Master Thesis

submitted within the UNIGIS MSc programme
Interfaculty Department of Geoinformatics - Z_GIS
University of Salzburg

How ChatGPT can (not) help you with your GIS tasks

by

Daniel Marchetti
u106973

A thesis submitted in partial fulfilment of the requirements of the degree of
Master of Science – MSc

Advisor:

Leitner Michael, Dipl.-Ing. (FH) Dr. MSc.

Wien, 02.12.2023

List of Figures

Contents

1. Abstract

ChatGPT is an artificial intelligence (AI)-based large language model (LLM) that is capable of synthesizing human language. It is able to converse with the user and answer questions on a wide range of topics. In addition, it is able to generate programming code in several common programming languages such as Java, C++, and Python. The applications of this software are diverse, ranging from education to software development to copywriting. This thesis investigates ChatGPT capabilities to solve and automate tasks in geoinformatics using the popular open-source software QGIS, as well as its python programming-interface.

For the experiments, the complexity of the tasks is assessed. The complexity score is calculated via a weighted evaluation matrix. The inputs are the number of different geoprocessing tools used and the amount of input required for them for an already known solution. The conversation with ChatGPT takes place via a defined prompt structure, where all the available data are described file by file and the desired output explained. ChatGPT is asked to provide a workflow and a Python script for the solution. The presented answers are then evaluated with a weighted evaluation matrix.

The evaluation of the experiments has shown that ChatGPT always answers the question completely, regardless of the complexity of the question. If it is not equipped with the necessary information, it will make up a convincing, but factually wrong answer. When providing guides and workflows it is able to provide truthful information in the majority of cases. Although small errors such as made-up menu options are a common occurrence. When asked to provide Python scripts to automate the task, it responded with a complete set of programming-code that, at first glance, appears to be a working solution of the task. However, in no test did the code provided produce the desired result. In the majority of cases, the code did not run and therefore failed to produce any output. The reasons for the failure of the code are not due to a single common error, but many different ones. The variety of the errors is huge and range from simply calling functions with the wrong number of arguments, to using no existing (or made up) libraries.

ChatGPT's ability to solve complete tasks is limited and the information provided unreliable. Despite its limitations, ChatGPT can provide useful assistance to a GIS-specialist. Its ability to provide steps to a possible solution and relevant tools is beneficial for users who have the ability to integrate ChatGPT's suggestions into their own work. While ChatGPT is not able to automate tasks on its own, it can provide valuable code snippets to be used in the users' own code-solution. While limited at the moment, with further development of AI, the improvement of its algorithms and the extension of its training data, the area of its application might expand in the future.

2. Introduction

2.1. What is ChatGPT

ChatGPT is a generative AI that synthesises Human Language, which a user can converse with. It has a vast range of training data, so it is able to provide answers for a wide range of topics. In addition, it is also trained on synthesising programming code which it can generate according to the users' specifications.

The first version GTP-1 was made available to the public in June 2018 for free on the official Website of OpenAI. This was later updated to Version 2 and 3. To operate the text generation abilities of the GTP-API, in depth knowledge was required from the user.

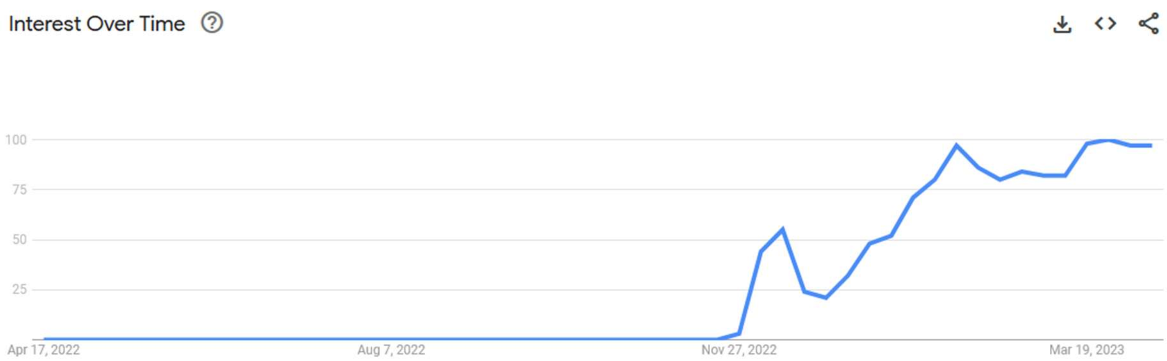

Figure 1 – ChatGPT on Google Trends

In December 2022 OpenAI released ChatGPT providing users access to the GTP-3 API via an easy-to-use web client. This ease of access led to an enormous growth in popularity, as shown in Figure 1. Since then, ChatGPT has become a prevalent topic in scientific discussion. How much it will affect different scientific fields is open to debate. While it might as well have minimal long term impact, some speculate that the scientific work of entire fields may be done by ChatGPT in the future (Gordijn and Have, 2023).

Whatever it might bring in the future, it is presently used from generating abstracts (Else, 2023) to writing complete papers (Stokel-Walker, 2023). In reaction to these AI-generated papers *Nature* has published some ground rules that include that it will reject all future works that list an AI as an author (Nature editorial, 2023).

2.2. A short overview on how ChatGPT works

To understand what ChatGPT can effectively do and how to use it to its potential, it is important to understand the underlying working principle of the software.

ChatGPT is a large language model or LLM. LLM's were introduced in the 2017 paper "Attention is All You Need" (Vaswani *et al.*, 2017). In its simplest terms LLM's are neural networks that are trained to predict the next word of a text. You can run these sequentially to not just get a single word but a whole text-block.

The part of the ChatGPT user experience, where the text is slowly typed out, is not for show. It is the neural network trying to guess the next word in real time, with the processing power assigned to the user. To come up with the next word in the sequence, the text is passed through a cascade of transformers. Each transformer predicts the most likely next word and passes it on to the next transformer, with the output of the final transformer printed out to the user. Depending on the structure of the LLM there can be a different number of transformers, with the most successful LLM's, like OpenAI's ChatGPT and Googles Palm containing a bit over 100 of them, with each transformer trained to predict different types of word associations.

Each transformer consists of an attention layer and a prediction layer. The attention layer assigns a weight to each word of the text and passes the words and their weight values on to the prediction layer. It basically points out which words are important to this particular transformer. The prediction layer now finds words that are most often associated with the input words, with higher weight words adding more to the probability than lower weight ones. A word with a high association probability is then passed to the next transformer. Note that this is not necessarily the word with the highest probability. Popular models choose one of the top probability words at random. That way a given input does not create a static output. Instead, a different answer is generated each time. Once the final transformer is run the resulting word is printed out and the whole text is then fed again into the transformer network to predict the next word in the sequence.

The above describes the basic principle of how the majority of popular AI-text generation models operate at the moment. The difference between them lies in how the transformers are designed and with what data they are trained.

2.3. On the intelligence of generative AI

Now that we looked behind the curtain and know that generative AI is just stringing up one word after another, according to some probability network, it begs the question: Is ChatGPT able to generate intelligent answers? The apparent intelligence is an emergent ability that emerges once a certain threshold of training data is processed in the model. Emergent abilities cannot be predicted simply by extrapolating the performance of smaller models (Wei *et al.*, 2022).

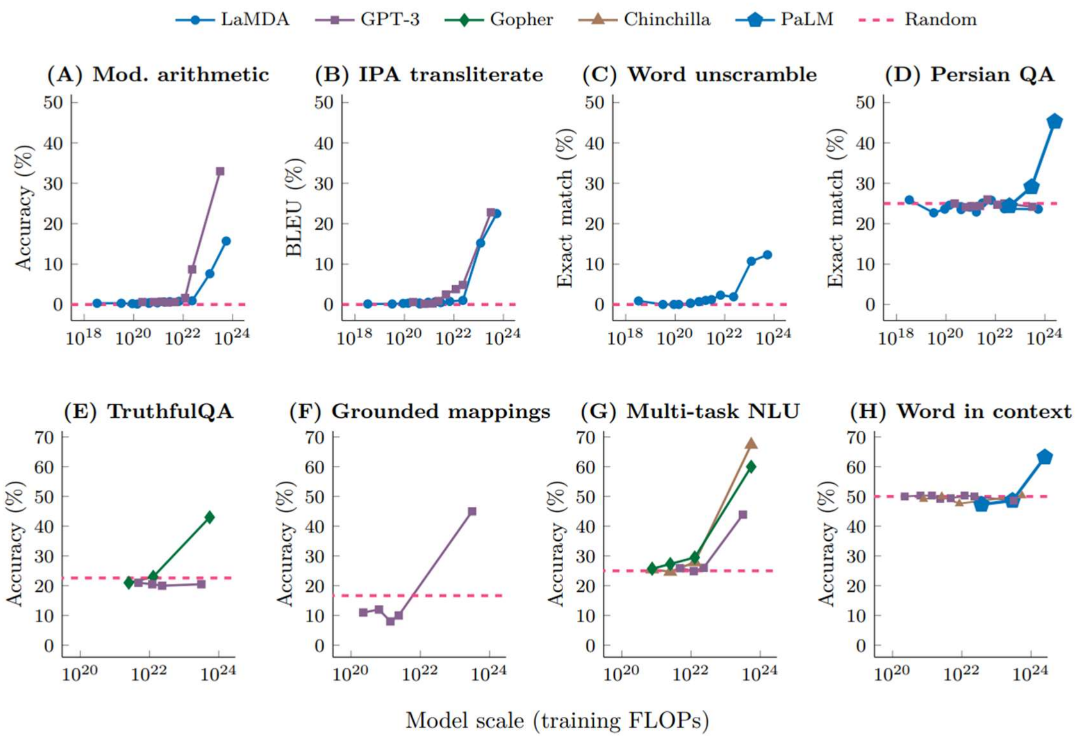

Figure 2 - Graph: LLM accuracy to model scale (Wei et al., 2022)

Figure 2 shows the accuracy of the word prediction on the y-axis and the amount of training data on the x-axis for different parameters. Note that each is a highly discontinuous function, where accuracy stays close to 0, which means that prediction quality is hardly better than random chance. Once certain threshold of training data is processed, prediction quality sharply increases. At this threshold the intelligence of the AI emerges, and it is then able to give productive answers.

2.4. ChatGPT in Geoinformatics

ChatGPT's training data comprise a huge portion of the visible internet, so it is safe to assume that it also learned a good deal about Geoinformatics in general. Popular geoinformatics software like QGIS, ArcGIS, and FME have all their documentation accessible online. They also possess a large and active userbase discussing numerous topics and problems. With that large amount of program-specific information and user-discussion online, it is another safe assumption that ChatGPT has learned, or is in the process of learning, about common workflows in these programs.

ChatGPT is not only trained to synthesize human-, but also programming-language. This includes python in general, but also PyQGIS specifically. So, generating code for GIS-automatization should also be within its capabilities.

There is certainly a big interest in ChatGPT in the field of Geoinformatics, but it has not yet found widespread use. Programming aids like GitHub's popular ChatGPT-powered assistant called "co-pilot" are still nowhere to be found in any mainstream GIS software.

This is however changing. FME already lets you connect to most major AI-System natively. Since programming assistants have quickly become a popular and sought-after feature in classic informatics software. The jump to geoinformatics might just be a question of time.

3. Aim and Objectives

3.1. Objectives

This work seeks to find out, to what degree ChatGPT can assist users of common Geoinformatics software. Its ability to provide workflows as well as its PyQGIS-scripting-solutions for a variety of tasks of different complexity levels, as well as its reliability, will be evaluated.

The aim is to find out if ChatGPT can provide meaningful assistance to common, but more difficult working tasks. It will be tested if, in addition to providing guidance, if it is also able to automate these tasks completely by providing working, but also user-readable PyQGIS-code with the appropriate functionality. ChatGPT might not use certain GIS functions properly and fail to provide suitable answers if these functions are required in the solution. The evaluation should identify these functions. Based on the findings the impact that ChatGPT capabilities might have on different users-groups should be assessed.

3.2. Research Questions

- **Does ChatGPT provide truthful information?**
  It should be tested if the information on hand is factually correct. This only works if each step in the workflow or code provided does what ChatGPT claims it does, not if what it does makes any sense in consideration of the solution. Note that this means incomplete or wrong solutions, can be factually correct.

- **Can ChatGPT provide useful assistance on GIS workflows?**
  It should be evaluated if the provided guide is comprehensible, easy to follow, and if the solution provided is working.

- **Is ChatGPT able to provide the necessary PyQGIS code to automate GIS workflows?**
  For a positive evaluation, the code should be executable and contain all necessary features.

- **Is the code ChatGPT generates well structured, so a regular user can understand and modify it?**
  This question disregards the actual function of the code and only rates the code's structure and documentation. Note: A positive evaluation does not require the code to execute properly.

- **Does ChatGPT struggle with more complex issues?**
  Can ChatGPT only answer simple questions or is it able to answer questions where the solution requires multiple layers of steps.

- **Are there certain topics/solution-steps where ChatGPT falls flat?**
  Can certain criteria be identified where ChatGPT fails repeatedly to provide proper answers. The algorithm might fail to grasp the logic of certain types of GIS operations. It is also possible that ChatGPT does not have sufficient or conflicting training data on certain functions and/or topics. If possible, these criteria should be identified.

- **Overall, is ChatGPT a useful tool for a GIS specialist?**
  Is ChatGPT useful and reliable enough to provide sufficient benefit for a GIS-specialist so that its worth integrating it into the work routine.

4. Related Works

ChatGPT, as of this writing, is not even a year old. This means the process of assessing ChatGPT's capabilities, as well as its impact on different scientific fields, has only been started. This means that only a limited amount of related works exists at this moment, and barely any in regards of Geoinformatics. Through my research I have found that the most covered angle is ChatGPT's impact on software developing and its impact on academia and education.

Regarding its role in software developing "Role of ChatGPT in Computer Programming" comes to the conclusion that ChatGPT is a powerful and versatile tool for computer programming that can support developers and users in a wide range of tasks (Biswas, 2023). While "Use Chat GPT to Solve Programming Bugs" notes that ChatGPT can be a useful tool for solving programming bugs, it is not a perfect solution (Surameery and Shakor, 2023). The predominant opinion by scientific papers and general forum discourse on *reddit* and *discord* is, that ChatGPT in its current form will not replace programmers, but it will play a big role as a programming aid and might make *stackoverflow* as the biggest exchange for code snippets superfluous.
"Role of ChatGPT in Computer Programming" is interesting to this research, since they specifically tested python-code generation, which is part of the designed testing procedure. In "Role of ChatGPT in Computer Programming" ChatGPT was able to generate the correct code in every single test, although it is important to note that it was only asked to provide isolated code snippets, which never exceed 20 lines of code. It was not tasked to generate complete routines, which it will be required to provide in this paper's research.

The impact of ChatGPT in academia has been covered greatly by N*ature*, where a paper was published criticising that N*ature* already published several papers, where ChatGPT was listed as co-author or even the main author in case of the self-referencing paper (GPT-3, Osmanovic-Thunström and Steingrimsson, 2022; Stokel-Walker, 2023). Finally, *Nature* published an editorial outlining its ground-rules regarding ChatGPT's use for any paper wanting to publish in the future. In summary *Nature* does not accept ChatGPT as an author. The reason given is that AI cannot take countability of its work, which is a ground requirement for authorship. Furthermore authors using LLM tools should document this use in the methods or acknowledgements sections (Nature editorial, 2023).
Considering the influence of *Nature* in Science as a whole, these ground rules are now the de-facto-standard.

The great worry of educational institutions is that pupils and scholars might pass of ChatGPT's work as their own. Basically, professors fear students might use it to cheat on online-exams and assignments. The paper "ChatGPT: The End of Online Exam Integrity" states that ChatGPT presents a significant threat to the integrity of online exams (Susnjak, 2022). While I agree with the conclusions reached in general, I think the impact might be far less severe than assumed. As mentioned in the paper, there are already tools available to check for AI-generated text and the creator of ChatGPT, OpenAI, already announced that they will bring their own

“ChatGPT-detector” at some point. Tools for plagiarism checks are already the norm in academia, so running a separate tool to check for AI-generated texts is actually not that much out of the ordinary. It is a safe bet that “AI-checks” will soon be integrated features of already deployed plagiarism-check software.
I think it is important to recognize that tools like ChatGPT are actively used by a variety of academic fields and industries. The proper skills to utilise these tools is important to stay competitive, so the use of these tools should rather be supported than supressed. Digital literacy education is of critical importance and has to include AI tools, which should be part of the curriculum (Rudolph, Tan and Tan, 2023)

The before mentioned possibility of students using ChatGPT to do their assignments has been put to the test in the highly interesting paper “Ch(e)atGPT?” (Stutz *et al.*, 2023) An Anecdotal Approach on the Impact of ChatGPT on Teaching and Learning GIScience”. For this experiment the researchers created a fake student and input each assignment of a geoinformatics course, specifically on python-programming, into ChatGPT and submitted the output for grading. The made-up student was able to pass the course, even receiving a full score on the first two assignments. It did struggle on the third and fourth more complex tasks but was still able to pass. They noted that the AI-detection tool “zeroGPT” was able to reliable detect the generated texts. The plagiarism software “Ouriginal” however was not able to. The takeaway is that ChatGPT can generate proper python code for GIS, but struggles at more complex tasks. The research done in this thesis work will provide more insight on these issues.

5. Methodology

5.1. Prompt Structure

To make results comparable between each other and in an effort to make experiments reproducible, the prompts are structured via the following syntax:

Prompt1:

```
I have the following data in QGIS:

- [filename1]
contains [polygons/polylines/points]
contains the attribute [field1]
contains the attribute [field2]
contains the attribute [field3]
…

- [filename2]
contains [polygons/polylines/points]
contains the attribute [field1]
contains the attribute [field2]
contains the attribute [field3]
…

[Description of the desired output]
Provide the workflow to achieve this.
```

After receiving the answer to the first prompt the second one is structured as follows:

Prompt2:

```
Provide a python script to achieve this.
Use D:\GIS\[experiment-number] as the working directory.
```

Note that the order in which information is given, as well as the exact wording used to provide the information does influence ChatGPT's answers. This is called prompt-compliance which is already a valued skill among tech workers. The paper "Conversational Process Modelling: State of the Art, Applications, and Implications in Practice" (Klievtsova *et al.*, 2023) investigates the limitations in ChatGPT's ability to extract tasks from prompts. It finds that ChatGPT especially struggles to break down a big task from the prompt text into sensible subtasks. Also, tasks that are only implied are not able to be recognized, as ChatGPT is not able to "read between the lines". This is a topic of research on its own. This thesis however does not delve further into this subject and instead argues that, if ChatGPT was able to work intelligently, it should be sufficient to give all necessary information to achieve a meaningful answer, regardless of wording or order of this information.

5.2. Scorecard

Each experiment is evaluated via a scorecard. It is designed to assess the complexity level of the task and the quality of the answer. Each experiment asks ChatGPT to create a workflow solution, as well as a code solution solving the same problem. These solutions are graded separately.

Evaluating the complexity of a question is not straight forward, but we have the benefit of only wanting ChatGPT to provide answers to problems that have known solutions. I postulate that a problem is more complex, the higher the complexity of the solution. Furthermore, it is assumed that a solution is more complex, the more different tools are needed to achieve the solution in demand.

So, to evaluate the complexity score, a few indicators are taken that represent commonly used functions that are needed in geospatial processing. Each indicator is scored against a measure. The score is an integer number and must be chosen by picking the category that describes the indicator of the task in question best. All measures have a minimum of 0, but a variant maximum, so each indicator is normalized to a float-value between 0 and 1 by the normalizing factor. Each indicator is given a weight, indicating its importance to the final score. The magnitude of the weight is derived from how difficult to solve tasks are, according to my personal user-experience as a professional GIS-User. This weight-factor is multiplied by the normalized score and output as the weighted score. The sum of all weighted scores is the total complexity score.

It is important to note that complexity is only scored not rated. This means no statement is made if a given task is simple or complex, since a threshold may proof impossible to be determined. Instead, a score for tasks should be compared relatively to each other, since a higher score does imply a higher complexity level for a particular task in comparison to another.

Besides the complexity of the question, the scorecard will also evaluate the quality of the answer. It is separated into the workflow and code solution, with both being calculated as per the system described above. In addition, the score is also mapped to a rating system. This rating is adapted from the US school grading system, where A-D are the passing grades, with A being excellent and D being sufficient and F being the failing grade. The score is translated into this rating, which is finally used to determine the quality of the answer.

5.3. Evaluation Matrix

The evaluation matrix is specifically designed to identify strengths and weaknesses regarding the individual indicators. It is not designed to give an overall evaluation of answer quality, which is already given by the summary of all answer quality ratings. Instead, the evaluation matrix maps each indicator's particular score to a percentage scale, where 0% represents a score of 0 and 100% presents the maximum achievable score for each individual indicator. (Note: Weight as a scaling factor does not influence these percentages. So, the percentage is representative of both score and weighted score.) The percentages are then presented visually. Indicators of the complexity score are mapped to a temperature colour gradient (blue to orange), where higher scored indicators are assigned a hotter colour. Indicators of the quality score are mapped to a traffic light gradient. Where red represents the worst scores (roughly representing failing scores), followed by yellow for medium scores (roughly representing passing scores) and green for top scores.

This condensed visual representation should be able to identify patterns, if any are present, of indicators greatly influencing the quality rating.

5.4. Experimental Design

By explicitly excluding refactoring, experiments are concluded by packaging all necessary information into a single question via the prompt structure described in chapter 5.1. For all files used, the necessary parts of the data-structure will be described per file. Any additional requirements of the expected output are clearly defined.

While an interesting topic for further research, this work shall not require improvising information, deduction, improvisation, or any form of guesswork from ChatGPT. Nor will it lay any form of trap such as providing a "red herring", such as superfluous information or surplus data. All information given will be factually correct and worded in the proper technical jargon. Only ChatGPT's immediate problem-solving skills are to be tested.

Tests will exclusively feature the QGIS environment, as both ChatGPT and QGIS are open access software with wide adoption. Tasks are based on, but not limited to, the QGIS-training-material (https://qgis.org/en/site/forusers/trainingmaterial/index.html), which is published under a Creative Commons ShareAlike 3.0 licence. (https://creativecommons.org/licenses/by-sa/3.0/).
The ChatGPT version used for the experiment is ChatGPT is GPT-3.5. For further information regarding the used version refer to chapter 9.1.

Each experiment features an explanation of the aim of the experiment and a description of the question structure. The Chatlog documenting the interaction with ChatGPT will be included.

Finally, each experiment is scored with the scorecard. The scoring is done be me personally and is based on my personal experience as a GIS-professional. While the scoring process is setup to be as objective as possible, readers should be aware of this possible bias.

While results will be presented after the experimentation is concluded, the scorecard, as well as the experiment documentation, will provide room for necessary additional information to be documented. Should certain assumptions arise, this additional documentation will be used for steering the aim of further experiments into that particular direction, to provide enough data to develop a valid hypothesis

6. Experiment Documentation and Scoring

6.1. Experiment: Basic geoprocessing (Buffer, intersect)

- **Question structure and aim:**

The question involves processing two shapefiles with commonly used geoprocessing tools. The low number of input variables and the basic tools required in this question makes this a very basic task. ChatGPT's ability to perform basic, but common geoprocessing is tested.

- **Chatlog:**

```
User
I have the following Data in QGIS:

- railway_station_Karnataka.shp
contains points
contains the attribute "disused"

- bangalore.shp
contains a polygon

The output should be:
Polygon that is a 1 km buffer around all points in railway_station_Karnataka where the
attribute "disused" is not 'yes'. The polygon should be inside bangalore.

Provide a workflow to achieve the output with the provided data.

ChatGPT
To achieve the desired output in QGIS, you can follow the following workflow:

1. Load the "railway_station_Karnataka.shp" shapefile by dragging and dropping it into the
QGIS canvas or using the "Add Vector Layer" button.

2. Load the "bangalore.shp" polygon shapefile in a similar manner.

3. Ensure that both layers are in the correct projection system. If they have different
projections, right-click on each layer, go to "Properties," navigate to the "Source" tab,
and select the appropriate projection.

4. Open the attribute table of the "railway_station_Karnataka" layer by right-clicking on
it and selecting "Open Attribute Table."

5. Locate the "disused" attribute column in the attribute table. Identify the values
assigned to the points and note down the value that represents "yes" (e.g., 'yes', 'Yes',
'YES', or any other variant).

6. Close the attribute table and proceed to create a new selection based on the desired
criteria. Go to the "Selection" menu at the top of the QGIS window and choose "Select by
Expression."

7. In the expression builder, construct the following expression:
   "disused" != 'yes' (Replace 'yes' with the value noted in step 5, if it differs)

8. Click on the "Select" button to apply the expression-based selection. Only the points
where the "disused" attribute is not 'yes' will be selected.

9. With the points selected, go to the "Vector" menu, select "Geoprocessing Tools," and
then choose "Buffer."

10. In the Buffer dialog, select the "railway_station_Karnataka" layer as the input layer
and set the buffer distance to 1 km. Make sure to select "Save to file" and provide a name
and location for the output buffer polygon.
```

11. Click "Run" to execute the buffer operation. The resulting buffer polygon will be saved as a new shapefile.

12. Now, to ensure that the buffer polygon is inside the "bangalore" polygon, go to the "Vector" menu, select "Geoprocessing Tools," and choose "Clip."

13. In the Clip dialog, select the buffered polygon shapefile as the input layer and the "bangalore" polygon shapefile as the clipping layer. Provide a name and location for the output clipped polygon.

14. Click "Run" to perform the clip operation. The resulting polygon will be saved as a new shapefile, which represents the 1 km buffer around the points in "railway_station_Karnataka" that meet the criteria and is located within the "bangalore" polygon.

15. You can now load and visualize the final output polygon on the QGIS canvas.

By following this workflow, you should be able to create the desired polygon that represents a 1 km buffer around the points in "railway_station_Karnataka" where the "disused" attribute is not 'yes' and is located within the "bangalore" polygon.

User
Provide a python script to achieve the output with the provided data, with D:\GIS\Exp1 as the working directory.

ChatGPT

To achieve the desired output using Python and the provided data, you can use the GDAL and Geopandas libraries. Here's an example code snippet that you can use:

python

```
import geopandas as gpd
from shapely.geometry import Point

# Set the working directory
working_directory = r"D:\GIS\Exp1"

# Load the railway_station_Karnataka shapefile
stations = gpd.read_file(working_directory + r"\railway_station_Karnataka.shp")

# Filter the stations based on the "disused" attribute
stations_filtered = stations[stations['disused'] != 'yes']

# Load the bangalore polygon shapefile
bangalore = gpd.read_file(working_directory + r"\bangalore.shp")

# Create a buffer around the filtered stations
buffer_size = 1000  # 1 km buffer
stations_buffered = stations_filtered.buffer(buffer_size)

# Clip the buffered stations to the bangalore polygon
stations_clipped = gpd.clip(stations_buffered, bangalore)

# Save the output polygon to a new shapefile
output_filename = working_directory + r"\output.shp"
stations_clipped.to_file(output_filename)
```

Make sure to replace the file paths with the actual paths to the shapefiles on your system. This code assumes that you have GDAL and Geopandas libraries installed. Also, ensure that the CRS of both shapefiles is the same before running the code.

- **Notes:**

The workflow is easy to follow and does work. Some steps are unnecessary and could be skipped. This concerns setting the CRS to be equal and saving the filtered layer to a new file.
The Code however does not run, since the overlay function of Geopandas is not implemented in QGIS. As a note: Posting the error message to ChatGPT does

prompt it to give an updated working version. However, this updated answer is not considered for the evaluation.

- **Complexity score:**

Complexity score						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.7**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	2	3.3
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	1	3.5
	Select by location	0...not used 1...used	1.00	2	0	0.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	0	0.0
	Sum			23		**6.8**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**0.0**
Total complexity score				51		**7.5**

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0...not understandable 1...intention is conveyed 2...clearly understandable 3...easy to follow guide	0.33	2	3	2.0
	Complete	0...problem wasn't solved 1...solved to some extend 2...minor details missing 3...solved to full extend	0.33	3	3	3.0
	Correct	0...incorrect 1...overall correct with minor errors 2...correct in all aspects	0.50	5	2	5.0
	Sum			10		**10.0**
Total workflow solution quality score				10		**10.0**
Total workflow solution quality rating		**8 to 10 ... A** **7 to 8 ... B** **6 to 7 ... C** **5 to 6 ... D** **<= 5 ... F**				**A**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0...no code structure or proper naming 1...code is structured and readable 2...proper naming of variables and functions 3...code properly documented	0.33	2	3	2.0
	Complete	0...problem wasn't solved 1...solved to some extend 2...minor details missing 3...solved to full extend	0.33	3	3	3.0
	Correct	0...incorrect 1...major errors 2...overall correct with minor errors 3...correct in all aspects	0.33	5	1	1.7
	Sum			10		**6.7**
Total code solution quality score				10		6.7
Total code solution quality rating		**8 to 10 ... A** **7 to 8 ... B** **6 to 7 ... C** **5 to 6 ... D** **<= 5 ... F**				**C**

6.2. Experiment: Basic geoprocessing (Buffer, Select by location)

- **Question structure and aim:**

Since ChatGPT could not produce a working script on first try in Experiment 1, because it used a function that is not implanted. This experiment tries a similar task that uses different geoprocessing tools in case the python-buffer function is an isolated problem.

- **Chatlog:**

```
I have the following data in QGIS:

- parcels.shp
contains polygons
contains the attribute s_access

- sewer.shp
contains lines

Save parcels.shp as parcels_new.shp. Then set the attribute s_access to 'yes' if the
polygon is less than 50m from a sewer. Otherwise set the attribute to 'no'

Provide a workflow that achieves this
ChatGPT

To achieve the desired workflow in QGIS, you can follow these steps:

1. Open QGIS and load the `parcels.shp` and `sewer.shp` layers into your project.

2. Right-click on the `parcels` layer in the Layers panel and select "Export" > "Save
Features As..." to save it as a new shapefile. Choose a name like `parcels_new.shp` and
specify the output directory.

3. In the "Save Vector Layer As" dialog, make sure to select "ESRI Shapefile" as the format
and set the CRS (Coordinate Reference System) if needed. Then click "OK" to save the new
shapefile.

4. Next, open the Processing Toolbox by going to "Processing" > "Toolbox" in the menu bar.

5. In the Processing Toolbox, search for the "Join attributes by location" algorithm and
double-click on it to open the dialog.

6. In the "Join attributes by location" dialog, set the following parameters:
   - Target layer: `parcels_new`
   - Join layer: `sewer`
   - Geometric predicate: `Within distance`
   - Join type: `Take attributes of the first matching feature only`
   - By default, the `Join layer field` should be set to the attribute that uniquely
identifies the sewer lines. If not, select the appropriate field.
   - Set the `Maximum distance` to `50` (assuming the distance unit is in meters).

7. Click "Run" to execute the algorithm. This will perform a spatial join between the
parcels and sewer layers based on the specified distance criteria.

8. Once the algorithm completes, the `parcels_new` layer will have additional attributes
from the `sewer` layer. However, we need to calculate the `s_access` attribute based on the
join results.

9. Open the attribute table of the `parcels_new` layer by right-clicking on it in the
Layers panel and selecting "Open Attribute Table."

10. If the `s_access` field doesn't exist, create a new field by clicking on the "Open
field calculator" button (usually represented by a calculator icon).

11. In the Field Calculator dialog, set the following parameters:
    - Output field name: `s_access`
    - Output field type: `Text (string)`
```

- Expression: Use the following expression to set the value of `s_access` based on the join results:
```
CASE
  WHEN "sewer_ID" IS NOT NULL THEN 'yes'
  ELSE 'no'
END
```
Replace `"sewer_ID"` with the field name that represents the sewer attribute obtained from the join. This field should have values only if a parcel is within 50 meters of a sewer.

12. Click "OK" to execute the field calculation. The `s_access` attribute will be populated with 'yes' or 'no' based on the distance criteria.

13. You can now save the changes to the attribute table and close it.

After following these steps, you should have a new shapefile called `parcels_new.shp`, where the `s_access` attribute is set to 'yes' if the polygon is within 50 meters of a sewer line and 'no' otherwise.

User
Provide a python script to achieve this.
Use D:\GIS\Exp2 as the working directory
ChatGPT

Certainly! Here's a Python script that achieves the desired workflow using the geopandas library:

python

```python
import geopandas as gpd

# Set the working directory
working_dir = r"D:\GIS\Exp2"
# Load the parcels shapefile
parcels_path = f"{working_dir}\\parcels.shp"
parcels = gpd.read_file(parcels_path)

# Load the sewer shapefile
sewer_path = f"{working_dir}\\sewer.shp"
sewer = gpd.read_file(sewer_path)

# Perform spatial join
join = gpd.sjoin(parcels, sewer, how="left", op="within", distance_col="distance")

# Set the s_access attribute based on the distance
join["s_access"] = join["distance"].apply(lambda x: 'yes' if x < 50 else 'no')

# Create a new GeoDataFrame with the desired attributes
parcels_new = join[["geometry", "s_access"]]

# Save the new shapefile
parcels_new_path = f"{working_dir}\\parcels_new.shp"
parcels_new.to_file(parcels_new_path)

print("Shapefile saved successfully as parcels_new.shp")
```

Make sure to replace D:\GIS\Exp2 with your desired working directory that contains the parcels.shp and sewer.shp files. The script reads the shapefiles, performs a spatial join, sets the s_access attribute based on the distance, creates a new GeoDataFrame with the desired attributes, and saves it as parcels_new.shp in the working directory.

Note that you'll need to have the geopandas library installed to run this script. You can install it using pip install geopandas.

- **Notes:**

The workflow solution does not work as stated. The “Join attributes by location”-tool does not have a “within distance” option.

Neither the QGIS-GUI function for the spatial join, nor the equivalent Geopandas function used for the code solution have this parameter, so the code solution fails to run as well.

- **Complexity score:**

Complexity score						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.7**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	1	1.7
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0
	Select by location	0...not used 1...used	1.00	2	1	2.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	2	3.0
	Sum			23		**6.7**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**0.0**
Total complexity score				51		**7.3**

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0...not understandable 1...intention is conveyed 2...clearly understandable 3...easy to follow guide	0.33	2	3	2.0
	Complete	0...problem wasn't solved 1...solved to some extend 2...minor details missing 3...solved to full extend	0.33	3	3	3.0
	Correct	0...incorrect 1...overall correct with minor errors 2...correct in all aspects	0.50	5	0	0.0
	Sum			10		**5.0**
Total workflow solution quality score				10		**5.0**
Total workflow solution quality rating		**8 to 10 ... A** **7 to 8 ... B** **6 to 7 ... C** **5 to 6 ... D** **<= 5 ... F**				**F**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0...no code structure or proper naming 1...code is structured and readable 2...proper naming of variables and functions 3...code properly documented	0.33	2	3	2.0
	Complete	0...problem wasn't solved 1...solved to some extend 2...minor details missing 3...solved to full extend	0.33	3	3	3.0
	Correct	0...incorrect 1...major errors 2...overall correct with minor errors 3...correct in all aspects	0.33	5	0	0.0
	Sum			10		**5.0**
Total code solution quality score				10		5.0
Total code solution quality rating		**8 to 10 ... A** **7 to 8 ... B** **6 to 7 ... C** **5 to 6 ... D** **<= 5 ... F**				**F**

6.3. Experiment: Simple join

- **Question structure and aim:**

This question tests ChatGPT's ability to perform table joins. This particular case tests the most basic join, which requires no form of pre-processing. As always, the data structure of each table is described. The primary keys are pointed out in the running text and are named identical in both tables.

- **Chatlog:**

```
User
I have the following data in QGIS:

- DKM.shp
contains polygons
contains the attribute GST

- EZ.csv
contains the attribute GST
contains the attribute EZ

The goal is to join the attribute EZ to the DKM Shape by matching the attribute GST in both
files. Provide the workflow to achieve this.

ChatGPT
To join the attribute EZ from the EZ.csv file to the DKM.shp Shapefile based on the
matching attribute GST, you can follow these steps in QGIS:

1. Open QGIS and make sure both the DKM.shp and EZ.csv files are added to the project.

2. Verify that the attribute names in both files are exactly the same. In this case, it is
"GST." Ensure that the attribute GST in both files contains matching values that can be
used for the join.

3. If necessary, save the project to avoid losing any unsaved work during the process.

4. Go to the "Processing" menu at the top of the QGIS interface and select "Toolbox" to
open the Processing Toolbox.

5. In the Processing Toolbox, search for the "Join attributes by field value" tool and
double-click on it to open the dialog box.

6. In the "Join attributes by field value" dialog box, select the DKM.shp file as the
"Input layer."

7. Select the EZ.csv file as the "Join layer."

8. Choose the attribute field "GST" in both the "Input layer field" and "Join layer field"
drop-down menus. This ensures that the join is based on the matching values in the GST
field.

9. Specify the "Output shapefile" where the joined data will be saved. Choose a suitable
location and filename for the output file.

10. Click on the "Run" button to execute the join operation.

11. After the join process completes, you will see a new shapefile with the joined
attributes added to the attribute table of the DKM.shp file. The new shapefile will be
saved in the location you specified.

12. Open the attribute table of the joined shapefile to verify that the attribute EZ has
been successfully joined to the DKM.shp file based on the matching GST values.

That's it! You have now joined the attribute EZ from the EZ.csv file to the DKM.shp
Shapefile using the attribute GST as the matching field.That's it! You have now joined the
attribute EZ from the EZ.csv file to the DKM.shp Shapefile using the attribute GST as the
matching field.
```

```
User
Provide a python script to achieve this.
Use D:\GIS\Exp3 as the working directory

ChatGPT
Certainly! Here's a Python script that you can use to achieve the attribute join between
the DKM.shp Shapefile and the EZ.csv file in QGIS:

python

import os
from qgis.core import QgsVectorLayer, QgsProject

# Set the working directory
working_directory = r'D:\GIS\Exp3'
os.chdir(working_directory)

# Define the paths to the Shapefile and CSV files
dkm_file = 'DKM.shp'
ez_file = 'EZ.csv'

# Load the Shapefile and CSV files as vector layers
dkm_layer = QgsVectorLayer(dkm_file, 'DKM', 'ogr')
ez_layer = QgsVectorLayer(ez_file, 'EZ', 'ogr')

# Check if the layers were loaded successfully
if not dkm_layer.isValid():
    print('DKM layer failed to load!')
    exit()
if not ez_layer.isValid():
    print('EZ layer failed to load!')
    exit()

# Specify the join fields
join_field = 'GST'

# Perform the attribute join
join_object = QgsVectorLayerJoinInfo()
join_object.setJoinFieldName(join_field)
join_object.setTargetFieldName(join_field)
join_object.setJoinLayerId(ez_layer.id())
join_object.setUsingMemoryCache(True)
join_object.setJoinLayer(ez_layer)
dkm_layer.addJoin(join_object)

# Save the joined layer as a new Shapefile
joined_file = os.path.join(working_directory, 'DKM_Joined.shp')
QgsProject.instance().writeEntry('LayerPaths', dkm_layer.id(), joined_file)

# Refresh the project to display the joined layer
QgsProject.instance().reload()

print('Attribute join completed successfully.')

Make sure you have QGIS installed and the qgis.core module accessible in your Python
environment to run this script. Update the working_directory variable with the desired path
where your DKM.shp and EZ.csv files are located. The script loads the Shapefile and CSV as
vector layers, performs the attribute join using the specified field, and saves the joined
layer as a new Shapefile named "DKM_Joined.shp" in the working directory.

After running the script, you can open the resulting Shapefile in QGIS to verify the
attribute join.
```

- **Notes:**

The workflow works perfectly. The script solution however does not run. It uses the "reload" function to update the map, which is discontinued in QGIS. This information is already part of ChatGPT's training data, since it does point this out when prompted with the interpreter error and does provide the correct solution in response. However, only the first solution is scored.

- **Complexity score:**

Complexity score

Section	Indicator	Measure	Norm-alizing	Weight	Score	Weighted Score
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.7**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0
	Select by location	0...not used 1...used	1.00	2	0	0.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	1	2.5
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	0	0.0
	Sum			23		**2.5**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**0.0**
Total complexity score				51		**3.2**

- **Solution quality score:**

Workflow solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	2	5.0
	Sum			10		**10.0**
Total workflow solution quality score				10		**10.0**
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**A**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	1	1.7
	Sum			10		**6.7**
Total code solution quality score				10		6.7
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**C**

6.4. Experiment: Join via composite key

- **Question structure and aim:**

The task is to perform another join. This time no primary keys will be provided. Instead, composite keys need to be created. The field names for each table are described. The running text points out the fields which should be used to create the key.

- **Chatlog:**

```
User
I have the following data in QGIS:

- DKM.shp:
contains polygons
contains the field KG
contains the field EZ

- Eigentümer.csv
contains the field KG
contains the field EZ
contains the field Eig_Name

Join the data over a composite key consisting of the fields KG and EZ

ChatGPT
To join the data in QGIS using a composite key consisting of the fields KG and EZ, you can
follow these steps:
1. Load the DKM.shp file into QGIS by dragging and dropping it into the Layers panel or by
using the "Add Vector Layer" option from the Layer menu.

2. Load the Eigentümer.csv file into QGIS by selecting "Add Delimited Text Layer" from the
Layer menu. In the dialog box, browse and select the CSV file, choose the appropriate
delimiter, and specify the field types. Make sure to assign the correct coordinate
reference system if prompted.

3. Once both the shapefile and the CSV file are loaded, you should see them listed in the
Layers panel.

4. Right-click on the DKM layer and select "Properties" from the context menu.

5. In the Layer Properties window, go to the "Joins" tab.

6. Click the "+" button to add a new join.

7. In the Join Properties dialog, select the Eigentümer.csv layer from the "Join layer"
dropdown menu.

8. Specify the join field from the DKM layer by selecting "KG" from the "Join field"
dropdown menu.

9. Specify the target field from the Eigentümer.csv layer by selecting "KG" from the
"Target field" dropdown menu.

10. Repeat the previous two steps for the "EZ" field, selecting "EZ" as both the join field
and target field.

11. Choose the join type based on your requirements. For example, "Keep all records" will
preserve all features from the DKM layer, even if there is no match in the Eigentümer.csv
file.

12. Click OK to apply the join.

After completing these steps, the DKM layer will be updated with the attributes from the
Eigentümer.csv file based on the composite key (KG and EZ). You can access the joined
attributes by opening the attribute table of the DKM layer. The joined fields from
```

Eigentümer.csv, such as Eig_Name, will be available in the attribute table of the DKM layer.

User:
Provide a python script to achieve this.
Use D:\GIS\Exp4 as the working directory

ChatGPT
Certainly! Here's a Python script that uses the QGIS Python API to join the data in QGIS using a composite key:

```python
import os
from qgis.core import QgsVectorLayer, QgsVectorFileWriter, QgsProject

# Set the working directory
working_dir = r"D:\GIS\Exp4"
os.chdir(working_dir)

# Load the DKM shapefile
dkm_shp = "DKM.shp"
dkm_layer = QgsVectorLayer(dkm_shp, "DKM", "ogr")

# Load the Eigentümer CSV file
eig_csv = "Eigentümer.csv"
eig_layer = QgsVectorLayer(f"{eig_csv}?delimiter=,", "Eigentümer", "delimitedtext")

# Join the layers using composite key (KG and EZ)
join_field = "KG,EZ"
target_field = "KG,EZ"
join_type = QgsVectorLayer.JoinFieldMode.JoinOneToOne
eig_layer.join(dkm_layer, QgsVectorLayer.JoinFieldOptions(join_field, target_field,
join_type))

# Save the joined layer
joined_shp = "DKM_Joined.shp"
QgsVectorFileWriter.writeAsVectorFormat(dkm_layer, joined_shp, "UTF-8", dkm_layer.crs(),
"ESRI Shapefile")

# Add the joined layer to the project
project = QgsProject.instance()
project.addMapLayer(dkm_layer)

# Refresh the map canvas
iface.mapCanvas().refreshAllLayers()
```

Make sure to replace `"DKM.shp"` and `"Eigentümer.csv"` with the actual file names and provide the correct path for the working directory (`r"D:\GIS\Exp4"`). This script assumes that you have the necessary QGIS Python packages installed and are running it within a QGIS Python environment.

The script loads the DKM shapefile and Eigentümer CSV file as vector layers, performs the join using the composite key, and saves the joined layer as a new shapefile called `"DKM_Joined.shp"`. Finally, it adds the joined layer to the QGIS project and refreshes the map canvas.

Note: When running this script, ensure that QGIS is properly installed and that the necessary Python packages are available in your Python environment.

- **Notes:**

The Workflow-Solution is not feasible. The Python Script doesn’t run. Both solutions look like they would make sense, but are in fact absolutely non-sensical

- **Complexity score:**

Complexity score							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.7**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0
	Select by location	0...not used 1...used	1.00	2	0	0.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	2	5.0
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	0	0.0
	Sum			23		**5.0**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**0.0**
Total complexity score				51		**5.7**

- **Solution quality score:**

Workflow solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	0	0.0
	Sum			10		**5.0**
Total workflow solution quality score				10		**5.0**
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	0	0.0
	Sum			10		**5.0**
Total code solution quality score				10		5.0
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**

6.5. Experiment: Join after table pre-processing

- **Question structure and aim:**

Before performing a simple join, the table needs to be pre-processed with a simple script. I am interested in ChatGPT's ability in table processing, but I am also eager to see how it handles the join in the second step, if it fails to do the pre-processing in the first step.

- **Chatlog:**

```
User
I have the following data in QGIS:

- DKM.shp:
contains polygons
contains the field EZ

- Eigentümer.csv
contains the field EZ
contains the field Eig_Name

In the Eigentümer.csv, merge all features that have the same EZ. Concatenate Eig_Name from
all merged features and add ";" as a delimiter.
Then join Eigentümer.csv to DKM.shp

ChatGPT
To merge the features in the Eigentümer.csv file based on the same EZ field and concatenate
the Eig_Name field with a ";" delimiter, you can follow these steps:

1. Open the Eigentümer.csv file in a spreadsheet application like Microsoft Excel or Google
Sheets.
2. Sort the data based on the EZ field to group all features with the same EZ together.
3. Add a new column, let's call it "Concatenated_Eig_Name," to the right of the Eig_Name
column.

4. In the first cell of the "Concatenated_Eig_Name" column (e.g., B2), enter the following
formula:
   ```
   =IF(A2=A1,B1&";"&C2,C2)
   ```
   This formula checks if the EZ value in the current row (A2) is the same as the previous
row (A1). If it is, it concatenates the Eig_Name from the previous row (B1), the delimiter
";" and the current Eig_Name (C2). Otherwise, it takes the current Eig_Name only.

5. Drag the formula down to apply it to all rows.

6. Save the modified Eigentümer.csv file.

Now, in QGIS, you can join the Eigentümer.csv file to the DKM.shp file using the following
steps:

1. Open QGIS and add the DKM.shp file to the project.

2. Right-click on the DKM layer in the Layers panel and select "Properties."

3. In the "Properties" window, go to the "Joins" tab.

4. Click the "+" button to add a new join.

5. In the "Join Layer" dropdown, select the Eigentümer.csv file.

6. Choose the "EZ" field as the join field in both the DKM layer and the Eigentümer.csv
file.

7. Select the "Concatenated_Eig_Name" field as the join field in the Eigentümer.csv file.

8. Choose the join type based on your requirements (e.g., "Keep all records").
```

9. Click "OK" to apply the join.

Now, the Eigentümer.csv attributes, including the concatenated Eig_Name field, should be joined to the DKM.shp layer in QGIS based on the EZ field.

User
Provide a VBA-Script for Excel and a Python script for QGIS that achieves this.

ChatGPT
Certainly! Here's an example of a VBA script for Excel and a Python script for QGIS that can accomplish the task you described.

VBA Script for Excel:

```vba
Sub MergeAndConcatenate()
    Dim ws As Worksheet
    Dim lastRow As Long
    Dim i As Long
    Dim currentEZ As String
    Dim concatenatedEigName As String

    ' Set the worksheet containing Eigentümer.csv data
    Set ws = ThisWorkbook.Sheets("Eigentümer")

    ' Sort the data based on EZ field
    ws.Sort.SortFields.Clear
    ws.Sort.SortFields.Add Key:=Range("A2:A" & ws.Cells(ws.Rows.Count, 1).End(xlUp).Row), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ws.Sort
        .SetRange Range("A1:B" & ws.Cells(ws.Rows.Count, 1).End(xlUp).Row)
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With

    lastRow = ws.Cells(ws.Rows.Count, 1).End(xlUp).Row

    ' Concatenate Eig_Name for same EZ values
    currentEZ = ws.Cells(2, 1).Value
    concatenatedEigName = ws.Cells(2, 2).Value

    For i = 3 To lastRow
        If ws.Cells(i, 1).Value = currentEZ Then
            concatenatedEigName = concatenatedEigName & ";" & ws.Cells(i, 2).Value
        Else
            ws.Cells(i - 1, 3).Value = concatenatedEigName
            currentEZ = ws.Cells(i, 1).Value
            concatenatedEigName = ws.Cells(i, 2).Value
        End If
    Next i

    ' Add the concatenated Eig_Name for the last group
    ws.Cells(lastRow, 3).Value = concatenatedEigName

    ' Save the modified Eigentümer.csv file
    ThisWorkbook.Save

    MsgBox "Merge and concatenation completed successfully!"
End Sub
```

This VBA script assumes that the Eigentümer.csv file is in a worksheet named "Eigentümer" and that the EZ field is in column A and the Eig_Name field is in column B. The script sorts the data based on the EZ field and concatenates the Eig_Name for the same EZ values in the third column.

Python script for QGIS:

````
```python
import csv
from collections import defaultdict

# Specify the paths to the Eigentümer.csv and DKM.shp files
csv_file = '/path/to/Eigentümer.csv'
shp_file = '/path/to/DKM.shp'

# Create a dictionary to store Eig_Name values for each EZ key
eig_names = defaultdict(list)

# Read Eigentümer.csv and store Eig_Name values for each EZ key
with open(csv_file, 'r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        eig_names[row['EZ']].append(row['Eig_Name'])

# Open DKM.shp layer
layer = QgsVectorLayer(shp_file, 'DKM', 'ogr')

# Check if the layer was loaded successfully
if not layer.isValid():
    print("Error loading shapefile:", layer.errorMessage())
else:
    # Start editing the layer
    layer.startEditing()

    # Add a new field to store the concatenated Eig_Name values
    layer.dataProvider().add

Attributes([QgsField('Concatenated_Eig_Name', QVariant.String)])
    layer.updateFields()

    # Iterate over features and update the Concatenated_Eig_Name field
    for feature in layer.getFeatures():
        ez_value = feature['EZ']
        eig_name_values = eig_names[ez_value]
        concatenated_eig_name = ';'.join(eig_name_values)
        layer.changeAttributeValue(feature.id(),
layer.fields().indexFromName('Concatenated_Eig_Name'), concatenated_eig_name)

    # Stop editing and save changes
    layer.commitChanges()
    layer.updateFields()

    print("Join and concatenation completed successfully!")
```

Make sure to replace `'/path/to/Eigentümer.csv'` and `'/path/to/DKM.shp'` with the actual
file paths for the Eigentümer.csv and DKM.shp files, respectively. The script reads the
Eigentümer.csv file, stores the Eig_Name values for each EZ key in a dictionary, and then
joins and updates the Concatenated_Eig_Name field in the DKM.shp layer accordingly.

Note: Remember to run the Python script within the QGIS environment or using a compatible
Python IDE with QGIS libraries properly installed.
````

- **Notes:**

Preprocessing the CSV in Excel is a good solution in principle. The Formula provided goes in the right direction as well, but the cell references in the Formula are completely scrambled. Even if the cell references would be correct, it would leave duplicate data. While the Formula is not feasible at all, surprisingly the code solution almost works. It only misses to delete the empty rows.
For the join it assumes its preprocessing is correct. It successfully considers what it did in the preprocessing, since it chooses the newly created field for the join.
The workflow on the join is correct in all regards. The code solution still uses the discontinued function and fails to run.

- **Complexity score:**

Complexity score						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.7**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0
	Select by location	0...not used 1...used	1.00	2	0	0.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	1	2.5
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	2	3.0
	Sum			23		**5.5**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**0.0**
Total complexity score				51		**6.2**

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0...not understandable 1...intention is conveyed 2...clearly understandable 3...easy to follow guide	0.33	2	3	2.0
	Complete	0...problem wasn't solved 1...solved to some extend 2...minor details missing 3...solved to full extend	0.33	3	2	2.0
	Correct	0...incorrect 1...overall correct with minor errors 2...correct in all aspects	0.50	5	1	2.5
	Sum			10		**6.5**
Total workflow solution quality score				10		**6.5**
Total workflow solution quality rating		**8 to 10 ... A** **7 to 8 ... B** **6 to 7 ... C** **5 to 6 ... D** **<= 5 ... F**				**C**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0...no code structure or proper naming 1...code is structured and readable 2...proper naming of variables and functions 3...code properly documented	0.33	2	3	2.0
	Complete	0...problem wasn't solved 1...solved to some extend 2...minor details missing 3...solved to full extend	0.33	3	2	2.0
	Correct	0...incorrect 1...major errors 2...overall correct with minor errors 3...correct in all aspects	0.33	5	1	1.7
	Sum			10		**5.7**
Total code solution quality score				10		5.7
Total code solution quality rating		**8 to 10 ... A** **7 to 8 ... B** **6 to 7 ... C** **5 to 6 ... D** **<= 5 ... F**				**D**

6.6. Experiment: Vector-layer styling

- **Question structure and aim:**

A number of layers are given with no further processing required. The goal is to create a serviceable symbology and labelling according to the specifications. Note that the solution quality does not consider how "pretty" the map is, only if the specifications are met.

- **Chatlog:**

```
I have the following layers in QGIS:

- roads
contains polylines
has the field "type" thats either "Major Highway", "Road" and "Unknown"

- places
contains points
has the field "NAME"
has the field "SCALERANK" with values from 0 to 9

- urban_areas
contains polygons
has the field "area_sqkm"

- state_boundary
contains polygons
has the field "ST_NM"

The layers should be styled as follows:
- For the roads make Major Highways thicker than the other types.
- places should show as dots that are bigger the smaller the SCALERANK
- urban areas should be styled halftransparent and labeled with area_sqkm
- state_boundary should only show as a thick borderline and ST_NM as a label along the
inside of the  border

Provide instructions on how to achieve this.

ChatGPT
To achieve the desired styling in QGIS, follow these instructions:

1. Open QGIS and add the layers (roads, places, urban_areas, state_boundary) to your
project.

2. Styling the "roads" layer:
   - Right-click on the "roads" layer in the Layers Panel and select "Properties" from the
context menu.
   - In the Layer Properties dialog, go to the "Symbology" tab.
   - Select "Categorized" from the top drop-down menu.
   - Choose the "type" field as the "Value" to categorize the features.
   - Click on the "Classify" button to generate categories based on the "type" field.
   - Select each category (Major Highway, Road, Unknown) one by one and set the desired
line thickness for each category. Make the Major Highway thicker than the others.
   - Click "OK" to apply the styling.

3. Styling the "places" layer:
   - Right-click on the "places" layer in the Layers Panel and select "Properties" from the
context menu.
   - In the Layer Properties dialog, go to the "Symbology" tab.
   - Select "Graduated" from the top drop-down menu.
   - Choose the "SCALERANK" field as the "Value" for graduation.
   - Select the desired symbol for the points.
   - Adjust the size of the symbol using the "Size" parameter. You can use an expression to
make it smaller for higher scalerank values, e.g., `10 - "SCALERANK"` or `10 /
"SCALERANK"`.
   - Click "OK" to apply the styling.

4. Styling the "urban_areas" layer:
```

- Right-click on the "urban_areas" layer in the Layers Panel and select "Properties" from the context menu.
   - In the Layer Properties dialog, go to the "Symbology" tab.
   - Choose the desired fill symbol for the polygons.
   - Adjust the transparency of the fill symbol using the "Opacity" parameter.
   - Enable labeling by checking the "Label this layer with" box.
   - Choose the "area_sqkm" field for labeling.
   - Adjust the label placement and style as desired.
   - Click "OK" to apply the styling.

5. Styling the "state_boundary" layer:
   - Right-click on the "state_boundary" layer in the Layers Panel and select "Properties" from the context menu.
   - In the Layer Properties dialog, go to the "Symbology" tab.
   - Select "Single symbol" from the top drop-down menu.
   - Choose a line symbol for the border.
   - Set the desired line thickness for the border.
   - Enable labeling by checking the "Label this layer with" box.
   - Choose the "ST_NM" field for labeling.
   - Adjust the label placement and style to be placed along the inside of the border.
   - Click "OK" to apply the styling.

Now, your layers should be styled according to the provided instructions in QGIS. Adjust the styling parameters as needed to achieve the desired visual representation.

User
Provide a pyton script that achieves this.
Use the layers that already exist in the project.

ChatGPT
Certainly! Here's a Python script that utilizes the QGIS Python API to achieve the desired styling for the existing layers in your QGIS project:

```python
# Import the required QGIS modules
from qgis.core import QgsProject, QgsVectorLayerSimpleLabeling

# Access the current project
project = QgsProject.instance()

# Get the reference to the layers
roads_layer = project.mapLayersByName("roads")[0]
places_layer = project.mapLayersByName("places")[0]
urban_areas_layer = project.mapLayersByName("urban_areas")[0]
state_boundary_layer = project.mapLayersByName("state_boundary")[0]

# Styling the roads layer
renderer = roads_layer.renderer()
symbol = renderer.symbol()
symbol.setWidth(0.6)  # Set the default line width for roads
expression = "type"
value_list = ["Major Highway", "Road", "Unknown"]
for value in value_list:
    symbol_layer = symbol.symbolLayer(0).clone()
    if value == "Major Highway":
        symbol_layer.setWidth(1.2)  # Increase line width for Major Highways
    renderer.addCategory(value, symbol_layer, value)
roads_layer.triggerRepaint()

# Styling the places layer
places_layer.setRenderer(QgsGraduatedSymbolRenderer("SCALERANK"))
symbol = QgsSymbol.defaultSymbol(places_layer.geometryType())
symbol.setSize(1.5)  # Set the default symbol size for places
renderer = places_layer.renderer()
renderer.updateClasses(places_layer, QgsGraduatedSymbolRenderer.ModeEqualInterval, 4)
places_layer.triggerRepaint()

# Styling the urban_areas layer
symbol = QgsSymbol.defaultSymbol(urban_areas_layer.geometryType())
symbol.setColor(QColor(127, 127, 127, 127))  # Set the default semi-transparent fill color
renderer = urban_areas_layer.renderer()
renderer.setOpacity(0.5)  # Set the default fill opacity for urban areas
```

```
labeling = QgsVectorLayerSimpleLabeling()
labeling.setEnabled(True)
labeling.setField("area_sqkm")
labeling.setFormattedNumbers(False)
labeling.setLabelPlacement(QgsPalLayerSettings.OverPoint)
urban_areas_layer.setLabeling(labeling)
urban_areas_layer.setLabelsEnabled(True)
urban_areas_layer.triggerRepaint()

# Styling the state_boundary layer
symbol = QgsSymbol.defaultSymbol(state_boundary_layer.geometryType())
symbol.setColor(QColor(0, 0, 0, 0))  # Set the default transparent fill color
symbol.setWidth(0.6)  # Set the default line width for state boundary
renderer = state_boundary_layer.renderer()
renderer.setSymbol(symbol)
labeling = QgsVectorLayerSimpleLabeling()
labeling.setEnabled(True)
labeling.setField("ST_NM")
labeling.setFormattedNumbers(False)
labeling.setLabelPlacement(QgsPalLayerSettings.Line)
state_boundary_layer.setLabeling(labeling)
state_boundary_layer.setLabelsEnabled(True)
state_boundary_layer.triggerRepaint()
```

```
Make sure to run this script within the Python Console in QGIS. It will style the existing
layers according to the provided requirements.
```

- **Notes:**

The styling instructions were mostly good. The instruction on how to set the label to the border were lacking in detail. The solution to use data defined symbology for the graduated point symbols are overkill for such a simple task. How ChatGPT handles graduated symbol definitely needs to be looked further into, in the next experiments. The code-solution reads promising, but fails to run. Prompting the interpreter errors back to ChatGPT produces revised code, which contains a different set of errors.

- **Complexity score:**

Complexity score						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0…1 file 1…<=2 files 2…<=5 files 3…>5 files	0.33	2	2	1.3
	Dataconversion	0…no datatypeconversion 1…simple shape type conversion 2…advanced datatype conversion 3…non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**1.3**
Vectorprocessing	Number of different vectorprocessing tools used	0…no tools used 1…<=2 2…<=5 3…>5	0.33	5	0	0.0
	Select from table	0…not used 1…simple attribute query 2…multipart query	0.50	7	0	0.0
	Select by location	0…not used 1…used	1.00	2	0	0.0
	Join from table	0…not used 1…join by existing key 2…composit key creation and join	0.50	5	0	0.0
	Join by location	0…not used 1…used	1.00	1	0	0.0
	Field calculation	0…not used 1…simple Field calculation 2…multipart expression	0.50	3	0	0.0
	Sum			23		**0.0**
Rasterprocessing	Number of different rasterprocessing tools used	0…no tools used 1…<=2 2…<=5 3…>5	0.33	5	0	0.0
	Raster calculation	0…not used 1…simple attribute query 2…multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0…not used 1…categorized 2…graduated 3…rules based	0.33	2	2	1.3
	Labeling	0…not used 1…simple label 2…multipart expression	0.50	4	1	2.0
	Layouting	0…not used 1…used	1.00	1	0	0.0
	Sum			7		**3.3**
Total complexity score				51		**4.7**

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	2	2.0
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	2	5.0
	Sum			10		**9.0**
Total workflow solution quality score				10		**9.0**
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**A**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	0	0.0
	Sum			10		**5.0**
Total code solution quality score				10		5.0
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**

6.7. Experiment: Raster Layer Styling

- **Question structure and aim:**

A raster layers is given with no further processing required. The goal is to create a graduated symbology. This time the colour-scheme that should be used is specified and is therefor part of the evaluation.

- **Chatlog:**

```
User
I have the following data in QGIS:

- denali_dgm.tif
the raster contains values from aprox. -2000 to +3000 representing heights above sea level.

Style the layer as follows:
- dont show positive values
- choose graduated colors to represent water depth of the remaining values

Provide instructions on how to do this

ChatGPT
To style the denali_dgm.tif layer in QGIS as described, you can follow these instructions:

1. Open QGIS and load the denali_dgm.tif raster layer into your project.
   - Go to the "Layer" menu.
   - Click on "Add Layer" and select "Add Raster Layer."
   - Browse to the location of the denali_dgm.tif file and click "Open" to load it into
QGIS.

2. Open the Layer Styling panel for the denali_dgm.tif layer.
   - Right-click on the denali_dgm.tif layer in the Layers panel.
   - Choose "Properties" from the context menu.

3. Set the symbology to display only negative values.
   - In the Layer Styling panel, select the "Symbology" tab.
   - Change the "Render type" to "Singleband pseudocolor."
   - Next to the "Color ramp" dropdown, click on the button with three dots ("...") to open
the color ramp selection dialog.

4. Create a custom color ramp to represent water depth.
   - In the "Color ramp" dialog, select the "New color ramp" option (the button with a plus
sign).
   - Choose "Graduated" as the ramp type.
   - Set the "Mode" to "Equal interval."
   - Adjust the number of classes and the color scheme according to your preference.
     Note: Since you want to represent water depth, you might consider using blue or shades
of blue as the color scheme.

5. Exclude positive values from the color ramp.
   - In the "Color ramp" dialog, click on the first class range (e.g., -2000 to -1500).
   - In the "Value" field, replace the minimum value with 0 (zero).
     Note: This will exclude positive values from the color ramp, as you don't want to
display them.
   - Repeat this step for each class range until you have excluded all positive values.

6. Apply the color ramp to the raster layer.
   - After setting up the custom color ramp, click "OK" to close the "Color ramp" dialog.
   - In the Layer Styling panel, you should now see the custom color ramp you created.
   - Click "OK" to apply the symbology settings and close the Layer Properties dialog.

The denali_dgm.tif raster layer should now be styled to show only negative values and
represent water depth using a graduated color ramp. Positive values will not be displayed,
and the remaining values will be represented with the chosen colors based on the water
depth intervals you defined.

User
```

Provide a pyton script that achieves this.
Use the layer denali_dgm that already exist in the project.

ChatGPT
Certainly! Here's a Python script that achieves the styling you described using the denali_dgm.tif raster layer in QGIS:

```python
# Import the necessary QGIS modules
from qgis.core import QgsProject, QgsRasterRenderer, QgsColorRampShader, QgsRasterShader,
QgsSingleBandPseudoColorRenderer

# Get the denali_dgm layer from the current project
layer = QgsProject.instance().mapLayersByName('denali_dgm')[0]

# Set up the raster renderer
renderer = QgsSingleBandPseudoColorRenderer(layer.dataProvider(), 1)

# Create a color ramp shader for the water depth
color_ramp = QgsColorRampShader()
color_ramp.setColorRampType(QgsColorRampShader.Interpolated)
color_ramp.setClassificationMode(QgsColorRampShader.EqualInterval)
color_ramp.setSourceColorRamp(QgsColorRampShader.blue())

# Exclude positive values
ranges = color_ramp.ranges()
ranges[0].minimumValue = 0

# Apply the color ramp shader to the raster renderer
shader = QgsRasterShader()
shader.setRasterShaderFunction(color_ramp)
renderer.setShader(shader)

# Set the renderer to the layer
layer.setRenderer(renderer)

# Refresh the layer to update the styling
layer.triggerRepaint()

# Print a message to confirm that the styling has been applied
print('Styling applied to the denali_dgm layer.')
```

Make sure the denali_dgm layer is already loaded in your QGIS project before running this script. Once executed, the script will apply the desired styling to the denali_dgm layer, excluding positive values and using a graduated color ramp to represent water depth.

Note: This script assumes that you have the necessary QGIS Python environment set up and can be executed within QGIS or in an external Python environment with the required QGIS dependencies installed.

- **Notes:**

The instruction mostly works, although it does not give the style it claims it does. Setting the minimum value to zero does not get rid of the class. QGIS stretches the first and last class to the upper and lower limit of the raster. Instead, a class should have been created for the values that should not be shown and set to no symbol. The given instruction, while claiming to, do not solve the problem given. Code solution did not run.

- **Complexity score:**

Complexity score						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	0	0.0
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.0**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0
	Select by location	0...not used 1...used	1.00	2	0	0.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	0	0.0
	Sum			23		**0.0**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	2	1.3
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**1.3**
Total complexity score				51		**1.3**

- **Solution quality score:**

Workflow solution quality

Section	Indicator	Measure	Norm-alizing	Weight	Score	Weighted Score
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	1	2.5
	Sum			10		**7.5**
Total workflow solution quality score				10		**7.5**
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**B**

Code solution quality

Section	Indicator	Measure	Norm-alizing	Weight	Score	Weighted Score
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	0	0.0
	Sum			10		**5.0**
Total code solution quality score				10		5.0
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**

6.8. Experiment: Advanced rules-based layer styling

- **Question structure and aim:**

A more complicated styling needs to be created. This is only solvable by creating a rules-based symbology with the proper SQL queries.

- **Chatlog:**

User
I have the following layer in QGIS:
- Landuse
contains polygons
contains the field Area that contains the area in m²
contains the field Type that specifies the type of landuse as text

Style the layer Landuse as follows:
- features with an area less than 20000m² should be grey
- features where the type starts with "ö" should be colored green
- features of type "BL" and "LN" should be red
- any other features should be yellow

Provide instructions on how to do this

ChatGPT
To style the "Landuse" layer in QGIS according to your requirements, you can follow these steps:

1. Open QGIS and load the "Landuse" layer containing the polygons.

2. Right-click on the "Landuse" layer in the Layers panel and select "Properties" from the context menu. Alternatively, you can double-click on the layer to open the Layer Properties dialog.

3. In the Layer Properties dialog, go to the "Symbology" tab.

4. By default, you will see a single symbol renderer. Click on the drop-down menu next to the symbol and select "Categorized" to create categorized styling based on your criteria.

5. In the "Value" column, click on the small button with three dots to open the Expression dialog.

6. To define the first rule, where features have an area less than 20000m² and should be styled as grey, enter the following expression:
```
"Area" < 20000
```
Then, click OK.

7. In the "Symbol" column for this rule, click on the color swatch and choose grey as the fill color.

8. To define the second rule, where the type starts with "ö" and should be colored green, click on the green plus button below the first rule.

9. In the "Value" column of the new rule, enter the following expression:
```
"Type" LIKE 'ö%'
```
Then, click OK.

10. In the "Symbol" column for this rule, click on the color swatch and choose green as the fill color.

11. To define the third rule, where the type is "BL" or "LN" and should be colored red, click on the green plus button below the second rule.

12. In the "Value" column of the new rule, enter the following expression:
```
```

```
"Type" IN ('BL', 'LN')
```
Then, click OK.

13. In the "Symbol" column for this rule, click on the color swatch and choose red as the fill color.

14. Finally, to define the fourth rule, where any other features should be colored yellow, click on the green plus button below the third rule.

15. Leave the "Value" column empty for this rule.

16. In the "Symbol" column for this rule, click on the color swatch and choose yellow as the fill color.

17. Click OK to apply the symbology and close the Layer Properties dialog.

Now, the "Landuse" layer should be styled according to your specifications, with features that have an area less than 20000m² in grey, features with a type starting with "ö" in green, features of type "BL" and "LN" in red, and any other features in yellow.

User
Provide a pyton script that achieves this

ChatGPT
Certainly! Here's a Python script that achieves the styling requirements for the "Landuse" layer in QGIS:

```python
# Import the necessary QGIS modules
from qgis.core import QgsProject, QgsVectorLayer, QgsCategorizedSymbolRenderer, QgsSymbol,
QgsRendererCategory

# Define the layer name and field names
layer_name = 'Landuse'
area_field = 'Area'
type_field = 'Type'

# Get the layer by its name
layer = QgsProject.instance().mapLayersByName(layer_name)[0]

# Create a categorized symbol renderer
renderer = QgsCategorizedSymbolRenderer()

# Define the symbol for features with an area less than 20000m² (grey)
grey_symbol = QgsSymbol.defaultSymbol(layer.geometryType())
grey_symbol.setColor(QColor('grey'))
renderer_category = QgsRendererCategory('Less than 20000m²', grey_symbol, f'"{area_field}"
< 20000')
renderer.addCategory(renderer_category)

# Define the symbol for features where the type starts with "ö" (green)
green_symbol = QgsSymbol.defaultSymbol(layer.geometryType())
green_symbol.setColor(QColor('green'))
renderer_category = QgsRendererCategory('Type starts with "ö"', green_symbol,
f'"{type_field}".startsWith(\'ö\')')
renderer.addCategory(renderer_category)

# Define the symbol for features of type "BL" and "LN" (red)
red_symbol = QgsSymbol.defaultSymbol(layer.geometryType())
red_symbol.setColor(QColor('red'))
renderer_category = QgsRendererCategory('Type is BL or LN', red_symbol, f'"{type_field}" IN
(\'BL\', \'LN\')')
renderer.addCategory(renderer_category)

# Define the symbol for any other features (yellow)
yellow_symbol = QgsSymbol.defaultSymbol(layer.geometryType())
yellow_symbol.setColor(QColor('yellow'))
renderer_category = QgsRendererCategory('Other', yellow_symbol, 'ELSE')
renderer.addCategory(renderer_category)

# Assign the renderer to the layer
```

~~~
layer.setRenderer(renderer)

# Refresh the layer to update the styling
layer.triggerRepaint()
```

You can run this script within the QGIS Python Console or as a standalone Python script
within the QGIS Python environment. Make sure you have the "Landuse" layer loaded in QGIS
before running the script. The script will apply the categorized styling to the layer based
on the specified criteria.
~~~

- **Notes:**

Despite this symbology being harder to achieve then the ones in Experiment 6 and 7. This is the closest solution yet. The workflow has one error where it says to set the symbology type to categorized instead of rules based. The rest of the instructions are correct. The most surprising outcome of this experiment is, that this code solution actually runs, despite the very similar and easer solution fail to parse. It still carries the same error from the workflow where it sets the wrong symbology type, but this is very close to a correct solution.

- **Complexity score:**

Complexity score						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	0	0.0
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0
	Sum			10		**0.0**
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0
	Select by location	0...not used 1...used	1.00	2	0	0.0
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0
	Join by location	0...not used 1...used	1.00	1	0	0.0
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	0	0.0
	Sum			23		**0.0**
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0
	Sum			11		**0.0**
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	3	2.0
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0
	Layouting	0...not used 1...used	1.00	1	0	0.0
	Sum			7		**2.0**
Total complexity score				51		**2.0**

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	1	2.5
	Sum			10		**7.5**
Total workflow solution quality score				10		**7.5**
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**B**

Code solution quality						
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	1	1.7
	Sum			10		**6.7**
Total code solution quality score				10		6.7
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**C**

6.9. Experiment: Area Coverage 1

- **Question structure and aim:**

This task has several possible solutions. I will formulate the same question differently in the next experiment to see if ChatGPT prefers one solution to another based on the words used in the question.

- **Chatlog:**

```
User
I have the following layers in QGIS:

- DKM
contains polygons
contains the field GST

- roadplanning
contains polygons
contains the field Type with possible values of Road, Parking, Plants

For each polygon in DKM i need the total area of the polygon and the area that is used by
Road, Parking and Plants
Provide instructions on how to do this

ChatGPT
To calculate the total area of each polygon in the "DKM" layer and the area that is used by
"Road," "Parking," and "Plants" in the "roadplanning" layer within each polygon of the
"DKM" layer, you can use the following steps in QGIS:

1. Load both layers ("DKM" and "roadplanning") into your QGIS project.

2. Make sure that both layers are in the same Coordinate Reference System (CRS) to ensure
accurate area calculations.

3. Open the Field Calculator to create new fields for the "DKM" layer where you'll
calculate the total area and areas used by different types from the "roadplanning" layer.

Here's how you can calculate these fields step by step:

   a. Right-click on the "DKM" layer in the Layers panel and choose "Open Attribute Table."

   b. Click on the "Open Field Calculator" button (the abacus icon).

   c. Create a new field for the total area:
      - Output field name: Total_Area (or any name you prefer)
      - Output field type: Decimal number (real)
      - Output field length: Adjust this based on your precision needs (e.g., 10)
      - Output field precision: Adjust this based on your precision needs (e.g., 2)

   d. In the Expression box, type the following expression to calculate the total area for
each polygon in the "DKM" layer:
      ```
      $area
      ```
      This expression will calculate the area of each "DKM" polygon.

   e. Click "OK" to create the "Total_Area" field.

   f. Now, create new fields for the areas used by "Road," "Parking," and "Plants":
      - Output field name: Road_Area
      - Output field type: Decimal number (real)
      - Output field length: Adjust this based on your precision needs (e.g., 10)
      - Output field precision: Adjust this based on your precision needs (e.g., 2)
      - Repeat these steps for "Parking_Area" and "Plants_Area."

   g. In the Expression box for the "Road_Area" field, type the following expression to
calculate the area used by "Road" within each "DKM" polygon:
```

```
aggregate('roadplanning', 'sum', "Type", 'Road')
```

h. In the Expression box for the "Parking_Area" field, type the following expression to calculate the area used by "Parking" within each "DKM" polygon:
```
aggregate('roadplanning', 'sum', "Type", 'Parking')
```

i. In the Expression box for the "Plants_Area" field, type the following expression to calculate the area used by "Plants" within each "DKM" polygon:
```
aggregate('roadplanning', 'sum', "Type", 'Plants')
```

j. Click "OK" to create the new fields.

4. After following these steps, your "DKM" attribute table will now contain the "Total_Area," "Road_Area," "Parking_Area," and "Plants_Area" fields, providing the desired area calculations.

These calculations use the aggregate function to sum the areas of different types from the "roadplanning" layer within each polygon of the "DKM" layer.

User
Provide a pyton script that achieves this.
with D:\GIS\Exp9 as the working directory.

ChatGPT
To achieve this using Python within the QGIS environment, you can create a script using the QGIS Python API. Make sure that your QGIS project is open and set the working directory to "D:\GIS\Exp9" using Python's `os` module. Here's a Python script to accomplish the task:

```python
import os
from qgis.core import QgsProject, QgsVectorLayer, QgsExpression, QgsExpressionContext, QgsExpressionContextUtils

# Set the working directory
working_dir = r'D:\GIS\Exp9'
os.chdir(working_dir)

# Load the layers
dkm_layer = QgsVectorLayer('DKM.shp', 'DKM', 'ogr')
roadplanning_layer = QgsVectorLayer('roadplanning.shp', 'roadplanning', 'ogr')

# Add the layers to the project
QgsProject.instance().addMapLayer(dkm_layer)
QgsProject.instance().addMapLayer(roadplanning_layer)

# Create field calculator expressions
expression_context = QgsExpressionContext()
expression_context.appendScope(QgsExpressionContextUtils.layerScope(dkm_layer))

# Calculate the total area
total_area_expression = QgsExpression('$area')
total_area_expression.prepare(dkm_layer.pendingFields())
total_area_field = QgsExpression.prepareField(total_area_expression, dkm_layer)
dkm_layer.startEditing()
for feature in dkm_layer.getFeatures():
    expression_context.setFeature(feature)
    total_area = total_area_expression.evaluate(expression_context)
    feature[total_area_field] = total_area
    dkm_layer.updateFeature(feature)
dkm_layer.commitChanges()

# Calculate the areas used by different types
types = ['Road', 'Parking', 'Plants']
for type in types:
    expression = f'aggregate("roadplanning", "sum", "Type", \'{type}\')'
    expression_context.setFields(roadplanning_layer.fields())
```

```
expression_context.appendScope(QgsExpressionContextUtils.layerScope(roadplanning_layer))
    type_area_expression = QgsExpression(expression)
    type_area_expression.prepare(roadplanning_layer.pendingFields())
    type_area_field = QgsExpression.prepareField(type_area_expression, dkm_layer)

    dkm_layer.startEditing()
    for feature in dkm_layer.getFeatures():
        expression_context.setFeature(feature)
        area = type_area_expression.evaluate(expression_context)
        feature[type_area_field] = area
        dkm_layer.updateFeature(feature)
    dkm_layer.commitChanges()

# Save changes
QgsProject.instance().write()

# Remove layers from the project (optional)
QgsProject.instance().removeMapLayer(dkm_layer)
QgsProject.instance().removeMapLayer(roadplanning_layer)
```

Save this script as a `.py` file and run it within the QGIS Python console. This script sets the working directory, loads the layers, and uses field calculator expressions to calculate the total area and areas used by different types. The results are added as new fields in the "DKM" layer. You can then save the project to apply the changes permanently.

Make sure to replace `'DKM.shp'` and `'roadplanning.shp'` with the correct file paths if your shapefiles have different names or are located in subdirectories within "D:\GIS\Exp9."

- **Notes:**

Using the aggregate function is not a working approach

- **Complexity score:**

Complexity score							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7	-
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0	-
	Sum			10		**0.7**	
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	2	3.3	-
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0	-
	Select by location	0...not used 1...used	1.00	2	0	0.0	-
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0	-
	Join by location	0...not used 1...used	1.00	1	0	0.0	-
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	1	1.5	-
	Sum			23		**4.8**	
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0	-
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0	-
	Sum			11		**0.0**	
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0	-
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0	-
	Layouting	0...not used 1...used	1.00	1	0	0.0	-
	Sum			7		**0.0**	
Total complexity score				51		**5.5**	

- **Solution quality score:**

Workflow solution quality								
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0	-
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	0	0.0	-
	Sum			10		**5.0**	
Total workflow solution quality score				10		**5.0**	
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**	

Code solution quality								
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0	-
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	0	0.0	-
	Sum			10		**5.0**	
Total code solution quality score				10		5.0	
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**	

6.10. Experiment: Area Coverage 2

- **Question structure and aim:**

Continues the series of Experiment 9. The goal is the same, but this time the prompt includes the word "intersect" to see if this entices ChatGPT to answer with the solution to this task that is based on an intersection of the data.

- **Chatlog:**

```
User
I have the following layers in QGIS:

- DKM
contains polygons
contains the field GST

- roadplanning
contains polygons
contains the field Type with possible values of Road, Parking, Plants

For each polygon in DKM i need the total area of the polygon and the areas of the
roadplanning polygons that intersects each feature of the DKM.
Provide instructions on how to do this

ChatGPT
To calculate the total area of each polygon in the "DKM" layer and the areas of the
"roadplanning" polygons that intersect each feature in the "DKM" layer in QGIS, you can use
the following steps:

1. Open your QGIS project and ensure that both the "DKM" and "roadplanning" layers are
loaded into your project.

2. First, calculate the total area of each polygon in the "DKM" layer:

   a. Right-click on the "DKM" layer in the Layers Panel.
   b. Select "Open Attribute Table" to open the attribute table for the "DKM" layer.
   c. Click on the "Open Field Calculator" icon (an abacus) in the attribute table toolbar.

3. In the Field Calculator:

   a. Create a new field to store the total area. Name it something like "Total_Area."
   b. Use the expression `$area` to calculate the area for each polygon feature in "DKM."
   c. Click "OK" to calculate and update the field with the total areas.

4. Now, you can calculate the areas of "roadplanning" polygons that intersect each feature
in the "DKM" layer:

   a. Go to the "Processing" menu and select "Toolbox" to open the Processing Toolbox.
   b. In the Processing Toolbox, search for the "Intersection" tool.

   c. Double-click the "Intersection" tool to open its dialog box.

   d. Set the "Input layer" to the "DKM" layer.
   e. Set the "Overlay layer" to the "roadplanning" layer.
   f. Specify an output file for the result, such as a temporary file or a new layer. Make
sure to select the option to "Create multipart features," which will ensure that the tool
calculates the total area for each intersection.

   g. Click "Run" to execute the Intersection tool.

5. Once the intersection is complete, you will have a new layer that contains the
intersected polygons from the "DKM" layer along with the corresponding attributes of the
"roadplanning" layer.

6. Open the attribute table of the new intersected layer.

7. Calculate the area of each intersected feature in this new layer:
```

a. Click on the "Open Field Calculator" icon in the attribute table toolbar.

b. Create a new field, for example, "Intersection_Area."

c. Use the expression `$area` to calculate the area for each intersected feature.

d. Click "OK" to calculate and update the field with the intersection areas.

Now, your "DKM" layer's attribute table should have two new fields: "Total_Area," which contains the total area of each "DKM" polygon, and "Intersection_Area," which contains the area of the "roadplanning" polygons that intersect each "DKM" feature.

User
Provide a pyton script that achieves this.
with D:\GIS\Exp10 as the working directory.

Certainly, you can achieve this using a Python script in QGIS. To run the script, make sure you've loaded the "DKM" and "roadplanning" layers in your QGIS project. Save the following Python script in a `.py` file, and then run it from the QGIS Python console, specifying your working directory.

Here's a Python script that accomplishes the task:

```python
import os
from qgis.core import QgsVectorLayer, QgsProject, QgsExpression, QgsFeature, QgsField,
QgsVectorFileWriter, QgsGeometry, QgsWkbTypes
from qgis.PyQt.QtCore import QVariant

# Define your working directory
working_directory = r'D:\GIS\Exp10'

# Set the working directory
os.chdir(working_directory)

# Load the "DKM" and "roadplanning" layers
dkm_layer = QgsVectorLayer('DKM.shp', 'DKM', 'ogr')
roadplanning_layer = QgsVectorLayer('roadplanning.shp', 'roadplanning', 'ogr')

# Create a list to store the results
results = []

# Iterate through "DKM" features
for dkm_feature in dkm_layer.getFeatures():
    dkm_area = dkm_feature.geometry().area()
    dkm_id = dkm_feature.id()

    # Get the intersecting "roadplanning" features
    intersecting_features = []
    for roadplanning_feature in roadplanning_layer.getFeatures():
        if dkm_feature.geometry().intersects(roadplanning_feature.geometry()):
            intersection =
dkm_feature.geometry().intersection(roadplanning_feature.geometry())
            intersection_area = intersection.area()
            intersecting_features.append(intersection_area)

    # Calculate the total intersection area for the DKM feature
    total_intersection_area = sum(intersecting_features)

    # Add the results to the list
    results.append((dkm_id, dkm_area, total_intersection_area))

# Create a new memory layer to store the results
layer_name = 'DKM_Intersection_Areas'
crs = dkm_layer.crs()
new_layer = QgsVectorLayer('Polygon?crs={}'.format(crs.authid()), layer_name, 'memory')
provider = new_layer.dataProvider()
new_layer.startEditing()

# Define fields for the new layer
fields = [
```

```
    QgsField('DKM_ID', QVariant.Int),
    QgsField('DKM_Area', QVariant.Double),
    QgsField('Total_Intersection_Area', QVariant.Double)
]

provider.addAttributes(fields)
new_layer.updateFields()

# Add features to the new layer
for result in results:
    dkm_id, dkm_area, total_intersection_area = result
    new_feature = QgsFeature()
    new_feature.setGeometry(QgsGeometry.fromWkt(dkm_feature.geometry().asWkt()))
    new_feature.setAttributes([dkm_id, dkm_area, total_intersection_area])
    new_layer.addFeature(new_feature)

new_layer.commitChanges()

# Add the new layer to the project
QgsProject.instance().addMapLayer(new_layer)

# Save the new layer to a shapefile
output_path = os.path.join(working_directory, 'DKM_Intersection_Areas.shp')
QgsVectorFileWriter.writeAsVectorFormat(new_layer, output_path, "utf-8", crs, "ESRI
Shapefile")

print("Processing complete. Results saved to DKM_Intersection_Areas.shp.")
```

Make sure to replace 'DKM.shp' and 'roadplanning.shp' with the actual file paths of your shapefiles. This script will create a new layer called 'DKM_Intersection_Areas' with the desired attributes.

- **Notes:**

Despite the error in the instructions where it says to select the non-existing option to "Create multipart features. This solution works.
The python script runs, but does not produce the desired output.

- **Complexity score:**

Complexity score							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7	-
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0	-
	Sum			10		**0.7**	
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	2	3.3	-
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0	-
	Select by location	0...not used 1...used	1.00	2	0	0.0	-
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0	-
	Join by location	0...not used 1...used	1.00	1	0	0.0	-
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	1	1.5	-
	Sum			23		**4.8**	
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0	-
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0	-
	Sum			11		**0.0**	
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0	-
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0	-
	Layouting	0...not used 1...used	1.00	1	0	0.0	-
	Sum			7		**0.0**	
Total complexity score				51		**5.5**	

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0	-
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	1	2.5	-
	Sum			10		**7.5**	
Total workflow solution quality score				10		**7.5**	
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**B**	

Code solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0	-
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	1	1.7	-
	Sum			10		**6.7**	
Total code solution quality score				10		6.7	
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**C**	

6.11. Experiment: Area Coverage 3

- **Question structure and aim:**

Continues the series of Experiment 9. The goal is the same, but this time the prompt includes the word “overlay” and “analysis” to see if this entices ChatGPT to answer with the solution to this task that is based on the overlay analysis tool.

- **Chatlog:**

```
User
I have the following layers in QGIS:

- DKM
contains polygons
contains the field GST

- roadplanning
contains polygons
contains the field Type with possible values of Road, Parking, Plants

I want to analyse how much area of the roadplanning polygon overlays the DKM polygons.
Provide instructions on how to do this.

ChatGPT
To analyze how much area of the "roadplanning" polygons overlays the "DKM" polygons in
QGIS, you can use the "Intersection" tool to create a new layer representing the
overlapping area. Here are the step-by-step instructions:

1. **Load Your Layers**:
   - Make sure both the "DKM" and "roadplanning" layers are loaded in your QGIS project.

2. **Access the Processing Toolbox**:
   - Go to the "Processing" menu at the top of QGIS.
   - Select "Toolbox" to open the Processing Toolbox panel.

3. **Search for Intersection Tool**:
   - In the Processing Toolbox panel, use the search bar to search for the "Intersection"
tool.

4. **Run the Intersection Tool**:
   - Double-click on the "Intersection" tool to open the tool's settings.

5. **Set Input Layers**:
   - In the "Input layer" section, select the "roadplanning" layer as the first input
layer.
   - Select the "DKM" layer as the second input layer.

6. **Output Layer**:
   - In the "Output layer" section, choose where you want to save the resulting layer by
clicking the three dots (...) next to the field and specifying the output file location and
name.

7. **Run the Tool**:
   - Click the "Run" button to execute the Intersection tool. This will create a new layer
that represents the overlapping area between the "roadplanning" and "DKM" polygons.

8. **View the Result**:
   - Once the tool has finished running, you can add the resulting layer to your QGIS
project by right-clicking on the layer in the Layers panel and selecting "Add to Project."

9. **Calculate Overlapping Area** (optional):
   - To calculate the area of the overlapping polygons, you can open the attribute table of
the resulting layer. In the attribute table, you can add a new field and calculate the area
using the field calculator. To do this, create a new field with the expression `$area`,
which calculates the area of each polygon.
```

Now, you have a new layer that represents the overlapping area between the "roadplanning" and "DKM" polygons, and you can use the attribute table to see the area of each overlapping feature.

Remember to save your project and data as needed throughout the process.

User
Provide a pyton script that achieves this.
with D:\GIS\Exp10 as the working directory.

ChatGPT
You can achieve the same intersection analysis between the "roadplanning" and "DKM" layers using Python and the PyQGIS API. To do this, you can create a Python script and run it within the QGIS environment. Here's a Python script that performs this task:

```python
# Import the necessary QGIS modules
from qgis.core import QgsVectorLayer, QgsVectorFileWriter, QgsProcessingAlgorithm,
QgsProcessingParameterFeatureSource, QgsProcessingParameterFeatureSink
from qgis.PyQt.QtCore import QVariant
import os

# Define the working directory and set it as the project's home path
working_directory = r'D:\GIS\Exp11'
os.chdir(working_directory)

# Define the paths to your input layers
dkm_layer_path = os.path.join(working_directory, 'DKM.shp')
roadplanning_layer_path = os.path.join(working_directory, 'roadplanning.shp')

# Load the input layers
dkm_layer = QgsVectorLayer(dkm_layer_path, 'DKM', 'ogr')
roadplanning_layer = QgsVectorLayer(roadplanning_layer_path, 'roadplanning', 'ogr')

# Define the output layer path
output_layer_path = os.path.join(working_directory, 'Intersection_Result.shp')

# Create a processing algorithm for the Intersection operation
class IntersectionAlgorithm(QgsProcessingAlgorithm):
    INPUT_LAYER_1 = 'INPUT_LAYER_1'
    INPUT_LAYER_2 = 'INPUT_LAYER_2'
    OUTPUT_LAYER = 'OUTPUT_LAYER'

    def __init__(self):
        super().__init__()

    def name(self):
        return 'intersection_algorithm'

    def tr(self, text):
        return text

    def createInstance(self):
        return IntersectionAlgorithm()

    def processAlgorithm(self, parameters, context, feedback):
        layer1 = self.parameterAsVectorLayer(parameters, self.INPUT_LAYER_1, context)
        layer2 = self.parameterAsVectorLayer(parameters, self.INPUT_LAYER_2, context)
        output_layer = self.parameterAsOutputLayer(parameters, self.OUTPUT_LAYER, context)

        options = QgsVectorFileWriter.SaveVectorOptions()
        options.actionOnExistingFile = QgsVectorFileWriter.CreateOrOverwriteLayer

        QgsVectorFileWriter.writeAsVectorFormatV2(
            layer1,
            output_layer,
            QgsProcessing.FeatureSourceDefinition(layer1.wkbType(), layer1.sourceCrs(),
layer1.sourceExtent(), layer1.sourceFields(), layer1.sourceWkbType(),
layer1.sourceOptions()),
            options
        )
```

```
        for feature in layer2.getFeatures():
            intersection = feature.geometry().intersection(layer1.geometry())
            if not intersection.isGeosEmpty():
                new_feature = QgsFeature(layer1.fields())
                new_feature.setGeometry(intersection)
                output_layer.dataProvider().addFeatures([new_feature])

        return {self.OUTPUT_LAYER: output_layer}

# Register the Intersection algorithm
QgsApplication.processingRegistry().addProvider(IntersectionAlgorithm())

# Define the processing parameters
parameters = {
    'INPUT_LAYER_1': dkm_layer,
    'INPUT_LAYER_2': roadplanning_layer,
    'OUTPUT_LAYER': output_layer_path,
}

# Run the Intersection algorithm
feedback = QgsProcessingFeedback()
output = IntersectionAlgorithm().processAlgorithm(parameters, None, feedback)
```

Save this script as a `.py` file and run it within the QGIS Python Console. It will create an output layer called "Intersection_Result.shp" in your working directory, representing the overlapping area between the "roadplanning" and "DKM" layers.

- **Notes:**

Despite the prompt clearly hinting using the “overlap analysis”-tool, ChatGPT did not use this for its solution. Instead, it has gone for an intersect and area calculation, which is a correct solution. Despite using the same tool as in Experiment 10, this time it did not make up the “create multipart” option and instead explained its usage correctly. The code does not run, which is surprising, since the workflow is the same as suggested in Experiment 10, which failed to generate the desired output, but it managed to run.

- **Complexity score:**

Complexity score							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Filehandling	Number of input files to handle	0…1 file 1…<=2 files 2…<=5 files 3…>5 files	0.33	2	1	0.7	-
	Dataconversion	0…no datatypeconversion 1…simple shape type conversion 2…advanced datatype conversion 3…non-GIS native formats import	0.33	8	0	0.0	-
	Sum			10		**0.7**	
Vectorprocessing	Number of different vectorprocessing tools used	0…no tools used 1…<=2 2…<=5 3…>5	0.33	5	2	3.3	-
	Select from table	0…not used 1…simple attribute query 2…multipart query	0.50	7	0	0.0	-
	Select by location	0…not used 1…used	1.00	2	0	0.0	-
	Join from table	0…not used 1…join by existing key 2…composit key creation and join	0.50	5	0	0.0	-
	Join by location	0…not used 1…used	1.00	1	0	0.0	-
	Field calculation	0…not used 1…simple Field calculation 2…multipart expression	0.50	3	1	1.5	-
	Sum			23		**4.8**	
Rasterprocessing	Number of different rasterprocessing tools used	0…no tools used 1…<=2 2…<=5 3…>5	0.33	5	0	0.0	-
	Raster calculation	0…not used 1…simple attribute query 2…multipart query	0.50	6	0	0.0	-
	Sum			11		**0.0**	
Visualizing	Symbology	0…not used 1…categorized 2…graduated 3…rules based	0.33	2	0	0.0	-
	Labeling	0…not used 1…simple label 2…multipart expression	0.50	4	0	0.0	-
	Layouting	0…not used 1…used	1.00	1	0	0.0	-
	Sum			7		**0.0**	
Total complexity score				51		**5.5**	

- **Solution quality score:**

Workflow solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0	-
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	2	5.0	-
	Sum			10		**10.0**	
Total workflow solution quality score				10		**10.0**	
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**A**	

Code solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	1	1.0	-
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	0	0.0	-
	Sum			10		**3.0**	
Total code solution quality score				10		3.0	
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**	

6.12. Experiment: Calculating Volume

- **Question structure and aim:**

The task is to join area information to designated points and do a simple volume calculation based on it.

- **Chatlog:**

```
User
I have the following layers in QGIS:

- Trees
contains points of the individual trees

- WoodData
contains polygons
contains the field Height with the avearge height of the trees in meters
contains the field Diameter with the average diameter of the trees in centimeters

I want to calculate the total Volume of the trees.
Provide instructions on how to do this.

ChatGPT
To calculate the total volume of the trees in QGIS based on the data you have in the
"Trees" and "WoodData" layers, you'll need to follow these steps:

1. Make sure you have both the "Trees" (point layer) and "WoodData" (polygon layer) loaded
in your QGIS project.

2. You mentioned that the "WoodData" layer contains the average height and diameter of the
trees. To calculate the volume of each tree, you'll need to use a suitable formula. The
most common formula for calculating the volume of a tree is using the cylinder volume
formula:

   Volume (V) = π * (Diameter / 200)² * (Height / 100)

   Note that we're dividing the Diameter by 200 to convert it from centimeters to meters
and dividing the Height by 100 to convert it from centimeters to meters.

3. To calculate the volume for each tree, you can create a new attribute in the "Trees"
layer to store this information. To do this, go to the "Trees" layer's attribute table,
open the field calculator, and create a new field (e.g., "TreeVolume"). Use the formula
mentioned above in the field calculator to calculate the volume for each tree based on the
"Height" and "Diameter" attributes in the "WoodData" layer.

4. After calculating the volume for each tree in the "Trees" layer, you should have a new
attribute column with the volume values.

5. To calculate the total volume for all trees, open the attribute table of the "Trees"
layer, and in the bottom-right corner of the attribute table, you'll see the sum of the
"TreeVolume" column, which will give you the total volume of all the trees.

This process will give you the total volume of the trees based on the data you have in your
QGIS layers. Just make sure your data is correctly structured with the necessary attributes
before performing these calculations.

User
Provide a pyton script that achieves this.
with D:\GIS\Exp12 as the working directory

ChatGPT
You can achieve this in QGIS using Python scripting. Here's a Python script that calculates
the total volume of trees based on your data using the QGIS Python API. This script assumes
that you have already loaded the "Trees" and "WoodData" layers into your QGIS project:

```python
# Import the necessary QGIS libraries
```
```

```
from qgis.core import QgsProject, QgsVectorLayer, QgsExpression, QgsField, QgsFeature,
QgsFeatureRequest
from qgis.PyQt.QtCore import QVariant

# Set the working directory
working_directory = r'D:\GIS\Exp12'
QgsProject.instance().setFileName(working_directory)

# Get the "Trees" and "WoodData" layers
trees_layer = QgsProject.instance().mapLayersByName('Trees')[0]
wooddata_layer = QgsProject.instance().mapLayersByName('WoodData')[0]

# Define the formula for tree volume calculation
def calculate_tree_volume(height, diameter):
    return 3.14159265 * (diameter / 200) ** 2 * (height / 100)

# Create a new field for tree volume in the "Trees" layer
volume_field = QgsField('TreeVolume', QVariant.Double, 'Double')
trees_layer.dataProvider().addAttributes([volume_field])
trees_layer.updateFields()

# Initialize a list to store tree volumes
tree_volumes = []

# Calculate and update the volume for each tree
for tree_feature in trees_layer.getFeatures():
    tree_id = tree_feature.id()
    height = 0
    diameter = 0

    # Find the corresponding "WoodData" polygon
    for wooddata_feature in
wooddata_layer.getFeatures(QgsFeatureRequest().contains(tree_feature.geometry())):
        height = wooddata_feature['Height']
        diameter = wooddata_feature['Diameter']

    # Calculate the tree volume
    volume = calculate_tree_volume(height, diameter)

    # Update the "TreeVolume" field in the "Trees" layer
    trees_layer.startEditing()
    trees_layer.changeAttributeValue(tree_id,
trees_layer.fields().indexFromName('TreeVolume'), volume)
    trees_layer.commitChanges()

    # Append the volume to the list
    tree_volumes.append(volume)

# Calculate the total volume of all trees
total_volume = sum(tree_volumes)

print(f'Total Volume of Trees: {total_volume} cubic meters')
```

Make sure to save this script as a `.py` file and execute it within the QGIS Python Console or from an external Python environment. This script will calculate the total volume of the trees and print the result.

- **Notes:**

ChatGPT forgot to join the information of the two layers together before doing the calculation. Code does not run due to incorrect function calls.

- **Complexity score:**

Complexity score							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Filehandling	Number of input files to handle	0...1 file 1...<=2 files 2...<=5 files 3...>5 files	0.33	2	1	0.7	-
	Dataconversion	0...no datatypeconversion 1...simple shape type conversion 2...advanced datatype conversion 3...non-GIS native formats import	0.33	8	0	0.0	-
	Sum			10		**0.7**	
Vectorprocessing	Number of different vectorprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	1	1.7	-
	Select from table	0...not used 1...simple attribute query 2...multipart query	0.50	7	0	0.0	-
	Select by location	0...not used 1...used	1.00	2	0	0.0	-
	Join from table	0...not used 1...join by existing key 2...composit key creation and join	0.50	5	0	0.0	-
	Join by location	0...not used 1...used	1.00	1	1	1.0	-
	Field calculation	0...not used 1...simple Field calculation 2...multipart expression	0.50	3	1	1.5	-
	Sum			23		**4.2**	
Rasterprocessing	Number of different rasterprocessing tools used	0...no tools used 1...<=2 2...<=5 3...>5	0.33	5	0	0.0	-
	Raster calculation	0...not used 1...simple attribute query 2...multipart query	0.50	6	0	0.0	-
	Sum			11		**0.0**	
Visualizing	Symbology	0...not used 1...categorized 2...graduated 3...rules based	0.33	2	0	0.0	-
	Labeling	0...not used 1...simple label 2...multipart expression	0.50	4	0	0.0	-
	Layouting	0...not used 1...used	1.00	1	0	0.0	-
	Sum			7		**0.0**	
Total complexity score				51		**4.8**	

- **Solution quality score:**

Workflow solution quality								
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Workflow solution	Comprehensible	0…not understandable 1…intention is conveyed 2…clearly understandable 3…easy to follow guide	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	2	2.0	-
	Correct	0…incorrect 1…overall correct with minor errors 2…correct in all aspects	0.50	5	1	2.5	-
	Sum			10		**6.5**	
Total workflow solution quality score				10		**6.5**	
Total workflow solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**C**	

Code solution quality							
Section	**Indicator**	**Measure**	**Norm-alizing**	**Weight**	**Score**	**Weighted Score**	***Comment***
Code solution	Code readability	0…no code structure or proper naming 1…code is structured and readable 2…proper naming of variables and functions 3…code properly documented	0.33	2	3	2.0	-
	Complete	0…problem wasn't solved 1…solved to some extend 2…minor details missing 3…solved to full extend	0.33	3	3	3.0	-
	Correct	0…incorrect 1…major errors 2…overall correct with minor errors 3…correct in all aspects	0.33	5	0	0.0	-
	Sum			10		**5.0**	
Total code solution quality score				10		5.0	
Total code solution quality rating		**8 to 10 … A** **7 to 8 … B** **6 to 7 … C** **5 to 6 … D** **<= 5 … F**				**F**	

7. Results and Discussion

7.1. Evaluation matrix

Experiment	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Number of input files to handle	33	33	33	33	33	67	0	0	33	33	33	33
Dataconversion	0	0	0	0	0	0	0	0	0	0	0	0
Sum Filehandling	**7**	**7**	**7**	**7**	**7**	**13**	**0**	**0**	**7**	**7**	**7**	**7**
Number of different vectorprocessing tools used	67	33	0	0	0	0	0	0	67	67	67	33
Select from table	50	0	0	0	0	0	0	0	0	0	0	0
Select by location	0	100	0	0	0	0	0	0	0	0	0	0
Join from table	0	0	50	100	50	0	0	0	0	0	0	0
Join by location	0	0	0	0	0	0	0	0	0	0	0	100
Field calculation	0	100	0	0	100	0	0	0	50	50	50	50
Sum Vectorprocessing	**30**	**29**	**11**	**22**	**24**	**0**	**0**	**0**	**21**	**21**	**21**	**18**
Number of different rasterprocessing tools used	0	0	0	0	0	0	0	0	0	0	0	0
Raster calculation	0	0	0	0	0	0	0	0	0	0	0	0
Sum Rasterprocessing	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Symbology	0	0	0	0	0	67	67	100	0	0	0	0
Labeling	0	0	0	0	0	50	0	0	0	0	0	0
Layouting	0	0	0	0	0	0	0	0	0	0	0	0
Sum Visualizing	0	0	0	0	0	48	19	29	0	0	0	0
Total complexity score	**15**	**14**	**6**	**11**	**12**	**9**	**3**	**4**	**11**	**11**	**11**	**9**
Comprehensible	100	100	100	100	100	100	100	100	100	100	100	100
Complete	100	100	100	100	67	67	100	100	100	100	100	67
Correct	100	0	100	0	50	100	50	50	0	50	100	50
Total workflow solution quality rating	**100**	**50**	**100**	**50**	**65**	**90**	**75**	**75**	**50**	**75**	**100**	**65**
Code readability	100	100	100	100	100	100	100	100	100	100	100	100
Complete	100	100	100	100	67	100	100	100	100	100	33	100
Correct	33	0	33	0	33	0	0	33	0	33	0	0
Total code solution quality rating	**67**	**50**	**67**	**50**	**57**	**50**	**50**	**67**	**50**	**67**	**30**	**50**

Figure 3 – Evaluation matrix

The evaluation matrix, shown in Figure 3, visualises the results of all experiments on a single page. The results are shown as percentages of the maximum attainable score in each category. The values are color-coded for easy visualisation.
The complexity scores shown as a temperature-gradient, where 0% is shown in blue and 100% is shown in orange. The quality scores are color-coded by a traffic-light-gradient, where 0% is red, 50% yellow and 100% green.

The evaluation matrix shows that the type of GIS-task(s) required to solve the question, does not correlate with the success-rate, as the solution score is close to constant across all experiments, even with highly fluctuating complexity scores. That means that ChatGPT does not have an apparent blind-spot in its algorithm or training data for any particular type of task. Instead, it is roughly equally good (or bad) at everything GIS-related.

7.2. General assessment

This thesis was aimed at testing ChatGPT ability to solve GIS-related tasks. Experiments showed that the complexity threshold is extremely low, since ChatGPT fails even very basic tasks.

In general, it can be said that the workflows provided, were for the most part ok. They were well written across the board, but frequently unreliable. The python scripts provided failed to run almost all of the time, often due to multiple errors.

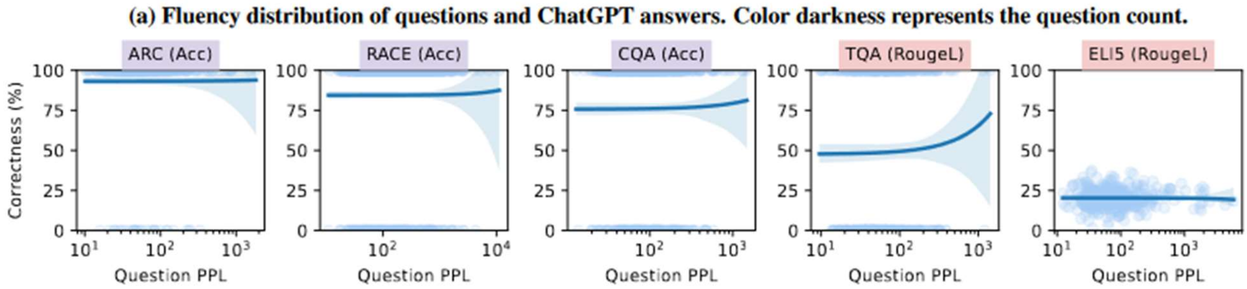

Figure 4 – ChatGPT answer correctness distribution, from paper "In ChatGPT we trust?"

ChatGPT is proven to be a reliable tool in certain fields, as seen by the correctness-percentage in Figure 4 (Shen *et al.*, 2023). It can be assumed that its failure on GIS-related tasks is due to its lack of GIS-related training data. GIS-applications in general are very well documented, with QGIS especially having a very detailed user guide/manual available online, covering basically all its functionalities. Its programming interface is also excellently documented in the form of the PyQGIS Cookbook, which covers all available functions with detailed explanations and template scripts. It is a safe assumption that this documentation is part of ChatGPT's training data, since it does have knowledge of the functions and this is by far the most probable source. It seems though, that a single source (or a low number of sources) is not enough for its algorithm to synthesise proper answers. It seems that it needs a bigger pool of data to process in its transformers, for its LLM to work properly in its current form and GIS compared to general software programming is too small of a field, and its documentation to centralized to provide that. To give a perspective, stackexchange (the most popular forum for programming) reports having 11.7 million registered users in total. 170,000 of those are registered with the GIS-specific exchange.

7.3. Evaluation of the workflow solutions

The second thing that catches the eye when looking at the evaluation matrix is the perfect comprehensibility-score across all experiments. Through the course of the experiments ChatGPT never failed to give a clear and structured answer. In the large majority of cases the answer explained all steps required for the solution and set the user to work with a fully comprehensible and clear guide.

While the workflows delivered by ChatGPT make a convincing impression, they fall apart when put to the test. While a good number of solutions do work, some ask the user to set parameters the processing tools do not provide, or ask for menu-options and function that simply do not exist in the first place. ChatGPT never says something along the lines of "I don't have sufficient data to answer…" or "I don't have certain solution, but you can try…" It always answers confidently and presents its solution as the definite one, even when it clearly lacks data to provide the information required. While it is correct in many cases, the fact that it presents made up information with the same degree of certainty that it presents factual information makes it a very unreliable tool.

Although the workflows provided are not reliable, they are extremely well structured and easy to follow, and I would still recommend its use to experienced users, despite the possible frustration of invented menus and non-existent options. I think it is still an excellent starting point if you just need a quick reference to the tools used to solve a specific task in GIS and have the ability to integrate that information into your own solution. The main benefit of ChatGPT is its ease of use. The official documentation, as well as the official discussion forums provide better quality information, but require a higher research effort.

7.4. Evaluation of the code solutions

All code solutions received perfect scores in code-readability across the board. The code was well structured, the variable names were chosen sensibly, and fit the information given in the task. The code was sufficiently documented and very easy to read for a user. It appeared to do everything that the task required, hence the almost flawless scores in completeness.

Sadly, the code solution correctness paints a dire picture. Like the workflows, the code is presented confidently as the correct and complete solution, but in the great majority of cases the code provided fails to run. The mishaps are numerous. From using unimplemented library functions, using depreciated function to interface with QGIS to passing wrong parameters to function calls. These errors are not based on ChatGPT's lack of training data past 2021. The functions were depreciated long before 2021. Other errors like using the wrong passthrough variables, are also not a result of a lack of recent training data, since the functions in question were never changed since they were first implemented. So even though the training data clearly includes all necessary information, ChatGPT fails to put together working python-scripts for QGIS.

This complete fail comes to a surprise, since ChatGPT is widely used as a programming aid and is included in numerous development environments. It seems to know about the function required to solve GIS-task, but is not able to use them properly. It stands to reason that ChatGPT has not passed a critical threshold on collecting enough training data on GIS-related tasks. ChatGPT does not warn you about its lack of data and passes its solution as the correct one regardless. Some parts of the solution do work, but it needs an experienced programmer to get it to run.

I see its use mostly in providing code snippets to integrate in your own scripts, which can speed up programming tasks for proficient user who already know exactly how their script is supposed to look like and use ChatGPT as an aid to type it out and provide snippets that would otherwise be sourced from *stackoverflow* or similar sites. I do not see Novice PyGIS-users finding any use in ChatGPT's scripting capabilities, since it is in no way reliable enough to be considered as a tool that independently automates tasks. Fixing the broken code it provides, takes more time and yields less learning experiences than doing it from scratch in the first place.

7.5. Ironing out the Weaknesses

While it may make sense to integrate ChatGPT into your GIS working environment, its usefulness is limited. The first and most obvious improvement is expanding its training data on GIS-related topics. This would obviously lead to less errors and higher quality answers. As shown in Figure 2 ChatGPT's apparent intelligence is an emergent ability that does not scale linearly with the amount of training data, but leads to sharp increases in its capabilities at certain thresholds. It seems that, especially on the PyGIS-tasks, this threshold is not yet passed. The good news is that ChatGPT training data is continuously expanding and its reliability will increase even with no major changes made to the inner workings of its algorithm.

The main weakness though lies at the very heart of the algorithm. As explained in Chapter 2.2 of the introduction, ChatGPT strings together word after word to form the most probable answer. My evaluation shows that this algorithm managed to reliably put together a well formulated answer, which looks sound at first glance, but frequently contains wrong information. This happens since ChatGPT has no way of testing its output for plausibility. While I cannot see a way of implementing plausibility checks in its current algorithm, but I do see the possibility to add a form of confidence score or confidence rating based on the amount of training data available on the task at hand. Alternatively, it could change the phrasing of the answer to incorporate this. Instead of presenting an answer as the definitive answer it could state that it has a solution that "might work" or add a disclaimer to the running text that it has not enough data to give a certain answer. This way the users knows what to expect regarding the quality of the answer and is able to be more critical towards it, if ChatGPT indicates that its confidence level is low.

7.6. Similarities and differences to related works

My findings are in stark contrast to the findings in the related paper "Ch(e)atGPT?" (Stutz *et al.*, 2023) where ChatGPT was able to pass a beginners GIS-Programming course by submitting the prompt results "as is". Throughout all my tests, ChatGPT did not produce working code from the initial prompt and was only able to get it to work occasionally through further iteration. That sort of performance surely would not let a student pass a class. Unfortunately, the paper did not include an exact documentation of the tasks to be solved, nor did it include the prompts that were provided to ChatGPT. Through this lack of data, it is impossible to find the definitive reason for this discrepancy in outcome. This leaves several possible reasons for the difference in outcome of the tests:

- **Programming language used:**
  While Python is a common language used in GIS it is not the only one available.

Rank	Business Insider - ranking based on GitHub	IEEE Survey of members (2018)	Stack Overflow Developer Survey (2019)	GoGeomatices: Top Languages in the GIS World	Full Stack Academy (9 best to learn)	Women Who Code
1	JavaScript	Python	JavaScript	Python	JavaScript	Python
2	Java	C++	HTML/CSS	JavaScript	Swift	Java
3	Python	Java	SQL	R	Java	JavaScript
4	PHP	C#	Python	SQL	C/C++	Rust
5	C++	R	Java	Java	Python	Kotlin
6	C#	PHP	Bash/Shell	C/C++	PHP	_
7	TypeScript	JavaScript	C#	C#	Ruby	_
8	Shell	Go	PHP	-	C#	-
9	C	Assembly	C++	-	Rust	-
10	Ruby	Matlab	TypeScript	-	-	-

Figure 5 – Common Programming Languages in the Geospatial Industry by PennState College of Earth and Mineral Sciences

  Figure 5 shows a ranking of the most used programming languages by different sources in the Geospatial Industry. It shows that Python is popular, but that there are many alternatives. Especially JavaScript is widely used. It is a possibility that ChatGPT was asked to solve the tasks in a different language and that it performed better than if it were tasked to solve it in Python.

- **GIS-library used:**
  In my tests I did not specify the library to be used and ChatGPT defaulted to using mostly the Geopandas-library and less frequently PyQGIS. Although the library used is not specific to the software used (you can use PyQGIS in ArcGIS and ArcPy in QGIS and vice versa) there is a high possibility that ChatGPT defaults to a different library depending on what software you specify in the prompt. There is

also a good chance that ChatGPT can handle different libraries better or worse than others. The documentation style and detail as well as the amount of user discussion that is available online and part of its training data are not equal to one another.

- **Semantics:**
  The way you formulate your prompt influences the output, even if the same information is given. For example, prompting *"Write a thank you letter to the ZGIS department naming 3 random staff members"* will not produce the desired output and instead will produce a generic letter without the required personalisation. Prompting *"Name 3 random staff members from the ZGIS- department. Then write a thank you letter to the ZGIS-department naming those 3 people"* will produce a personalised letter, despite the information given being the same and (from a human perspective) more awkwardly worded request. This means that semantics matter greatly and might be the reason why ChatGPT performs so poorly in my tests, despite being given all the information needed to perform the tasks.

- **Training data available on the task:**
  Since ChatGPT was asked to solve tasks regularly given out to students, there might be more relevant training data on these particular tasks. Maybe the exact tasks were discussed by students on different forums and are part of ChatGPT's training data. This seems more far-fetched at first, but if you consider that tasks for students often have a certain scheme even through different organizations and that students in general are more active in online discussions than workers in the field it becomes a realistic scenario.

8. Conclusion

8.1. Prelude

ChatGPT algorithm is first and foremost designed to synthesize human language. It does not understand logic and it does not differentiate if it pulls its sentences from a comprehensive set of data or from basically thin air. It is designed to string together sentences that have the best probability of fitting the user's input. The apparent intelligence only emerges if a certain threshold of data in the LLM is reached.

8.2. Answering the research questions

- **Does ChatGPT provide truthful information?**
  No. While ChatGPT tries its best to answer with truthful information, when it does lack the proficiency to do so, it will make up the most probable, but wrong answer. The user has no way to check if ChatGPT is sufficiently equipped to answer a particular question and so cannot trust the accuracy of the information he/she is given by ChatGPT.

- **Can ChatGPT provide useful assistance on GIS workflows?**
  Yes. The explanations provided by ChatGPT on how to solve various tasks, while not always correct, do at the very least guide the user in the right direction and present the user with relevant GIS-tools, which can be used to adapt their own solution.

- **Is ChatGPT able to provide the necessary PyQGIS code to automate GIS workflows?**
  No. The usefulness of the programming code provided by ChatGPT is very limited. Due to numerous errors in the code, the time investment needed to get it running is so high that programming it from scratch might be the better option in the majority of cases. It is however useful in providing code snippets to integrate in your own solution, but it does take a proficient user who already knows how the code is supposed to look beforehand and just uses ChatGPT to save the time writing it out.

- **Is the code ChatGPT generates well structured, so a regular user can understand and modify it?**
  Yes. The code provided by ChatGPT (while mostly erroneous) is very well structured and documented. Variable and function-names are derived from the set task and are highly descriptive. All this makes it very easy to understand and modify.

- **Does ChatGPT struggle with more complex issues?**
  Yes. In its current state, it already struggles with very simple tasks.

- **Are there certain topics/solution-steps where ChatGPT falls flat?**
  No. My research has not shown a particular blind spot for a certain topic. My data conclude that ChatGPT is equally good (or bad) at everything GIS-related.

- **Overall, is ChatGPT a useful tool for a GIS-specialist?**
  Yes. Despite its limitations, ChatGPT can provide useful assistance to a GIS-specialist. While one should not rely on ChatGPT's answers, they do provide good input, ideas, relevant tools, and even possible solutions.

8.3. Findings

My research has shown that ChatGPT's ability to provide instructions to solve GIS-tasks is unreliable. While it was able to provide full workflows encompassing all subtasks every single time, the information given was often not truthful. This unreliability does not correlate with the task's complexity level.

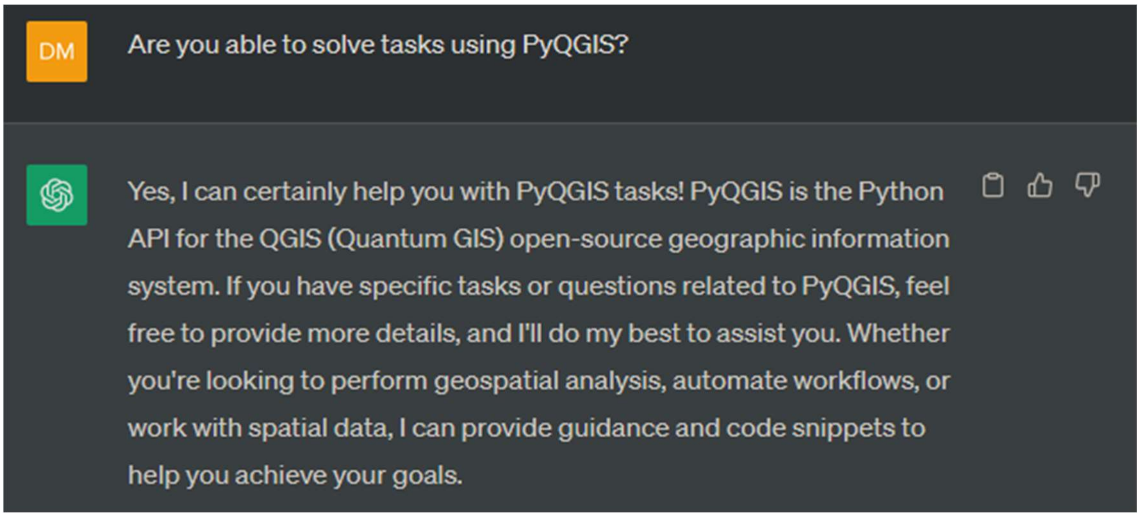

Figure 6 – ChatGPT self-reports on its PyGIS capabilities

While ChatGPT claims to be able to solve tasks using PyQGIS, my investigation has shown that in reality this is not within ChatGPT's capabilities. During my testing, it was not able to solve a single task using PyQGIS. It does, without exception, provide the code asked for. While this code is well structured, in the vast majority of cases it fails to run. In the rare cases where it does, it does not provide the requested output.

My assessment is that ChatGPT is able to aid GIS-professionals by providing selective assistance, but it is not able to fully solve or automate tasks on its own.

9. Limitations

9.1. Reproducibility of experiments

The web-client provided by OpenAI always represents the most recent version of ChatGPT. There is no option to revert to a previous version of the software in the client. In addition, OpenAI does not offer a downloadable version of ChatGPT to the public. This means that old versions of ChatGPT are neither accessible nor in any way preservable.

As of writing of this master thesis in 2023, the current version of ChatGPT is GPT-3.5. Its training data cutoff point is January 2022. The fluid state of the software means that experiments that may be conducted in this thesis may not be reproduceable, since the output may change due to the continuously updated training data.

Even if a particular version of ChatGPT could be preserved it is important to note that reproducibility of experiments would still be limited, since the LLM's are designed to vary their responses, even for fixed inputs. This means that it could very well give a proper answer one time and a wrong answer the next. An examination of the variance of possible answers to the same question is not provided, only one answer is prompted and evaluated.

9.2. Omission of the refactoring capabilities of ChatGPT

One key feature of ChatGPT is its capability to refactor its answer by further user input. This is useful if users have provided insufficient information or is pointing out errors in ChatGPT's answers.

The first case can be ignored if all necessary information is provided in the initial question.
In the second case the user needs to identify that error in the answer in the first place. Which means the user needs to already know the correct answer beforehand and provide appropriate hints to ChatGPT to properly refactor. There is no longer a clear line between what knowledge is provided by ChatGPT and what knowledge is added indirectly by the user.
Furthermore, it begs the question up to which point refactoring is acceptable. At some point the user is just answering his/her own question.

Since refactoring may muddle the water between ChatGPT's and the user's abilities, this work omits this feature on purpose and evaluates the single answer to a single complete question.

10. Further Research

ChatGPT is a very broad topic and there are many angles to explore. This thesis is just the tip of the iceberg and opens up many possible approaches to expand on the topic of the use of ChatGPT in GIS.

I see two main approaches to the topic. First the analysis of the technical capabilities, which this thesis contributes to, and secondly an exploration on how ChatGPT is or may be used by different user-groups and how it influences them.

The following topics would be interested to explore and to expand on this work:

- **Can ChatGPT make GIS more accessible for different user-groups?**
  It can be explored, if ChatGPT enables laymen and beginners to solve GIS tasks with little to no prior knowledge. Does it provide proper guidance even without usage of the proper technical terms? Can ChatGPT guide users through tasks without in depth knowledge on the subject, or can it aid in the learning experience.

- **How does the wording of the prompt influence the response accuracy?**
  The comparison of my thesis to related papers has shown a great discrepancy in ChatGPT GIS-capabilities. This inconsistency could not be explained in this work. To get to the bottom of this, a testing procedure could be setup to explore the difference in ChatGPT responses depending on the wording used.

- **How can refactoring improve the initial response?**
  One big feature of ChatGPT that was not explored in this thesis is its ability to refactor answers with additional user input. It can be assessed, how the quality and accuracy can be improved through each iteration of refactoring.

- **Can the direct implementation of ChatGPT in a GIS-software UI improve work performance?**
  Many development environments now include ChatGPT integration directly into their UI. To date, none of the major GIS-software packages includes such a feature, with only FME having announced to implement such a feature in the near future. A prototype-plugin could be assembled and its impact on work performance measured.

11. Bibliography

Biswas, S. (2023) 'Role of ChatGPT in Computer Programming.: ChatGPT in Computer Programming.', *Mesopotamian Journal of Computer Science*, 2023, pp. 8–16. Available at: https://doi.org/10.58496/MJCSC/2023/002.

Else, H. (2023) 'Abstracts written by ChatGPT fool scientists', *Nature*, 613(7944), pp. 423–423. Available at: https://doi.org/10.1038/d41586-023-00056-7.

Gordijn, B. and Have, H. ten (2023) 'ChatGPT: evolution or revolution?', *Medicine, Health Care and Philosophy*, pp. s11019-023-10136–0. Available at: https://doi.org/10.1007/s11019-023-10136-0.

GPT-3, Osmanovic-Thunström, A. and Steingrimsson, S. (2022) 'Can GPT-3 write an academic paper on itself, with minimal human input?'

Klievtsova, N. *et al.* (2023) 'Conversational Process Modelling: State of the Art, Applications, and Implications in Practice'. arXiv. Available at: https://doi.org/10.48550/arXiv.2304.11065.

Nature editorial, 613 (2023) 'Tools such as ChatGPT threaten transparent science; here are our ground rules for their use', *Nature*, 613(7945), pp. 612–612. Available at: https://doi.org/10.1038/d41586-023-00191-1.

Rudolph, J., Tan, Samson and Tan, Shannon (2023) 'ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?', *Journal of Applied Learning and Teaching*, 6(1). Available at: https://doi.org/10.37074/jalt.2023.6.1.9.

Shen, X. *et al.* (2023) 'In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT'. arXiv. Available at: https://doi.org/10.48550/arXiv.2304.08979.

Stokel-Walker, C. (2023) 'ChatGPT listed as author on research papers: many scientists disapprove', *Nature*, 613(7945), pp. 620–621. Available at: https://doi.org/10.1038/d41586-023-00107-z.

Stutz, P. *et al.* (2023) *Ch(e)atGPT? An Anecdotal Approach on the Impact of ChatGPT on Teaching and Learning GIScience*. preprint. EdArXiv. Available at: https://doi.org/10.35542/osf.io/j3m9b.

Surameery, N.M.S. and Shakor, M.Y. (2023) 'Use Chat GPT to Solve Programming Bugs', *International Journal of Information Technology & Computer Engineering (IJITC) ISSN : 2455-5290*, 3(01), pp. 17–22. Available at: https://doi.org/10.55529/ijitc.31.17.22.

Susnjak, T. (2022) 'ChatGPT: The End of Online Exam Integrity?' arXiv. Available at: https://doi.org/10.48550/arXiv.2212.09292.

Vaswani, A. *et al.* (2017) 'Attention is All you Need', in *Advances in Neural Information Processing Systems*. Curran Associates, Inc. Available at:

https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (Accessed: 28 March 2023).

Wei, J. *et al.* (2022) 'Emergent Abilities of Large Language Models', *Transactions on Machine Learning Research* [Preprint]. Available at: https://openreview.net/forum?id=yzkSU5zdwD (Accessed: 29 March 2023).