Karl Oskar Magnus Holm

LLMS - The Death of GIS Analysis?

An Investigation into Using Large Language Models to Make GIS Analysis Simpler, Faster, and More Accessible

Specialization Project in Computer Science and Geomatics, June 2024
Supervisor: Hongchao Fan
External supervisors from Norkart: Alexander Salveson Nossum, Arild Nomeland and Rune Aasgaard

Department of Geomatics
Faculty of Engineering
Norwegian University of Science and Technology

Abstract

Sammendrag

Preface

Karl Oskar Magnus Holm
Trondheim, 4th December 2023

Contents

List of Figures

List of Tables

1. Introduction

1.1. Background and Motivation

The field of Large Language Models (LLMs) is an emerging one. Fan et al. (2023, p. 2) found that the proportion of papers about LLMs[1] to arXiv has skyrocketed since 2020, with a six-times increase in percent points from 2022 to 2023. They write that prompt engineering has been extensively used as a way to improve code generation (Fan et al., 2023, p. 7)

1.2. Goals and Research Questions

The overarching goal of this specialization project is to investigate how Large Language Model (LLM) can be utilized to make GIS analysis simpler, faster, and more accessible. As exemplified in the task description provided by Norkart (see appendix A), such a system should be able to create a meaningful response to a query such as:

> "Find all buildings within a 100-meter belt that are above 100 meters above sea level and have docks."

The task then is to investigate how modern language models can be used to perform classical GIS analyses using standard GIS technologies like PostGIS and geospatial data catalogues adhering to OGC or STAC standards. Based on the task description I have constructed three research questions that I will attempt to answer in this specialization project report:

1. What is the potential of LLM-based GIS analysis?
2. How can OGC API Features be used in an overlay analysis using ChatGPT-4?
3. How can we give ChatGPT-4 access to external tools?

Researching the potential of using LLMs in GIS could uncover new methodologies for spatial analysis, predictive modeling, and decision-making. The aim with RQ1 would be to assess the capabilities and limitations of integrating machine learning algorithms with Geographic information systems (GISs), and also touch on how a mature LLM-based GIS could impact the daily work of (human) GIS professionals.

[1] Including articles whose title or abstract includes "LLM", "Large Language Model", or "GPT".

Column1	arXiv	Google Scholar	Web of Science	PapersWithCode
Autonomous AI	2342	234	2	342
GIS	2342	234	2	342
Conditional random field	2342	234	2	342
Geospatial standardization	2342	234	2	342
LLM	2342	234	2	342
GPT	2342	234	2	342
BERT	2342	234	2	342
Transfer learning	2342	234	2	342
Foundation model	2342	234	2	342
Embedding	2342	234	2	342
Fine-tuning	2342	234	2	342
Prompt engineering	2342	234	2	342
Retrieval-augmented generation	2342	234	2	342

Table 1.1.: Literature study search results.

RQ2 focuses on the feasibility of using the OGC API - Features Standads in a typical overlay analysis within a conversational AI context like ChatGPT-4, having the users be able to express themselves using natural language queries. An answer to RQ2 would describe how OGC APIs can be called and manipulated in a flexible manner during a conversation to perform spatial queries or analyses, like intersecting layers or filtering features based on certain criteria.

RQ3 delves into the technical and ethical considerations of expanding ChatGPT-4's capabilities through integration with external tools, such as GIS software or data analytics platforms. It would explore options for secure and efficient data exchange, and assess the implications of such access in terms of data privacy and user consent.

1.3. Research Method

My research methods consist of a literature study and a set of experiments/proofs of concept, the latter of which is described in detail in chapter 3.

1.3.1. Literature Study

I set out by deciding on some search terms, and used a "snowball sampling" technique to find relevant articles. Table 1.1 shows the number of search results on various sites.

Fix table data and styling

1.4. Contributions

1.5. Thesis Structure

2. Theory

Chapter 2 of this specialization project will talk about the leading technologies in the field of Large Language Model (LLM), which in itself is a subfield of Natural Language Processing (NLP). section 2.1 will go over the leading LLM models, their strengths and weaknesses, and briefly mention differences in model architecture. Section 2.3 will name the most prominent providers of LLM services.

Figure 2.1 shows an actor map which includes stakeholders, providers, and other groups and organizations that could have some relevance to an autonomous LLM-based GIS-agent. Following subsections will go over the different groups and explain why they are included in the actor map.

2.1. LLMs

Disclaimer: I will reuse parts of the Background section of a paper I wrote in the theory module called "TDT13 - Advanced Text Analytics and Language Understanding", for this section on Large Language Models. This includes the subsection 2.1.1 and subsection 2.1.3.

This section will open with an explanation of the building block of most modern Large Language Models, namely the Transformer, which employs self-attention. Subsection 2.1.2 and subsection 2.1.3 will then discuss the two most famous families of LLMs, namely the GPT's and the BERT's. Section something will discuss other modern LLMs which, though not as famous, are powerful and deserve mentions. Subsection 2.2.1 and subsection 2.2.2 will then enlighten the reader about the different ways in which Large Language Models can be benchmarked and how we can measure the quality of a natural language response produced by an LLM.

All?

ref

2.1.1. Attention and The Transformer Architecture

Vaswani et al. (2017) managed to achieve new state-of-the-art results for machine translation tasks with their introduction of the Transformer architecture. The Transformer has later been proved effective for numerous downstream tasks, and for a variety of modalities. Titling their paper *Attention Is All You Need*, Vaswani et al. suggest that their attention-based architecture renders Recurrent Neural Networks (RNNs) redundant, due to its superior parallelization abilities and the shorter path between combinations of position input and output sequences, making it easier to learn long-range dependencies (Vaswani et al., 2017, p. 6).

Figure 2.1.: Actor map for stakeholders, providers, and other groups and organizations that could have some relevance to an autonomous LLM-based GIS-agent.

The Transformer employs self-attention, which enables the model to draw connections between arbitrary parts of a given sequence, bypassing the long-range dependency issue commonly found with RNNs. An attention function maps a query and a set of key-value pairs to an output, calculating the compatibility between a query and a corresponding key (Vaswani et al., 2017, p. 3). Looking at Vaswani et al.'s proposed attention function (2.1), we observe that we take the dot product between the query Q and the keys K, where Q is the token that we want to compare all the keys to. Keys similar to Q will get a higher score, e.g., be *more attended to*. These differences in attention are further emphasized by applying the softmax function. The final matrix multiplication with the values V, being the initial embeddings of all input tokens, will give us a new embedding in which all tokens have some context from all other words. We improve the attention mechanism by multiplying queries, keys, and values with weight matrices learned through backpropagation. Self-attention is a special kind of attention in which queries, keys, and values are all the sequence.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Attention blocks can be found in three places in the Transformer architecture (Vaswani et al., 2017, p. 5) (I will use machine translation from Norwegian to German as an example):

1. In the encoder block to perform self-attention on the input sequence (which is in Norwegian)
2. In the decoder block to perform self-attention on the output sequence (which is in German)
3. In the decoder block to perform cross-attention (or encoder-decoder attention) where each position in the decoder attends to all positions in the encoder

The Transformer represented a breakthrough in the field of NLP, and is the fundamental building block of LLMs like BERT.

2.1.2. The GPT Family

Generative Pre-trained Transformers (GPTs) are a type of LLM first introduced by OpenAI in 2018 (Radford and Narasimhan, 2018).

2.1.3. The BERT Family

Bidirectional Encoder Representation from Transformers (BERT) is a family of language models which was first introduced in 2018 and is designed to facilitate a wide range of downstream tasks (Devlin et al., 2019, p. 5). The BERT architecture consists of stacked bidirectional Transformer encoders. The self-attention mechanism allows for training of deep bidirectional representations. The input sequence is transformed into

embeddings (vector representations). These per-token embeddings include information about the meaning of the word itself, the meaning of the sentence/segment it belongs to, and the token's position in the full input. These embeddings then pass through a stack of Transformer encoders (12 and 24 for $\mathbf{BERT_{BASE}}$ and $\mathbf{BERT_{LARGE}}$, respectively), allowing the model to learn more complex patterns and of different granularities (token, sentence, document) (Devlin et al., 2019, p. 5).

In the BERT framework, there are two training steps, namely the pre-training and fine-tuning procedures. BERT is pre-trained on two NLP tasks. One is Masked Language Modelling (MLM), in which 15% of words are masked with the special `[MASK]` token and are left for the model to predict (Devlin et al., 2019, p. 4). The MLM task helps the model learn bidirectional representations. The second of the two unsupervised tasks used during pre-training is Next Sentence Prediction (NSP), where the special `[CLS]` token (found at the start of each tokenized sequence) is used to predict if a sentence `B` follows `A`. During this pre-training step, the input sequence looks like this:

`[CLS]` this is sentence A `[SEP]` and this is sentence B `[SEP]`

The `[CLS]` token is used to label sentence B as either `IsNext` or `NotNext`.

BERT is normally fine-tuned to specific downstream tasks by using the `[CLS]` token, which captures an aggregated representation of the input sequence. This vector representation can then be used as input to a classification layer for tasks like multi-label classification and regression.

PaLM 2

Codey

2.1.4. Meta Products

Like PaLM 2 The language models developed by Meta (previously Facebook) differ from the products from OpenAI in that Meta's LLama models are open-source.

LLama 2

Code LLama

2.1.5. Honourable Mentions

Falcon 180B

Mistral

Vicuna

2.2. Benchmarking and Evaluation of LLMs

There are several ways of benchmarking Large Language Models but this section will focus on HumanEval, Multitask Language Understanding (MMLU), and Grade School Math 8K

(GMS8K). According to metrics displayed on the various pages for these datasets found on PapersWithCode, they seem to be the three most common datasets for measuring the different abilities of a given LLM.

2.2.1. Benchmarks

HumanEval

HumanEval is a dataset of handwritten problems used to measure functional correctness for synthesizing programs for docstrings (Chen et al., 2021, pp. 2–4).

Multitask Language Understanding

First introduced by Hendrycks et al. (2021) Multitask Language Understanding is a way of tesing a Large Language Model's multitask accuracy, covering 57 tasks including mathematics, computer science, and others.

Grade School Math 8K

This is a dataset made to measure an LLM's abilities to perform mathemathical arthimetic.

2.2.2. Evaluation Metrics

Write

2.3. Large Language Model (LLM) Providers

2.3.1. OpenAI

Being the most widely famous actor within the field of LLMs, OpenAI has gained great influence through their vast portfolio.

2.3.2. Microsoft Azure

2.3.3. Google Cloud

2.3.4. Amazon Web Services

2.3.5. Anthropic

2.4. Geospatial Standards

2.4.1. International Standardization Work

OGC Api Standard

The Open Geospatial Consortium (OGC) API Standards serve as the glue in the field of Geographic information system (GIS), paving the way for interoperability and data exchange between diverse systems. Leveraging common web protocols like HTML and

supporting multiple data formats including JSON, GML, and HTML. The OGC API standard provides a modular architecture consisting of a core specification and various extensions. This modularity allows for flexibility, enabling users to customize their services according to specific needs. According to their webpages, they provide 80 different standards, each for a specific geospatial purpose. Notable examples are 3D Tiles, CityGML, GeoTiff, and OGC API - Features (OGC, 2023).

OGC API Standards function as modern replacements to older standards like WMS and WFS, and presents an evolved and more adaptable framework for spatial data operations, setting the stage for future innovations in the GIS domain.

STAC Api Standard

The SpatioTemporal Asset Catalog (STAC) API is a standardized way to expose collections of spatial temporal data for online search and discovery. Built upon a JSON core, it aims to be a uniform and flexible environment from which developers can customize the API infrastructure to their domain. STAC API provides a powerful query language that allows users to search by various parameters like time, location, and keywords, making widely applicable. The STAC community has also defined specification in order remove the complexity associated with having to create unique pipelines when consuming different spatial-temporary collection. The significance of the STAC API lies in its ability to democratize access to large volumes of geospatial data. By offering a common standard for data cataloguing and discovery, it reduces the barriers that often exist due to incompatible data formats. Developers or GIS professionals can take advantage of this through built-in tooling in QGIS, a desktop GIS for viewing, editing, and analysing spatial data, or through third-party packages in the Python and R programming languages. The API is also accessible through the command line interface when using GDAL (*STAC Tutorials* n.d.).

As OGC board member Chris Holmes puts it: "The STAC API implements and extends the OGC API — Features standard, and our shared goal is for STAC API to become a full OGC standard." (Holmes, 2021).

2.4.2. Norwegian Standardization Work

Skriv om til ikke-gpt-språk

Geospatial standardization work has been on the agenda of Norwegian governing powers for decades and have materialized in frameworks/collaborations like Geovekst and Norge digitalt, as wells as the SOSI file format. Subsection 2.4.2 will delve into the work that has been done and what is expected for the future. The reasoning for the conclusion of this section is that the constraints of this specialization project is set by the Norwegian borders, and thus it is important to be aware of the standards that apply.

SOSI

Samordnet Opplegg for Stedfestet Informasjon (SOSI) is a Norwegian file format for storing and exchanging geospatial data. It was first introduced in 1987 and has since

approached international standards, the most important arenas currently being ISO/TC 211 and OGC (Mardal et al., 2015). SOSI is the adopted Norwegian standard for creating and delivering digital geographic data, administered by the Norwegian Mapping Authority (Statens kartverk) (Mæhlum and Rød, 2023).

In a SOSI dataset, terrain points, lines, and polygons are represented by their coordinates and classified into various object types according to the SOSI object catalog standard. However, there are few GIS systems that can read SOSI data directly, so data in SOSI format usually needs to be converted to another GIS-readable data format (Mæhlum and Rød, 2023).

Geovekst

Geovekst is a collaborative initiative in Norway aimed at collecting, managing, and distributing geospatial information. Established in 1992, it is a partnership between national, regional, and local government bodies, as well as several private companies. Geovekst's primary focus is on creating a comprehensive, standardized geographical database for Norway that is easily accessible and updated regularly. It has played a vital role in various planning and development projects across the country, from urban planning to environmental conservation.

Unlike other geospatial initiatives, Geovekst emphasizes shared responsibilities and costs among its partners. This cooperative model ensures consistent data quality and efficient use of resources. It utilizes a variety of data sources, including aerial photographs, laser scans, and mapping, making it a rich resource for both public and private sectors. Moreover, its open-access policy allows for wider dissemination of geospatial information, thus encouraging innovation and informed decision-making across multiple disciplines.

Norge digitalt

Established in 2005, Norge Digitalt is a more recent framework compared to Geovekst and is the name of Norway's national spatial data infrastructure. Norge Digitalt primarily involves governmental bodies (national, regional, and municipal), but also educational and research institutions and companies with responsibilities on a nation-wide scale; examples include Telenor and local and regional energy companies (Norge Digitalt, 2023, p. 6). Norge Digitalt aims to coordinate and streamline all geospatial activities in Norway, making it easier for users to discover, access, and use spatial data.

One key feature of Norge Digitalt is its focus on international standards and interoperability. While Geovekst is primarily a national initiative, Norge Digitalt aims to integrate Norway's geospatial data with that of other European countries. It supports a wide range of data formats and follows international standards, including those set by the Open Geospatial Consortium (OGC). The framework also provides various tools and services, like metadata catalogues and web services, making it a comprehensive solution for geospatial data management and distribution in Norway.

2.5. User Groups

There are several user groups that could take advantage of an AI-based agent with general geographic and GIS knowledge. Questions can span from simple retrieval questions such as "How many people live in Trondheim" and "How long is the drive from Oslo to Bergen?", to more complicated questions that require problem-solving abilities and reasoning. While it is difficult to obtain dataset over common queries, Kumar (2023), creater of the chatbot app Pocket AI[1], shared a dataset of ~13k user queries from his app along with classifications of these. Salient categories were:

- "task oriented" (23.1%)
- "informational" (20.2%)
- "social" (16.2%)
- "personal advice and self-improvement" (13.1%)

The main takeaway from these numbers is that the main motivation for use is productivity.

This aligns with the results of Skjuve et al. (2023) from their questionnaire-based study performed in late January 2023, about three months after its release (OpenAI, 2023). The goal with the study was to find out *why* people use ChatGPT. They found that most participants (55%) are motivated by productivity, and specifically applying it for routine tasks, information retrieval, text generation and writing support, and software development (Skjuve et al., 2023, pp. 17–21). Table 2.1 shows all categories and their frequencies. There were 197 samples in total, and more than one category could be assigned to each sample. It is worth noting that the study is likely to have included early adopters, and might therefore make the results less representative for the time at which this report is written (4th December 2023), now that use patterns have become more established (Skjuve et al., 2023, p. 37).

Table 2.1.: Categories and frequency of ChapGPT usage (Skjuve et al., 2023, pp. 16–17).

Category	% **(n)**
Productivity	55% (109)
Novelty	51% (101)
Fun and amusement	20% (41)
Creative work	18% (34)
Social interaction and support	9% (18)
Other	7% (15)

Given that the main reason people use conversational AI is for productivity, whether in a professional, academic, or personal context, such technology could be highly beneficial

[1] https://github.com/varunon9/pocket-ai

in a geospatial setting. 67 out of the 197 participants in Skjuve et al. (2023, p. 18) highlighted "ChatGPT's ability to understand complex queries" and that it is "efficient in alleviating the need to experiment with different phrasings of the query", as is often needed when 'Googling' for an answer to a specific question. This ease of information retrieval, along with it's problem-solving abilities (Skjuve et al., 2023, p. 20), could also make conversational AIs highly relevant for geospatial purposes, and in the field of GIS. The following sections will elaborate on the potential user groups presented in Figure 2.1 that could benefit from such an artificial, and spatially aware, companion.

2.5.1. General Public

2.5.2. GIS Professionals

2.5.3. City Planners

2.5.4. Business Analysts

2.5.5. Academics and Researchers

2.5.6. Emergency Services

2.6. Regulatory Bodies and Privacy Concerns

2.6.1. Local Regulations

2.6.2. The European Union

2.7. Related Work

2.7.1. GIS with LLMs

Roberts et al. (2023) investigated extent of GPT-4's geospatial awareness through a set of case studies with increasing difficulties, starting with general factual tasks and finishing with complex questions such as generating country outlines and travel networks.

Li and Ning (2023) states that "autonomous GIS will need to achieve five autonomous goals: self-generating, self-organizing, self-verifying, self-executing, and self-growing.", and provide a "divide-and-conquer"-based method to address some of these goals. Furthermore, they propose a simple trial-and-error approach to addressing the self-verifying goal. They also highlight need of a memory system in a mature LLM-based GIS system, referring to the use of vector databases in autonomous agents like AutoGPT (Richard, 2023). Even with its shortages, the solution that (Li and Ning, 2023) provide, called LLM-Geo, is able to solve provide good solutions in various case studies by providing executable assemblies in a Python environment when provided with URLs to relevant data sets, along with a user-specified query.

Zhang et al. (2023) uses the LangChain framework (Chase, 2022) in order to combine different GIS tools in a sequence in order to solve different sub-goals, and focuses on using the semantic understanding and reasoning abilities of LLMs like (e.g., ChatGPT) to

call externally defined tools, employing the LLM as an agent or controller. The authors take great inspiration from the AutoGPT framework (Richard, 2023). The externally defined tools are described (manually) by its name and description. Said description contains information about the input parameters and output types of the tools/functions. Tools are defined for geospatial data collection, data processing and analysis, and data visualization. The effectiveness of the system is showcased in four case studies.

2.7.2. Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is tightly interwoven with explainable AI, being a framework for retrieving facts from an external knowledge base to allow a LLM-based agent access to accurate up-to-date information (Martineau, 2023). A common problem when working with language models, especially those designed to be general-purpose, is hallucination; that is, when the model provides an answer that is completely wrong but in a very convincing manner. While progress is being made with newer models even the better ones, like GPT-4, gives an incorrect answer about 1 out 5 times, and even worse for certain categories of queries (for instance 'code' and 'business') (OpenAI, 2023, p. 10). Retrieval Augmented Generation can help mitigate this problem.

LangChain

LangChain (Chase, 2022) is an open-source project that provides tooling that can be used to create autonomous AI agents. It is designed to help with prompt management and optimization, creating chains of calls to LLMs, data augmentation, autonomous agent creation, and memory-related tasks.

Microsoft Semantic Kernel

Microsoft Semantic Kernel is an SDK that functions as the brain of an autonomous agent and provides connectors to models and memory, and connects to triggers and actions.

AutoGPT

Richard (2023) will try to split a task into subtasks and use the internet and other tools in an automatic loop to solve the task/subtasks.

2.7.3. Prompt Engineering and Planning Strategies

Zhou et al. (2023) introduces a framework called Language Agent Tree Search (LATS) "that synergizes the capabilities of LLMs in planning, acting, and reasoning.". As of writing (October 30th, 2023), the LATS framework is the highest scoring model on the HumanEval benchmark.

3. Experiments and Results

3.1. Experimental Plan and Setup

The experiments conducted for this report have been planned to answer the research questions described in section 1.2.

3.1.1. RQ1: Determining the Potential of LLM-based GIS analysis

RQ1 differs from RQ2 and RQ3 in that it is more open-ended. The tests will try to display what abilities GPT-4 has geospatial tasks out of the box, without providing it with any context or external tools.

Ability to Perform Geospatial Analysis

Tests were performed to assess ChatGPT's ability to perform geospatial analysis. The testing approach is inspired by the work of Roberts et al. (2023) (see subsection 2.7.1), who did experiments with increasing difficult on GPT-4 to characterize what GPT-4 knows about the geographical world, highlighting both capabilities and limitations. These focused on GPT-4's general geospatial awareness, and were not concerned with GIS-related tasks. Therefore, I will refer to Roberts et al. (2023) when highlighting its somewhat surprising geospatial awareness abilities, and focus my efforts to displaying its potential for use in the world of GIS. I will do this by constructing various tests that try to reflect its GIS knowledge.

I will use the Elveg 2.0 dataset (The Norwegian Mapping Authority, 2019), along with cadastral data. In order to assess ChatGPT's ability to read and understand different data formats, the data will be provided in both SOSI, GML, and GeoJSON format. Datasets for the first two formats were downloaded from https://geonorge.no, while the GeoJSON datasets were created using a custom Bash script which converts from GML to GeoJSON using the `ogr2ogr` program from GDAL. The Elveg 2.0 dataset contains a range of different layers for different types of geometries. In order to simplify the experiments, only the layer named "Fartsgrense" (eng. "Speed limit") was used fro Elveg 2.0.

Below are the questions that were asked, in rising order of predicted complexity.

1. "Provide a summary of the file contents, highlighting the file's most salient features."
2. "Provide a visual representation of the file contents."
3. "Find the mean location of the building locations."

4. "Extract all roads with a speed limit greater than or equal to 80 km/h."
5. "Select all buildings located within 50 metes of a high-speed road (speed limit >= 80 km/h)."
6. "Find the area best suited for expansion to accommodate residential buildings."

Some follow-up questions are added when needed, in order help the model understand the questions or when it stops and asks for permission to go forth with analysis.

Data Access

Another important thing to test is the issue of providing ChatGPT with relevant files on which it can perform analysis. ChatGPT Plus users will have access a range of advanced features, including web browsing with Bing, Dall-E Image Generation, and Code Interpreter. The latter of these allows the user to manually upload files into the chat instance and perform advanced analyses on the contents of these, which is what was used to upload the datasets for the tests above. While this is very powerful, having to manually upload files poses some limitations. A more flexible system should be capable of accessing web APIs in real time.

A dataset containing the border of Drammen Municipality was used to test compare ChatGPT's ability to perform analyses on manually uploaded data, versus data handed through it from passing a URL address. The data conforms to the GeoJSON standard and contains a FeatureCollection object with a single Feature, namely the border. When file/URL has been provided, ChatGPT is simply asked to present a visual presentation of its contents.

3.1.2. RQ2: Testing ChatGPT's Ability to Perform Overlay Analysis using OGC API Features

Experiments on RQ2 will require three different elements:

1. Provide ChatGPT with API URL(s) to relevant data collections corresponding with the OGC API - Features specification
2. Prompts to ChatGPT-4 using the "Code Interpreter" beta feature
3. Gold standard/expected output for the given combination of input data and prompts

These experiments will be limited to the collections found at https://alenos-tester001.azurewebsites.net/. This example OGC API was created by Norkart's Alexander Salveson Nossum for with the purpose of testing OGC API Features on Norwegian data. It was created using `pygeoapi`[1], which is a Python server implementation of the OGC API suite of standards. It allows for deployment of a RESTful OGC API endpoint using OpenAPI, GeoJSON, and HTML.

[1] https://pygeoapi.io/

	GML	GeoJSON	Shapefile
Summary	Yes	Partially	Partially
Plotting	When guided	When guided	Yes
Mean location	When guided	Yes	No
Filtering	No	Yes	Yes
Buffer + Intersect	No	No	No
Planning for expansion	No	Partially	No

Table 3.1.: The first three rows of the training dataset

3.1.3. RQ3: Testing ChatGPT's Ability to Use External Tools

Experiments on RQ3 will test different tools and techniques intended to give an LLM access to up-to-date information and external tools (see subsection 2.7.2 on Retrieval Augmented Generation). The main focus will be on how one can use a tool like LangChain hook up a conversational AI with sophisticated functionality found in various GIS software.

3.2. Experimental Results

3.2.1. Results for RQ1 Tests RQ1

3.2.2. Ability to Perform Geospatial Analysis

Preliminary experiments showed that ChatGPT's Code Interpreter is unable to read and write SOSI files. It was unable to manipulate the data directly and was also unable to convert the file into a more suitable format, failing to convert it GeoJSON using GDAL's `ogr2ogr`. SOSI is therefore excluded from Table 3.1, which shows the successfulness of ChatGPT to perform analyses on various file formats.

As Table 3.1 shows, ChatGPT's Code Interpreter did not manage to properly analyze the GML data without guidance. It created a parser that was difficult to use for further analysis. When guided into using the GeoPandas library, which accepts GML data, it managed to plot the contents and calculate a centroid. The buffering and intersection task "was interrupted due to its time-consuming nature", and it did not make an attempt at solving the planning task due to inability to analyze the GML files.

With the GeoJSON data, ChatGPT had difficulties reading the files and could not provide a good summary consistently. It *was* able to plot the data, but that had to be done in separate responses for each of the datasets. It was able to find the high-speed roads, but could not figure out which buildings were within a 50-meter buffer of these. However, when asked to plan for expansion to accommodate residential buildings, it managed to achieve a result close to what was expected in the "Buffer + Intersect" task. It accomplished this by creating a grid and figuring out which grid cells were within 50 meters of a high-speed road. This did not extract a subset of the building points—which would have been the desired output—but it had some minor value in terms of visualization

(see Figure 3.1). Though the result is useful, it was closer than in the GML attempt to what would be an expected response.

Using the shapefile formatted data, ChatGPT was able to produce a decent summary of the data, but the attribute names were cut off after about 10 characters. It was, however, able to produce quite good visual representations of both dataset, colouring the roads differently by their speed limits and the buildings by their building type.

3.2.3. Data Access

3.2.4. Results for RQ1 Tests RQ1

3.2.5. Results for RQ1 Tests RQ1

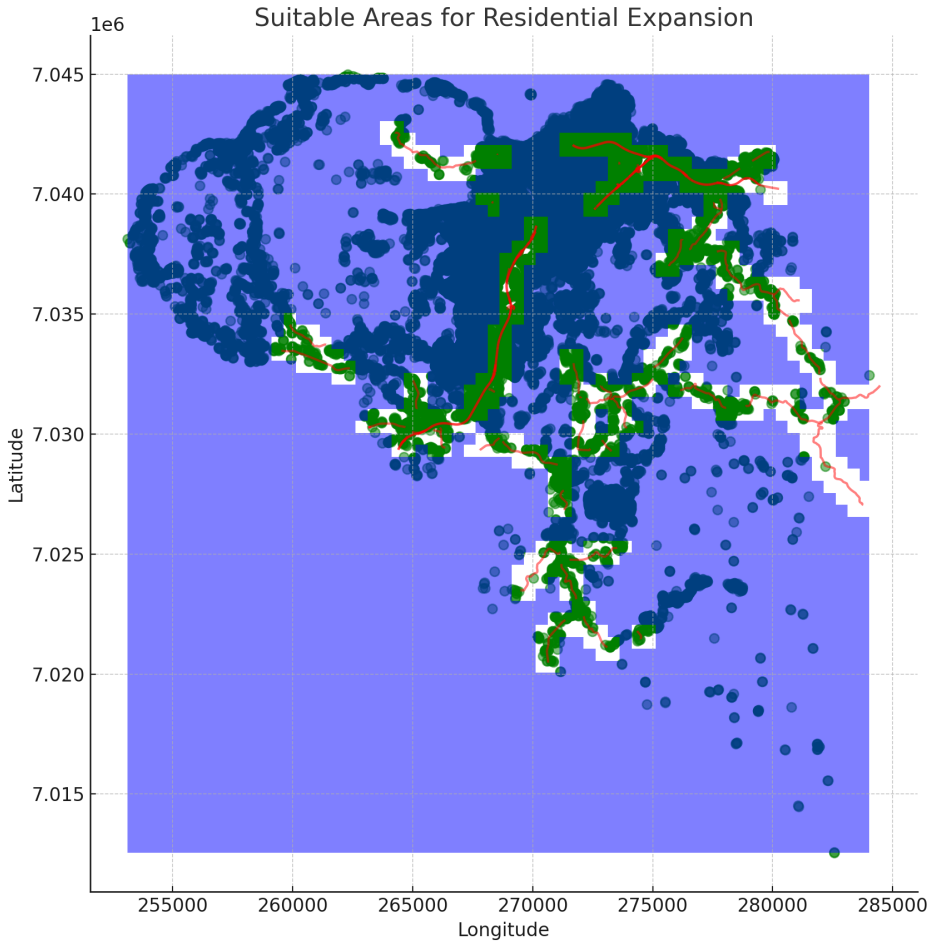

Figure 3.1.: The result of ChatGPT when asked to "Find the area best suited for expansion to accommodate residential buildings", using provided GeoJSON datasets. Potentially suitable areas for residential expansion are depicted in blue.

4. Discussion

4.1. Evaluation

4.2. Discussion

Yiu et al. (2023) calls AI technologies like Large Language Models "powerful imitation engines" but claim that they are unable to innovate and that significant advancements in AI development is required for them to be able to learn the in the same way human children do.

4.2.1. Using the In-Built ChatGPT Code Interpreter for Geospatial Analysis

When using ChatGPT's Code Interpreter with file uploads, it became apparent that it runs in a Linux environment, and that it uses a mounted drive in the `/mnt` director, which is used for temporarily mounted filesystems. From the initial experiments where the same data in different file formats was tested it tried to a GDAL command (`ogr2ogr -f "GeoJSON" {converted_geojson_path} {sosi_file_path}`) to perform a conversion from SOSI to GeoJSON, the latter of which is far easier to manipulate in a Python environment. This test failed, and the system's response was that "the `ogr2ogr` tool is not available in this environment".

This result was not very surprising, especially since the driver needed to read and write SOSI files—which is called *fyba* and is developed by The Norwegian Mapping Authority[1]—is almost certaintly not available in the standard Linux environment for ChatGPT's Code Interpreter. Seeing as the SOSI standard still is widely used for Norwegian geospatial purposes (though expected to be exchanged with the GML format in the future), it is important for an LLM-based GIS agent focused on the Norwegian market to be able to handle this file type.

The inability to manipulate the Linux environment using by the Code Interpreter clearly poses some limitations on the systems. A solution to the problem is to create a custom environment on a server and implement agent-like capabilities by other means (LangChain, AutoGPT, AutoGen, etc.). Having the agent run on an environment that we control ourselves gives us greater flexibility, and we can then allow the agent to access powerful GIS tooling, such as the GDAL library. This also allows us to avoid having to perform I/O on a mounted directory (in the `/mnt` directory), which can increase the speed of reads and writes.

[1] https://github.com/kartverket/fyba

5. Conclusion and Future Work

5.1. Contributions

5.2. Future Work

5.2.1. Test regime

In order to test the feasibility of different language models to serve as the brain of an autonomous GIS agent, a testing regime should be developed. In the examples of autonomous GIS agents described in the literature study of this report (see subsection 2.7.1), results have generally been presented in the form of case studies (Li and Ning, 2023; Zhang et al., 2023). This type of qualitative testing is entirely appropriate to showcase the possibilities of the technologies but may be insufficient when comparing performance of different systems. In the latter case a quantitative approach would probably be preferable.

One idea is to create a test dataset which consists of inputs and corresponding desired outputs of typical GIS tasks. Inputs would in this case be natural language queries inputted by a mock user, and the output would be what you would expect a GIS professional to return when given the same tasks/queries. Inputs should reflect the varying level of GIS knowledge in the different user groups (see section 2.5). Outputs could be files with typical geospatial extensions (.shp, .geojson, .sosi, etc.), or they could adhere to API schemas specified by geospatial standards (see section 2.4).

While the inputs should be fairly simple to construct there are several questions to be answered in regard to the outputs:

- How does one evaluate the accuracy of the output?
- How should the AI agent respond when the user does not specify an output file format?
- How does one evaluate the usefulness of outputs to questions that should not return geospatial files, e.g. answers to general questions about geo-related subjects?

These are questions outside the scope of this specialization project. They will, however, be pursued in my master thesis.

5.2.2. Framework for Planning, Acting, and Reasoning

5.2.3. Embeddings

5.2.4. Fine-Tuning

Bibliography

Chase, H. (2022). LangChain.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., & Zaremba, W. (2021). Evaluating Large Language Models Trained on Code.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., & Zhang, J. M. (2023). Large Language Models for Software Engineering: Survey and Open Problems.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring Massive Multitask Language Understanding.

Holmes, C. (2021). SpatioTemporal Asset Catalogs and the Open Geospatial Consortium.

Kumar, V. (2023). What are people asking to ChatGPT?

Li, Z., & Ning, H. (2023). Autonomous GIS: The next-generation AI-powered GIS.

Mæhlum, L., & Rød, J. K. (2023). SOSI. *Store norske leksikon*.

Mardal, G., Borreb, M., Christensen, L., Jetlund, K., Ryghaug, P., & Hokstad, I. (2015). Nasjonal strategi for videreutvikling av SOSI.

Martineau, K. (2023). What is retrieval-augmented generation?

Norge Digitalt. (2023). *Generelle vilkår for Norge Digitalt-samarbeidet* (tech. rep.).

OGC. (2023). OGC Standards.

OpenAI. (2023). GPT-4 Technical Report.

Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training.

Richard, T. B. (2023). AutoGPT: The heart of the open-source agent ecosystem.

Roberts, J., Lüddecke, T., Das, S., Han, K., & Albanie, S. (2023). GPT4GEO: How a Language Model Sees the World's Geography.

Skjuve, M., Bae Brandtzaeg, P., & Følstad, A. (2023). Why People Use ChatGPT.

STAC Tutorials. (n.d.).

The Norwegian Mapping Authority. (2019). *Elveg 2.0* (tech. rep.).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need.

Yiu, E., Kosoy, E., & Gopnik, A. (2023). Transmission Versus Truth, Imitation Versus Innovation: What Children Can Do That Large Language and Language-and-Vision Models Cannot (Yet). *Perspectives on Psychological Science*, 17456916231201401.

Zhang, Y., Wei, C., Wu, S., He, Z., & Yu, W. (2023). GeoGPT: Understanding and Processing Geospatial Tasks through An Autonomous GPT.

Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., & Wang, Y.-X. (2023). Language Agent Tree Search Unifies Reasoning Acting and Planning in Language Models.

Appendices

A. Task Description from Norkart

Oppgåve med omfang som kan tilpassast både prosjekt og masteroppgåve

LLMs - GIS-analysens død

(kan justerast seinare)

BAKGRUNN

Nyere modeller for kunstig intelligens har demonstrert spesielt gode evner til å kunne lære av store mengder ustrukturert og semi-strukturert informasjon. ChatGPT fra OpenAi tok verden med storm – og chat-baserte systemer florerer. Kan chat-baserte modeller skapes for å hente ut GIS-data effektivt? Norkart har en stor dataplattform hvor brukere utvikler mot API'er som i stor grad har GIS/Geografiske data i bunn. GeoNorge er en stor datakatalog hvor brukere slår opp, eller søker kategorisert for å finne data. QGIS, Python, PostGIS, FME og andre verktøy brukes ofte til å gjennomføre GIS-analyser – hvor en GIS-analytiker/data-scientist gjennomfører dette.

«*Finn alle bygninger innenfor 100-meters-belte som er over 100 kvm og har brygger*»

Er dette mulig å få til med dagens tilgjengelige chat-modeller?

OPPGAVEBESKRIVELSE

Oppgaven har som hovedmål å undersøke hvordan nyere språkmodeller kan benyttes for å gjennomføre klassiske GIS-analyser ved å bruke standard GIS-teknologi som PostGIS/SQL og datakataloger (OGC API Records fks). Hva finnes av tilgjengelig chat-løsninger? Hvordan spesialtilpasse til GIS-anvendelser? Hvor presise kan en GIS-Chat bli?

Relevante delmål for oppgaven:

1. Kartlegge state-of-the-art
2. Utvikle proof-of-concepts
3. Analysere begrensninger og kvalitet

Oppgaven vil med fordel deles i prosjektoppgave og masteroppgave

- Prosjektoppgave
  - State-of-the-art: Ai-modeller og multi-modal maskinlæring
  - Innhente og utvikle datagrunnlag og API-tilgjengelighet
- Masteroppgave
  - Utvikle proof-of-concepts med tilgjengelige åpne modeller/teknologi
  - Gjennomføre eksperimenter for analyse av kvalitet

Detaljert oppgavebeskrivelse utvikles i samarbeid med studenten.

ADMINISTRATIVT/VEILEDNING

Ekstern veileder: (en eller flere)
Mathilde Ørstavik, Norkart
Rune Aasgaard, Norkart
Alexander Nossum, Norkart

Aktuelle vegleiarar og ansvarleg professor ve NTNU (den som har fagansvar nærast oppgåva):

Terje Midtbø (GIS, kartografi, visualisering)
Hongchao Fan (3D modellering, fotogrammetri, laser)

B. API Schemas

B.1. OGC API - Features

OGC API specification for a 'collection' object[1]:

```
        type: object
        required:
        - id
        - links
        properties:
        id:
        description: identifier of the collection used, for example, in URIs
        type: string
        example: address
         title :
        description: human readable title of the collection
        type: string
        example: address
        description:
        description: a description of the features in the collection
        type: string
        example: An address.
         links :
        type: array
        items:
        $ref: link.yaml
    example:
      - href: http://data.example.com/buildings
       rel: item
      - href: http://example.com/concepts/buildings.html
       rel: describedby
        type: text/html
  extent:
    $ref: extent.yaml
  itemType:
  description: indicator about the type of the items in the collection (the default value is 'feature').
    type: string
    default: feature
    crs:
    description: the list of coordinate reference systems supported by the service
    type: array
    items:
      type: string
```

[1] https://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/schemas/collection.yaml (retrieved October 25, 2023)

default :
  – http://www.opengis.net/def/crs/OGC/1.3/CRS84
example:
  – http://www.opengis.net/def/crs/OGC/1.3/CRS84
  – http://www.opengis.net/def/crs/EPSG/0/4326

B.2. STAC API

C. Code Examples