
niecałych 95 m n.p.m.\nTeraz Twoja kolej:\nDane: 'Wysokość: " +
str(row['elevation']).replace(".",",") + ' Forma terenu: ' +
row['type']
            else:
                user_input = 'Dane: Wysokość: ' +
str(row['elevation']).replace(".",",") + '; Forma terenu: ' +
row['type']
            messages.append({"role": "user", "content": user_input})
            print (str(id) + "\n")
            message = chat(messages)
            messages.append(message)
            print("\n")
            writer_pl.writerow([row['X'], row['Y'], row['fid'],
row['elevation'], row['type'], message])
            id = id + 1

    # Generating content in English
    messages = []
    id = 0
    for index, row in data_en.iterrows():
        if id == 0:
            user_input = f"We are standing at the hill. Return 1-2
sentences that describe a place, following the examples below. Do
not add any comments, just write the descriptions. Try to avoid
repeating sentences. Here are some samples:\nInput: Elevation:
130.20; Geological unit: moraine plateau \nDescription: The
elevation we are standing on belongs to a moraine plateau and is
approximately 130 metres above sea level.\nInput: Elevation:
91.80; Geological unit: esker\nDescription: We are standing on an
esker. This elevated glacial landform is more than 90 metres above
sea level.\nInput: Elevation: 115.20; Geological unit: kame or
kame terrace\nDescription: This elevated point (about 115 metres
a.s.l.) is located on a kame or a kame terrace.\nInput: Elevation:

```

```

130.20; Geological unit: moraine plateau \nDescription: The height
we are on is part of a moraine plateau and is around 130 meters
above sea level.\nInput: Elevation: 130.20; Geological unit:
moraine plateau \nDescription: We find ourselves on a moraine
plateau that is approximately 130 meters above sea level.\nNow it
is your turn:\nInput: 'Elevation: ' + str(row['elevation']) + '
Geological unit: ' + row['type']
    else:
        user_input = 'Input: Elevation: ' +
str(row['elevation']) + '; Geological unit: ' + row['type']
        messages.append({"role": "user", "content": user_input})
        print (str(id) + "\n")
        message = chat(messages)
        messages.append(message)
        print("\n")
        writer_en.writerow([row['X'], row['Y'], row['fid'],
row['elevation'], row['type'], message])
        id = id + 1
    out_en.close()
    out_pl.close()

if __name__ == "__main__":
    main()

```

### Script for generation of descriptions of landforms and sediment types

```

import csv
import ollama
import pandas as pd

source = pd.read_csv(r'C:\\Users\\pwlw\\Desktop\\Pawel\\Dane_GIS\\
Geopark-Morasko\\Automat\\Dictionary-descriptions-tymcz.csv',
sep='\\t', encoding='utf-8')
out = open(r'C:\\Users\\pwlw\\Desktop\\Pawel\\Dane_GIS\\Geopark-
Morasko\\Automat\\Landforms-descriptions-generate.csv', 'w',
newline='', encoding='utf-8')
writer = csv.writer(out)

for index, row in source.iterrows():
    print(str(row['number']) + '\\n')
    input_en = row['desc_en']
    input_pl = row['desc_pl']
    name_en = row['name_en']
    name_pl = row['name_pl']
    count = str(row['count'])
    output_en = ollama.generate( model="llama3:latest",
prompt=f"You are a scientific journalist. You will receive a long
description of the following landform or sediment: {name_en}.
Prepare {count} shorter, three to four sentences long, different
summaries of the provided text. Do not use other sources, just use
the long description that is provided. Here comes the long
description:\\n{input_en}")

```

```
print(output_en['response'])
output_pl = ollama.generate( model="llama3:latest",
prompt=f"Jesteś dziennikarzem naukowym. Otrzymasz długi opis na
temat: {name_pl}. Przygotuj {count} krótszych, różnych streszczeń
długiego opisu. Używaj tylko języka polskiego. Nie korzystaj z
innych źródeł, tylko z otrzymanego tekstu. To jest ten długi
opis:\n{input_pl}")
print(output_pl['response'])
writer.writerow([row['number'], name_en, name_pl,
output_en['response'], output_pl['response']])

out.close()
```