Karl Oskar Magnus Holm

LLMs - The Death of GIS Analysis?

An Investigation into Using Large Language Models for GIS Data Analysis

Specialization Project in Computer Science and Geomatics, December 2023
Supervisor at NTNU: 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

The emergence of powerful Large Language Models (LLMs) with remarkable reasoning and coding capabilities—like GPT-4, the latest additions to the GPT series—enables automation of a wide range of tasks. ChatGPT's Code Interpreter is able to generate, execute, and review its own code, making development of autonomous AI agents far easier than before. This specialization project report explores the feasibility of LLM-based GIS agents, investigating how ChatGPT is currently being used in the field of GIS, how such LLMs could be used if applied in larger systems, and how such systems can be implemented. This report seeks to highlight the strengths of current LLM-based technologies, but also their weaknesses, and suggest areas of improvements and possible solutions to overcome current limitations. These goals are achieved through a literature study and three experiments that aim to support the findings of the literature study. The literature study presents the body of work that has already been done in regard to the position of LLMs in the field of GIS, as well as planning strategies applied in LLM-based agents, and retrieval-augmented generation—that is, giving the LLM "hooks" into the real world. The three experiments focus on ChatGPT's ability to handle geospatial data in various formats and through different access channels. The overall goal of this specialisation project is to lay the groundwork for development of LLM-based GIS agents.

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. Furthermore, they write that prompt engineering has been extensively used as a way to improve code generation (Fan et al., 2023, p. 7). LLM technologies provide great opportunities for automation of tasks in a variety of fields, possibly also in the field of Geographic Information System (GIS). With the capabilities of ChatGPT's Code Interpreter to generate, execute, and review its own code, research should be done to see if these abilities can be applied to GIS-related tasks.

1.2. Goals and Research Questions

The overarching goal of this specialization project is to investigate how Large Language Model 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 user queries such as:

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

The task is then to investigate how Large 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, in order to answer such a query. Based on the task description, three research questions have been constructed that this specialization project report will attempt to answer:

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

RQ1 aims to assess the capabilities and limitations of integrating LLMs with GISs. Researching the potential of using LLMs in GIS could uncover new methodologies for spatial analysis, predictive modelling, and decision-making.

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

RQ2 focuses on the feasibility of using the OGC API - Features standard 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. Examples of such analyses could be intersecting of map layers or filtering features based on certain criteria.

RQ3 delves into the technical considerations of expanding ChatGPT-4's capabilities through integration with external tools, such as GIS software or data analytics platforms. It would also evaluate options for efficient data exchange, such as web APIs.

1.3. Contributions

The main contributions of this specialization project are listed below:

1. Gather relevant literature to facilitate development of state-of-the-art LLM-based GIS agents.
2. Conduct tests on ChatGPT-4 to explore its proficiency in manipulating geospatial data across different formats and access channels.

1.4. Thesis Structure

The thesis is divided into four main sections:

- Chapter 2 lays the theoretical groundwork needed to understand the work.
- Chapter 3 discusses previous work related to the thesis.
- Chapter 4 explains the planning and setup for testing, and presents the results of the tests.
- Chapter 5 discusses finding from the tests, suggests areas of importance when developing LLM-based GIS agents, and concludes the project report.

2. Theory

Disclaimer: Parts from a paper written in the theory module "TDT13 - Advanced Text Analytics and Language Understanding" will be reused in the Theory chapter. This includes subsection 2.1.1 and subsection 2.1.3.

Chapter 2 of this specialization project will lay the theoretical groundwork needed to understand the thesis. This includes central technologies and other considerations related to Large Language Models (LLMs) and Geographic Information Systems (GISs). Figure 2.1 shows an actor map which includes these LLMs and their providers, as well as potential user groups, regulatory bodies that could affect adoption of AI-based technologies, and some relevant geospatial standards. Subsections 2.1 through 2.5 will elaborate on each of the clusters of the actor map, starting with LLMs.

2.1. Large Language Models

Large Language Models (LLMs) have proved proficient at Natural Language Understanding (NLU) and Natural Language Generation (NLG), which are both subfields of Natural Language Processing (NLP). Figure 2.1 includes six different LLMs that are all explained in the subsections below. All but the BERT model are made with text generation in mind, with BERT being more adapted for the task of predicting masked tokens, e.g. "I love going to the `[MASK]` during summer.". BERT is very efficient at NLU, and is applicable for a wide range of downstream tasks, one of which is discussed in section 3.1. As mentioned, the other LLMs in Figure 2.1 are geared towards text generation, with certain members of the GPT family and Google's Gemini having multimodal capabilities as well. As subsection 3.1.1 will illustrate, these generative models—known for their great conversational abilities and extensive knowledge from pre-training on large corpora—have been found useful in GIS analysis. The combination of conversational and visual skills—particularly in multimodal models—holds significant promise for GIS analysis, where visual interpretation of maps is crucial.

This section will open with an explanation of the fundamental building block of modern Large Language Models: the Transformer. 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. Subsection 2.1.4 will present the brand new multimodal Gemini model from Google, and subsection 2.1.5 will list various open-source alternatives.

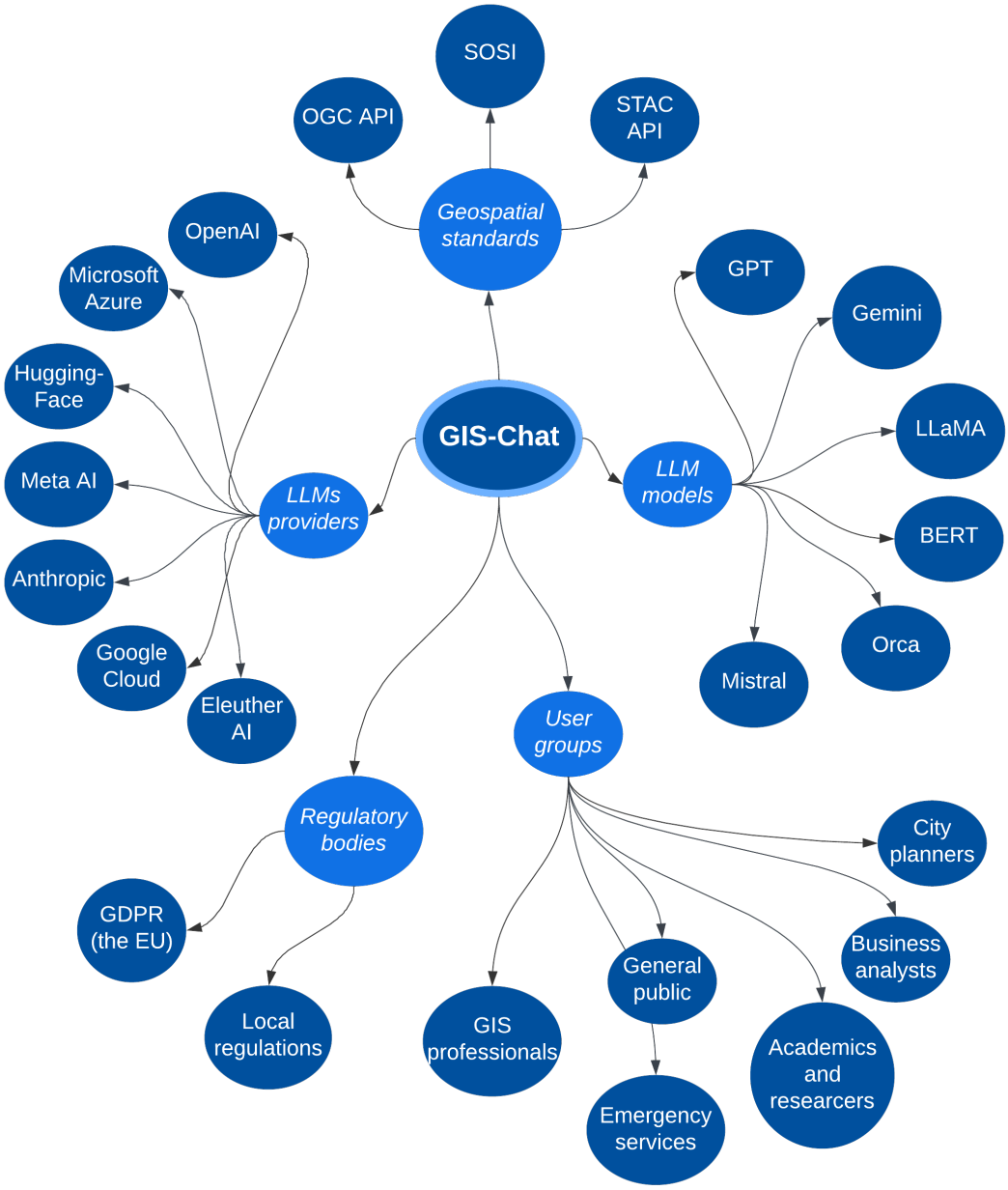


Figure 2.1.: Actor map for stakeholders, providers, and other groups and organizations that could have some relevance to an autonomous LLM-based GIS agent

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 network architectures like 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).

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 it takes 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, i.e., 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 the input tokens, will yield a new embedding in which all individual tokens have some context from all other tokens. We improve the attention mechanism by multiplying queries, keys, and values with weight matrices that are learned through backpropagation. Self-attention is a special kind of attention in which queries, keys, and values are all the same 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 (also known as 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 modern LLMs, most famous of which are the GPT's.

2.1.2. The GPT Family

Generative Pre-trained Transformer (GPT) is a type of LLM that was introduced by OpenAI in 2018 (Radford and Narasimhan, 2018). Specifically designed for text generation, a GPT is essentially a stack of Transformer *decoders*, and demonstrates through its vast pre-training on unlabelled data that such unsupervised training can help a language model learn good representations, providing a significant performance boost while alleviating the dependence on supervised learning. While the original Transformer architecture as described by Vaswani et al. (2017) was intended for machine translation—thus having encoders to learn the representation of the origin language representation of a given input sequence and decoders to learn the representation in the target language and perform cross-attention between the two—the GPT is designed only to *imitate* language. This is why there are no encoders to be found in the GPT architecture, only decoders. The model employs masked multi-head attention (running the input sequence through multiple attention heads in parallel), and is restricted to only see the last k tokens—with k being the size of the context window—and is tasked to predict the next one.

Training consists of two stages: unsupervised pre-training and supervised fine-tuning. The former is used to find a good initialization point, essentially teaching the model to imitate the corpora upon which it is trained. This results in a model that will ramble on uncontrollably, just trying to elaborate upon the input sequence it's given to the best of its knowledge. This will naturally produce undefined behaviour, and it is therefore necessary to fine-tune the model on target tasks in a supervised manner. Radford and Narasimhan (2018, p. 4) explain how the model can be fine-tuned directly on tasks like text classification, but how one for other tasks needs to convert structured inputs into ordered sequences because the pre-trained model was trained on contiguous sequences of text. In the case of ChatGPT, OpenAI used Reinforcement Learning from Human Feedback (RLHF) by employing a three-step strategy: first training using a supervised policy, then using trained reward models to rank alternative completions produced by ChatGPT models, before fine-tuning the model using Proximal Policy Optimization (PPO), which is a way of training AI policies. This pipeline is then performed for several iterations until the model produces the desired behaviour (OpenAI, 2022).

2.1.3. The BERT Family

Bidirectional Encoder Representation from Transformers (BERT), introduced about four months after Radford and Narasimhan (2018) presented the GPT architecture, is a family of language models which was first introduced in 2018 and is designed to facilitate a wide range of downstream NLP tasks (Devlin et al., 2019, p. 5). The BERT architecture consists of stacked bidirectional Transformer *encoders*. This makes BERT unsuitable for text generation, unlike the *decoder*-based GPT architecture. However, the self-attention mechanism in the encoder—in which tokens can *see* both past and future tokens—allows for training of deep bidirectional representations. The input sequence is first 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 patterns of different granularities (token, sentence, document) (Devlin et al., 2019, p. 5).

The BERT framework consists of two training steps: the pre-training and fine-tuning procedures. BERT is pre-trained on two NLP tasks. One of these 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 these unsupervised tasks 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 utilizing the `[CLS]` token representation after the last encoder layer, 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.

2.1.4. Gemini

As of writing, Google's Gemini (Gemini Team and Google, 2023), introduced on December 6th 2023, is the latest addition to the ever-growing pool of Large Language Models. Being fundamentally designed for multimodality, it is able to reason between text, images, video, audio, and code. It is being released in three different sizes: the Gemini Ultra, the Gemini Pro, and the Gemini Nano. The Gemini Ultra produces state-of-the-art performance on out of 32 widely-used academic benchmarks, and performs worse than OpenAI's GPT-4 on only one benchmark, namely the HellaSwag benchmark for common-sense reasoning for everyday tasks. The Gemini Ultra outperforms GPT-4 on the Multitask Language Understanding (MMLU) benchmark, various reasoning benchmarks, and shows significant improvements in maths and code related benchmarks (Gemini Team and Google, 2023, p. 7). Gemini performs better than OpenAI's multimodal equivalent in GPT-4V on all benchmarks (Gemini Team and Google, 2023, p. 12). Some of these benchmarks are discussed further in subsection 3.4.2.

Like the GPT architecture, Gemini is built on top of Transformer decoders. Gemini is trained to accommodate various modalities, supporting interleaved sequences of text, image, audio, and video as inputs (Gemini Team and Google, 2023, pp. 3–4). This highlights one of the greatest strengths of the Transformer architecture, namely that it can be adapted for multiple modalities. Commonly, multimodal AI architectures consist of an ensemble of models—one for each given modality—with different representations that can be difficult to combine. The Transformer solves this issue and provides a common

architecture that can be trained end-to-end.

2.1.5. Open-Source Alternatives

Seeing as the state-of-the-art models of today, like GPT-4 and Gemini Ultra, are generally closed-source, it is important to note that there are viable open-source alternatives out there. This section lists the most prominent ones.

LLaMA

Perhaps the most famous family of open-source LLMs are Meta AI's LLaMA (Large Language Model Meta AI) models, with LLaMA 2 being the latest addition (Touvron et al., 2023). LLaMA 2 is a powerful family of pre-trained and fine-tuned LLMs that outperformed open-source chat models on most benchmarks at its release. It also shows great results in terms of safety, even outperforming the closed-source ChatGPT-0301 model. The training process of LLaMAs is similar to that of the GPT model, with pre-training being performed using an optimized autoregressive transformer which is trained from a large corpus of unstructured data. LLaMA is then fine-tuned using various alignment techniques, and the authors also share a new technique called Ghost Attention, which aims to control dialogue flow over multiple turns. RLHF and PPO are other important techniques used to get the desired behaviour out of the model.

Many flavours of LLaMA have been trained. Most notably is Code LLaMA 2 (Rozière et al., 2023), a family of LLMs fine-tuned for code, scoring at 53% and 55% on the HumanEval and Mostly Basic Python Programming (MBPP) benchmarks, respectively. Vicuna-13B is another example of a LLaMA-based model, and is fine-tuned on user-shared conversations collected from ShareGPT[1], a website where users can upload ChatGPT conversations. At its release in March 2023, it achieved more than 90% quality of OpenAI's ChatGPT and Google's Bard, while also outperforming the original LLaMA model.

Mistral

Mistral 7B (Mistral AI, 2023) is another open-source LLM, and is claimed by its creators at Mistral AI to be "the most powerful language model for its size to date". Being a 7.3B parameter model, it is quite small compared to other models, such as the 1.76T parameter GPT-4 model. Mistral 7B outperforms the LLaMA 2 13B on all benchmarks, and approaches Code LLaMA 7B performance on code.

Orca 2

Orca 2 (Mitra et al., 2023)—developed at Microsoft—is an open-source LLM with exceptional step-by-step reasoning capabilities. It is based on LLaMA 2 and is fine-tuned on curated training data from a more capable teacher model, like GPT-4. Evaluation

[1] https://sharegpt.com/

results show that the Orca-2-13B model surpasses models of the same size, and that it is competitive with models 5-10x larger, exceeding the performance of LLaMA-2-Chat70B and performing comparably to ChatGPT on reasoning tasks (Mitra et al., 2023, pp. 11–12). It does have some limitations, most notably in terms of potential of misuse due to the lack of suitable safeguards like RLHF training (Mitra et al., 2023, p. 21).

2.2. Large Language Model Providers

The AI community called Hugging Face[2] is the most widely used hub for open-source and open-science machine learning. It is used by both individual community members and large commercial actors like Google Cloud and Meta AI as a way of contributing to the world open-source AI. Hugging Face's `transformers`[3] Python library provides an interface with their hub, which at the time of writing has 177k stars on GitHub.

OpenAI is one of the leading providers of LLMs, most prominently through their GPT series. While originally founded as an open-source company, their models are now closed-source and their GPT APIs are paid services. It should be noted, however, that they still are great contributors to open-source AI, having uploaded a range of speech detection and diffusion models to Hugging Face.

Microsoft Azure—the main partner of OpenAI—provides APIs for OpenAI's language models including the GPT-4, GPT-3.5-Turbo, and Embeddings model series. Unlike OpenAI's own APIs, Microsoft Azure also offers private networking, regional availability, and responsible AI content filtering.[4] Also, by using Microsoft Azure's API, LLM inputs and outputs are unavailable to OpenAI, which is not the case otherwise.[5]

Cleary (2023) did benchmarking of different LLMs on different providers. Important takeaways were that GPT-4 is about 6.3 times slower than GPT-3.5-Instruct, and that Microsoft Azure has far lower latency in most cases for inference on GPT models compared to OpenAI's own APIs. Such considerations are important when addressing usability of LLM-based applications, balancing accuracy against speed and cost.

Aforementioned Google Cloud is another large LLM provider. They provide a wide range of services, among these the Vertex AI platform.[6] Vertex AI provides API access to foundational models through their Model Garden platform which supports first-party, open-source, and third-party models. Vertex AI also provides experimental prompt design and fine-tuning services. Vertex AI is a paid service.

Anthropic[7] is another closed-source LLM provider that has grown rapidly in size since their launch in 2021. They focus on creating safer, steerable, and more reliable models. Their flagship model is Claude 2.1, which has a large context window of 200,000 tokens, enabling users to upload large technical documentation that the model can keep in its

[2] https://huggingface.co/
[3] https://github.com/huggingface/transformers
[4] https://learn.microsoft.com/en-us/azure/ai-services/openai/overview?source=recommendations
[5] https://learn.microsoft.com/en-us/legal/cognitive-services/openai/data-privacy
[6] https://cloud.google.com/vertex-ai/
[7] https://www.anthropic.com/

memory when generating responses. Anthropic lists complex reasoning, creativity, and coding as Claude's major strengths.

There are numerous open-source alternatives to these closed-source resources. Using free open-source alternatives where possible is a good option to reduce cost. Meta AI[8] is a big contributor to open-source AI, most famously through Pytorch—a popular deep learning framework—and their LLaMA models (see subsection 2.1.5). Eleuther AI[9] is an open-source centred company with aim to "increase transparency and reduce potential harms from emerging AI technologies". As such, they have released a range of trained LLMs along with the codebases used to train these. Perhaps most famous is their GPT-J model, which is an autoregressive model in the style of GPT-3, aimed at the English language.

2.3. Geospatial Standards

It is important for GIS applications of any kind to support modern geospatial standards. Subsection 2.3.1 will offer a brief overview of the standardization efforts on the international stage, while subsection 2.3.2 will similarly cover these efforts within the context of Norway.

2.3.1. International Standardization Work

International geospatial standardization involves the global collaboration and development of uniform standards and protocols for geospatial data and technologies. The OGC standards and the STAC API standard are examples of such work.

OGC Api Standard

The Open Geospatial Consortium (OGC) API standards serve as the glue in the field of GIS, paving the way for interoperability and data exchange between diverse systems. 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. 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). The OGC API standards function as modern replacements to older standards like WMS and WFS, and presents a more adaptable framework for spatial data operations, facilitating innovation 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

[8] https://ai.meta.com/
[9] https://www.eleuther.ai/

API infrastructure to their domain. STAC is closely related to OGC, and OGC board member Chris Holmes said the following in a blog post: "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). STAC API provides a powerful query language that allows users to search by various parameters like time, location, and keywords, making it widely applicable. The STAC community has also defined specifications in order remove the complexity associated with having to create unique pipelines when consuming different spatio-temporal collections. 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.).

2.3.2. Norwegian Standardization Work

Geospatial standardization work has been on the agenda of Norwegian governing powers for decades and have materialized in frameworks/collaborations like Geovekst and The National Spatial Data Infrastructure, as wells as the SOSI file format. This standardization work will be the topic of this section.

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, with the most important arenas being ISO/TC 211 and OGC (Mardal et al., 2015). SOSI is the adopted Norwegian standard for creating and delivering digital geographic data, and is administered by the Norwegian Mapping Authority (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 catalogue standard. However, there are few GIS systems that can read SOSI data directly, so data in SOSI format usually needs to be converted into more GIS-friendly data formats (Mæhlum and Rød, 2023).

Geovekst

Geovekst is a Norwegian initiative that aims for collaborative collection, distribution, management, and maintenance of geospatial information. It was established in 1992, and is a partnership between national, regional, and local government bodies, as well as several private companies.[10] The primary goal associated with Geovekst is to "collaborate

[10] https://www.kartverket.no/geodataarbeid/geovekst

to secure updated Geovekst data to help solve parts of the parties' societal missions" (The Norwegian Mapping Authority, 2023, p. 5). The objective of the collaboration is to ensure that geographical data is collected *once*, conforming to *one* standard, maintained in *one* place, and used by *many*. Responsibilities and costs are shared among the parties of the collaboration.

The most important contribution of Geovekst is Felles KartdataBase (FKB), a series of very detailed Norwegian mapping datasets, serving as rich resources for both public and private sectors. The datasets are obtained through a variety of data sources, including aerial photographs, laser scans, and manual mapping. Geovekst also provides airborne surveys in emergency situations, when the speed of surveying is important.[11]

The National Spatial Data Infrastructure

Established in 2005, the National Spatial Data Infrastructure (NSDI) is a more recent framework compared to Geovekst, and is more often referred to by its Norwegian name: *Norge Digitalt*. The NSDI 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). The NSDI infrastructure is the sum of common standards and rules, geographical data and services related to these, in addition to tools and deals. It aims to coordinate geospatial activities in Norway, making it easier to discover, access, and use spatial data. The framework is coordinated by the Norwegian Mapping Society (Norge Digitalt, 2023).

2.4. User Groups

Category	**Percentage**
Task oriented	23.1%
Informational	20.2%
Social	16.2%
Personal advice and self-improvement	13.1%

Table 2.1.: Distribution of query categories (Kumar, 2023)

There are several user groups that could take advantage of an LLM-based agent with extensive geographic and GIS knowledge. Queries to such an agent could span from simple retrieval questions like "How many people live in Trondheim?" and "What is Sweden's third largest city?", to more complicated questions that require problem-solving abilities and reasoning. While it is difficult to obtain good datasets of common queries, Kumar (2023)—creator of the chatbot app Pocket AI[12]—shared a dataset of ~13k user

[11] https://www.kartverket.no/geodataarbeid/geovekst/datafangst-i-krise
[12] https://github.com/varunon9/pocket-ai

queries from his app along with classifications of these. Salient categories are listed in Table 2.1. The main takeaway from these numbers is that the main motivation for use is productivity.

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)

Table 2.2.: Categories and frequency of ChatGPT usage (Skjuve et al., 2023, pp. 16–17)

This aligns with the results of Skjuve et al. (2023) from their questionnaire-based study performed in late January 2023, about two months after the release of ChatGPT. The goal with the study was to figure out why people use ChatGPT. They found that most participants (55%) are motivated by productivity, specifically applying it for routine tasks, information retrieval, text generation and writing support, and software development (Skjuve et al., 2023, pp. 17–21). Table 2.2 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 (18th December 2023), now that use patterns have become more established (Skjuve et al., 2023, p. 37).

The main reason people use conversational AI is clearly to increase productivity, whether in a professional, academic, or personal context. 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. The 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 GIS-related purposes.

One could imagine a range of potential user groups that could benefit from such an artificial, and spatially aware, companion. Some suggested user groups are presented in Figure 2.1. Perhaps the most obvious one is that of the GIS professionals. While an AI-based GIS agent more capable than the average GIS professional is currently far from becoming a reality, such an agent could help suggest strategies of solving a particular problem using the input data available and a good user prompt, or it could help solve mundane tasks in an automated way in order to allow GIS professionals to allocate more time to creative and complex tasks.

Closely related to GIS professionals are the city planners. Though often less knowledgeable in the field of GIS, they are increasingly dependent upon geospatial analyses in order to make informed decisions. Having an easy-to-use GIS agent ready at any

moment could prove both time- and cost-saving. The same goes for business analysts and people involved in academia. The *time* variable is especially important to the emergency services. At the impact of a natural disaster like a flood or forest fire, geospatial analysis could prove lifesaving. Having powerful geospatial knowledge even in the absence of a GIS professional is therefore important in order to focus resources to areas where the situation is most pressing.

An LLM-based agent with geospatial awareness could also be useful to the general public: one may want to find suitable biking routes or good hills for interval running, or to identify areas prone to flooding when buying a house. Most people do not possess the knowledge or time to perform such analyses themselves, so an automated AI-based agent could prove useful for such use cases.

2.5. Ethical and Privacy Concerns

Although an LLM-based GIS agent could prove powerful, there are some ethical pitfalls in terms of regulations and privacy concerns that need consideration. If such an agent is to make decision on behalf of humans it is important that one can hold someone accountable in the case that something goes wrong. If an AI-based agent is tasked to lead firefighters to the tactically optimal locations in order to quench a fire, but instead misleads them and traps them inside the fire, who is then to blame? Sparrow (2007) provides an interesting angle on such issues, though in his essay related the role of artificially intelligent robots in modern warfare.

Furthermore, it is vital to ensure that the AI agent is not utilized for immoral purposes, like finding the optimal location in which to dissolve poison into people's drinking water to maximize the number of people killed. Work has been done with newer LLMs to prevent them from producing dangerous, misinformed, or toxic text—this issue is widely discussed in the technical report of the latest GPT model (OpenAI, 2023, pp. 11–14)—but there have been cases where these issues haven't been considered well enough.[13]

The General Data Protection Regulation (GDPR) entered into applicability in the European Union on 25th of May 2018, and although not a member of the European Union, Norway incorporated the GDPR in July the same year due to its membership of the European Economic Area. The GDPR imposes stricter regulations about data privacy, meaning people have more control over their personal data, and that businesses get a level playing field in terms of what customer information is available (Datatilsynet, n.d.). With data privacy arguably being more relevant than ever, we must make certain that LLM-based GIS agents are unable to access and spread information of private or sensitive character, even if such information has become publicly available by accident—as was the case when the Norwegian Broadcasting Corporation (NRK) was able to (legally) obtain accuracy geolocations of central people in the Norwegian army from a commercial

[13]The LLaMA-based Alpaca model, developed at Stanford University, was taken offline after being shown to produce misinformation, toxic text: https://www.theregister.com/2023/03/21/stanford_ai_alpaca_taken_offline/

London-based company[14].

[14]Link to news article in Norwegian: https://www.nrk.no/norge/xl/norske-offiserer-og-soldater-avslort-av-mobilen-1.14890424

3. Related Work

The related works are divided into four sections, each being relevant to the project in different ways. Section 3.1 is the most obviously relevant section, discussing how LLMs have been employed to perform tasks in the geospatial realm. Section 3.2 delves into different planning strategies that could be utilized to make an autonomous GIS agent perform better and more reliably on complex tasks. Section 3.3 discusses Retrieval Augmented Generation (RAG), that is, how one can provide an autonomous LLM-based agent with external tooling and up-to-date information. Weng (2023) provides a good summary of techniques relevant to section 3.2 and section 3.3. Section 3.4 discusses common evaluation metrics and benchmarks for LLMs.

3.1. GIS with LLMs

A substantial body of work has been done in recent years to assess the geospatial knowledge of LLMs, and how they can be fine-tuned or embedded into frameworks to serve downstream tasks. Subsequent subsections will discuss this work, and subsection 3.1.1 will show how people currently use LLMs like ChatGPT for GIS-related purposes.

3.1.1. Taking the Temperature on GIS with LLMs on Social Media

The search term "LLM GIS" on Twitter/X shows various ways that people are using LLMs for GIS-related tasks. One user praises how ChatGPT is utilized to "extract and categorize data from unstructured text", sharing a video recording from an ESRI conference.[1] Twitter user Zeke Hausfather shares the discovery that "GPT4 now supports [of] processing netCDF files and other geospatial data, as well as some pretty amazing visualization".[2] Arpit Gupta shares a summary of a paper on generative regulatory measurement, where he explains how they have utilized LLMs to decode and interpret status updates and administrative documents, including mapping zoning and housing regulations for the suburbs of Chicago.[3] Yu Zhao speculates on the effectiveness of smaller LLMs fine-tuned on domain-specific knowledge for GIS or remote sensing,[4] an interest other users share.[5,6]

[1] https://twitter.com/mildthing99/status/1658507921234296833
[2] https://twitter.com/hausfath/status/1704199024675549485
[3] https://twitter.com/arpitrage/status/1723033894801309893
[4] https://twitter.com/zhaoyutim/status/1651233975946321920
[5] https://twitter.com/zhaoyutim/status/1651233975946321920
[6] https://twitter.com/DougButdorf/status/1670938318979121152

Swapping out "LLM" with "ChatGPT" gave more results. One user shows how using ChatGPT with tabular geographical data can increase productivity.[7] Other users show how they use ChatGPT for entertainment or as an educational tool in a GIS context.[8,9,10,11,12] This appears to be the primary method by which people use ChatGPT, and it seems to offer mostly adequate responses. Another user highlights ChatGPT's built-in geographical context, using it to get GeoJSON polygons for a specified area directly.[13]

On YouTube, the search term "ChatGPT GIS" yields a range of relevant responses. Several videos display how ChatGPT can be used to create Python code for GIS-related purposes. Examples were found of users highlighting ChatGPT's abilities to generate Python code to manipulate geospatial files, perform analysis, and visualize, using libraries like GeoPandas and Folium.[14,15,16] Another user shows how uploading geospatial files into ChatGPT using the Code Interpreter can be an efficient workflow.[17] Some users demonstrate the QChatGPT[18] plugin to QGIS, which is a plugin integration between QGIS and the OpenAI API. QChatGPT does not seem to have any context of the current QGIS project the user is working on, but appears to have sparked some excitement among certain users, seeing how LLMs can assist them in their daily work as GIS professionals.[19,20,21] One user shows an application with ChatGPT integration that can generate SQL code and visualize geospatial data.[22] A recurring user shows how one can use LangChain and its SQL database plugins to "unlock ChatGPT's potential".[23]

The "FME Channel" released a recording of a webinar to YouTube where they show how GPT-3 is being used in FME Data Integration Workflows.[24] They highlight how the OpenAI API allows for easy and automated no-code API-to-API workflows. On the FME Community website, an article writes about the `OpenAICompletionsConnector` and `OpenAIImageGenerator` transformers in FME.[25] They list use cases such as running data through the AI for analysis, generation of reports and summaries, generation of scripts or SQL for use in a data integration workflow, and automatic generation of images

[7] https://twitter.com/BooneLovesVideo/status/1617479222724857856
[8] https://twitter.com/briankingery87/status/1631365717269307394
[9] https://twitter.com/burdGIS/status/1614630141858316288
[10] https://twitter.com/_jsolly/status/1652867118797590528
[11] https://twitter.com/wanjohikibui/status/1628282272548806657
[12] https://twitter.com/GeoWithJustin/status/1641155652759199744
[13] https://twitter.com/at_dot_Py/status/1649985754800730112/
[14] https://www.youtube.com/watch?v=QDf-zc81NSE&t=1707s&ab_channel=GeoDeltaLabs
[15] https://www.youtube.com/watch?v=iNHQgLw7qZc&ab_channel=GeoDeltaLabs
[16] https://www.youtube.com/watch?v=BK2IzZZZC-k&ab_channel=MattForrest
[17] https://www.youtube.com/watch?v=dgzWLBYswh0&ab_channel=MiningGeologist
[18] https://plugins.qgis.org/plugins/QChatGPT/
[19] https://www.youtube.com/watch?v=zUZs4GsDk6I&ab_channel=GISWorld
[20] https://www.youtube.com/watch?v=eEkVTUS8Qtc&ab_channel=HansvanderKwast
[21] https://www.youtube.com/watch?v=Tc-hHaDqoxY&ab_channel=DEVICKSGEOSPATIALCO.
[22] https://www.youtube.com/watch?v=gaA46aaWDuc&ab_channel=GeospatialWorld
[23] https://www.youtube.com/watch?v=FoGm7d0paIo&t=1190s&ab_channel=MattForrest
[24] https://www.youtube.com/watch?v=94ZDhgW8yMY&ab_channel=FMEChannel
[25] https://community.safe.com/s/article/Tutorial-Getting-Started-with-OpenAI-in-FME

based on a dynamic input.

3.1.2. Geospatial Context in LLMs

Scherrer and Ljubešić (2020) were able to show that BERT can be fine-tuned to accurately predict geolocations from textual input, by winning a shared task on predicting geolocations from Twitter/Jodel messages in a workshop in 2020 (Gaman et al., 2020). By converting the task into a double regression problem, they were able to accurately predict latitude/longitude pairs from the output `[CLS]` representation of BERT models. For a subtask on a Swiss Jodel dataset, they were able to achieve a median distance of 15.72 km from the ground truth, showing that LLMs can be trained to correlate lingual features and geolocations.

Roberts et al. (2023) investigated the extent of GPT-4's geospatial awareness through a set of case studies with increasing difficulty, starting with general factual tasks and finishing with complex questions such as generating country outlines and travel networks. The authors found that GPT-4 is "skillful at solving a variety of application-centric tasks", almost having the ability to "see", despite being a language model and therefore only able to interface with the world through sequenced, textual input. Examples include its ability to serve as a travel assistant in providing itinerary suggestions for a trip when provided with requirements, and its ability to provide generally correct start and end locations of bird migration paths. While it quickly became obvious that a lot of geospatial context have been embedded within the model during the vast pre-training, the question of whether this is memorization or reasoning is a central one. The authors suggest that the variability of tasks in their experiments deems it unlikely that it is all memorization, but they say that some things appear to be memorized.

Mooney et al. (2023) examined the performance of ChatGPT in a Geographic Information System (GIS) exam, aiming to assess its ability to grasp various geospatial concepts, highlighting its capabilities and limitations. Experiments were conducted on GPT-3.5 and GPT-4, which delivered performances equivalent to grades of D and B+, respectively. Additional experiments were conducted for more specialized areas of GIS, including True/False questions about spatial analysis, and simple tasks in applied GIS workflows. Experiments on the latter showed that ChatGPT-4 was able to correctly answer a relatively complex GIS task involving seven different datasets, requiring seven steps in order to obtain a perfect score. Generally, ChatGPT-4 outperformed ChatGPT-3.5 in all tasks. While clearly powerful, the authors highlight a range of limitations, among which the multimodal nature of GIS, which would hinder a straightforward application of existing models.

Unlu (2023) discussed the importance of enabling LLMs to recognize and interpret geospatial data, and how OpenStreetMap (OSM) can play an important role in offering LLMs linguistic access to vast cartographic datasets. He exemplifies this claim through a proof of concept in which he performs small-scale fine-tuning on an LLM with 1B parameters, using an artificial supervised datasets curated by the more capable ChatGPT 3.5-turbo model, which functions as a teacher model, generating prompt-answer pairs for given preprompts. The fine-tuned model displays promising abilities in answer questions

about a location's attributes, allowing the user to inquire about things like tourist appeal and potential profitability of businesses in the vicinity of the given location. Unlu emphasizes the method's strengths when applied for small datasets and using minimal computational settings. The study also investigated the idea of using embeddings of the curated prepromts. Experimenting with average GLOVE embeddings, Unlu showed that the latent structure of verbal descriptions of OSM data can yield insightful patterns. This, he argues, can prove useful when creating Retrieval Augmented Generation (RAG) applications aimed at allowing users to retrieve geospatial information in a prompt-based manner.

3.1.3. Autonomous GIS

Z. 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.". They provide a "divide-and-conquer"-based method to address some of these goals. Furthermore, they propose a simple trial-and-error approach to address the self-verifying goal. They also highlight the need for 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 Z. Li and Ning (2023) provide—called LLM-Geo—is able to produce good solutions to 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) use the LangChain framework in order to combine different GIS tools in a sequence to solve various sub-goals, focusing on using the semantic understanding and reasoning abilities of LLMs 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 their names and descriptions. These descriptions contain 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.

Qi et al. (2023) discuss how LLMs can be used in spatial data management, facilitating a system that can learn from both structured and unstructured data, the latter of which is possibly the greatest strength of modern LLMs. They highlight the opportunity that LLMs provide in reducing the barrier to information retrieval for the general public, and discuss how these strengths can be used in spatial data management by leveraging a spatial database system trained from both structured and unstructured data. This could potentially allow for seamless access to spatial knowledge, also for those with little or no expertise in querying a spatial database. Qi et al. envisage to use *Machine learning models as a Spatial DataBase* (MaaSDB), that—when trained on structured and unstructured spatial data—can *generate query answers directly* instead of retrieving data from tables, the latter of which has been the most common way of using machine learning in database query processing. Based on preliminary studies they present an LLM-based system of query analysers, query plan generators, and query result generators to handle natural

language user queries. They propose a Generative Adverserial Network (GAN)-based model to generate tabular data, anticipating that such a model could remember the key characteristics of the data. A GAN will have a *generator G* that will produce a record, and a *discriminator D* that will classify whether the generated record resembles a real record. The results of this approach are promising, and further prompt-based tests performed on ChatGPT demonstrate its potential to learn spatial knowledge and answer queries. While potentially powerful, they highlight a range of challenges of implementing their proposed system, such as hallucination, the limited availability of structured spatial data, generalizability issues, and the problem of updating the trained models when the underlying data changes.

3.2. Planning Strategies

Large Language Models have shown great abilities in problem-solving and decision-making tasks, but generally struggle as they are presented with larger and more complex tasks. Also, seeing as they are pure stochastic machines, the output is seldom reproducible. While the temperature parameter of the GPT models help serve as a control mechanism for this randomness, it does not guarantee fully predictable text generation. These issues have led people into investigating *prompt engineering* and *planning* techniques to help LLMs form plans when faced with large and complex tasks, trying to guide them into producing the desired response.

The *Chain of Thought* strategy (Wei et al., 2023) is aimed at complex reasoning in Large Language Models and has showed that reasoning can emerge naturally from sufficiently large LLMs. *Chain-of-Though prompting* entails the inclusion of examples of chain of thought sequences (examples of how one might reason about a given problem in order to get to the answer) into the prompt. The exemplars are categorized into the types of tasks they aim to solve. This, along with instructing the model to think "step by step", achieved a new state-of-the-art accuracy on the GSM8K benchmark of math word problems in early 2023.

The *Tree of Thoughts* strategy (Yao et al., 2023) is a more recent planning strategy aimed at problem-solving with Large Language Models, and addresses a common limitation of *vanilla* LLM problem-solving, which often lacks the ability to explore strategically. Generalizing over *Chain of Thought*, *Tree of Thoughts* allows the LLM to consider multiple different reasoning paths and to perform self-evaluation to decide the next course of action. *Tree of Thoughts* can be used with different search algorithms. The authors discuss breadth-first search and depth-first search, and leave more advanced algorithms for future work. Using the *Tree of Thoughs* planning strategy proved very effective on certain tasks that are near impossible for the state-of-the-art LLM of GPT-4, particularly in the mathematical reasoning challenge called "Game of 24".

Zhou et al. (2023) introduce 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, demonstrating state-of-the-art performance on decision-making

tasks in a range of diverse domains. LATS performs a sequence of operations in succession until the task at hand is solved. These operations are *selection, expansion, evaluation, simulation, backpropagation, and reflection*. By employing Monte Carlo Tree Search they enable the LLM-based system to select among n sampled options while still exploring other promising alternatives, using a heuristic to rank alternatives. Through a shared space of thoughts and actions, the framework supports both reasoning and decision-making tasks. Observation and self-reflection abilities enables LATS to use external feedback, which proved valuable when testing the framework on different benchmarks, some of which are discussed in subsection 3.4.2.

3.3. Retrieval Augmented Generation and Frameworks

Retrieval Augmented Generation (RAG) is tightly interwoven with explainable AI, being a framework for retrieving facts from an external knowledge base to give LLM-based agents access to accurate and 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 a response that is completely wrong, but formulated in a very convincing manner. While progress is being made with newer models, even GPT-4, which is considered state-of-the-art, gives an incorrect answer about 1 out of 5 times, and the accuracy is even worse for certain categories of queries (for instance, 'code' and 'business') (OpenAI, 2023, p. 10). Retrieval Augmented Generation can help mitigate such problems.

Lewis et al. (2020) showed that a RAG model with access to a non-parametric memory in the form of a dense vector index of Wikipedia, would generate more specific and factual responses compared to state-of-the-art parametric-only sequence-to-sequence models at the time of publishing their paper. Their model architecture is a combination of a pre-trained retriever and a pre-trained sequence-to-sequence generative model, which is fine-tuned end-to-end (Lewis et al., 2020, p. 2). Their approach obtained state-of-the-art results on open-domain question answering (Lewis et al., 2020, pp. 5–6).

Shi et al. (2023) show that a simple RAG architecture provides significant improvement over state-of-the-art parametric-only LLMs like GPT-3. Their REPLUG framework works by retrieving documents and prepending these to a "black-box" LLM. They also propose a training scheme to further improve the retrieval model with supervision signals from the black-box LLM. Training is done with an objective that prefers documents that improve the perplexity (explained in subsection 3.4.1) of the model. This approach shows promising results relative to the original black-box model (Shi et al., 2023, pp. 5–6).

3.3.1. 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-augmented generation, autonomous agent creation, and memory-related tasks.

Nascimento et al. (2023) experimented by integrating ChatGPT and LangChain for Natural Language Interfaces for Databases (NLIDBs), that is, allowing the querier of a database to use natural language queries such as "Give me locations of all churches in Trondheim along with a short description" instead of SQL queries. They saw promising results when using LangChain's `SQLDatabaseChain` tool, which inspects database schemas, tables, and joins in the database one provides it with. Doing so also helps mitigate issues with exceeding the ChatGPT token limit, which can be a problem if one simply passes entire table schemas as prefaces to the prompt itself. However, while the method was able to answer 13/27 test queries correct, using keyword search tools along with ChatGPT proved significantly more applicable, answering 22 correctly and only getting 5 wrong.

3.3.2. AutoGPT

AutoGPT (Richard, 2023) is a Python framework for autonomous AI agent development that will try to split a task into subtasks and use the internet and other tools in an automatic loop to solve the tasks/subtasks. AutoGPT comes with ready-to-go code templates for various purposes, benchmarks for agent performance measurements, and UI and CLI tools to control and monitor agents. The AutoGPT project adopts the *Agent Protocol* (*Agent Protocol* n.d.), which is an OpenAPI specification v3-based protocol that provides a common interface for communicating with agents. This ensures compatibility with future applications, and is currently used for communication with the UI and CLI tools.

Firat and Kuleli (2023) performed an exploratory study to map different use cases and experiences of AutoGPT users. They found that content creation—e.g., making a podcast outline—is a common use case for AutoGPT-powered applications. Other applications include data summarization and information organization. The authors highlight limitations like token limit and inefficiency. AutoGPT is also known to behave unreliable at times, and a common complaint is that it gets stuck in "reasoning loops".[26,27]

3.3.3. Function Calling and the Assistants API (OpenAI)

Function calling[28] from OpenAI enables the connection of LLMs to external tools by allowing the developer to provide the LLM with functions descriptions. These function descriptions will be used by the LLM to select appropriate functions and call these with the arguments it finds appropriate. This way we can convert a natural language query such as "Show me Trondheim in a map" into a function call to `plot_map(latitude, longitude, zoom_level)` with fitting arguments.

OpenAI's Assistants API[29] allows developers to build AI assistants supporting three types of tools: Code Interpreter, Retrieval, and Function calling. This enables agent-like

[26] https://github.com/Significant-Gravitas/AutoGPT/discussions/1939
[27] https://github.com/Significant-Gravitas/AutoGPT/issues/1994
[28] https://platform.openai.com/docs/guides/function-calling
[29] https://platform.openai.com/docs/assistants/overview

behaviour, and it also supports file uploads. It can be thought of as OpenAI's answer to LangChain.

3.3.4. AutoGen and Microsoft Semantic Kernel

AutoGen and Microsoft Semantic Kernel were both created at Microsoft and are aimed at creating autonomous LLM-based agents. AutoGen (Wu et al., 2023) is a generic framework that allows for multi-agent applications in which agents can converse with each other. The authors demonstrate the effectiveness of the approach in domains including mathematics, coding, and online decision-making. They highlight improved performance, reduced development code, and decreased manual burden for existing applications as the main benefits. AutoGen also allows for limiting of fixed back-and-forth interactions between the AI agent and the human user by enabling this inter-agent communication.

Microsoft Semantic Kernel[30] is an SDK that functions as the brain of an autonomous agent and provides connectors to models and memory, and connects the agent to the outside world using triggers and actions. Maeda (2023) talks about how Semantic Kernel can be used to augment the abilities of AutoGen agents by providing it with *hooks* into the real world. These *hooks* can be native functions written by the developer, or existing OpenAI/Semantic Kernel plugins.

3.4. Evaluation and Benchmarking of LLMs

Subsection 3.4.1 will address common evaluation metrics used during model development, while subsection 3.4.2 will present some common benchmarks used to compare the performance of different LLMs.

3.4.1. Evaluation Metrics

Having objective ways of evaluating the performance of a textual response is as important as it is challenging. Such evaluations are often subjective in nature, and it is not immediately obvious how automate evaluation. This section presents common approaches for different objectives, and serves as inspiration for how an evaluation metric can be adapted for GIS-related purposes.

Human Evaluation

Human evaluation—though an obvious evaluation metric—can be powerful. Human evaluators can manually score generated text based on a range of criteria, including relevance, fluency, coherence, and overall impression (Ceylan, 2023). Human evaluation can be expensive and time-consuming, and researchers have therefore developed mathematical formulas for evaluation.

[30] https://github.com/microsoft/semantic-kernel

Perplexity

Perplexity is an evaluation metric suitable for autoregressive models, measuring the degree of uncertainty in predicting the next word in a sequence, based on the preceding words. It is essentially a way of evaluating a model's ability to predict uniformly among the tokens available in the corpus it is trained upon. This is done by calculating the negative average log-likelihood

$$\mathrm{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}\log p_\theta(x_i|x_{<i})\right\} \tag{3.1}$$

where $X = (x_0, x_1, \ldots, x_t)$ is the tokenized input sequence and $p_\theta(x_i|x_{<i})$ is the log-likelihood of token x_i given the preceding tokens $x_{<i}$. A lower score indicates better performance. Perplexity is normally calculated using a sliding window strategy, where a fixed number k preceding tokens $(x_{i-k-1}, x_{i-k}, \ldots, x_{i-1})$ are used to calculate the perplexity for token x_i (Hugging Face, n.d.).

BiLingual Evaluation Understudy (BLEU)

BiLingual Evaluation Understudy (BLEU) provides a quick, inexpensive, and language-independent method of automatic machine translation evaluation, allowing researchers to rapidly home in on effective modelling ideas (Papineni et al., 2002). The BLEU formula takes the geometric mean of the corpus' modified precision score and then multiplies it by an exponential brevity penalty factor. (3.2) shows the result when taking the log of the function, which makes the ranking behaviour more apparent (Papineni et al., 2002, p. 5).

$$\log \mathrm{BLEU} = \min\left(1 - \frac{r}{c}, 0\right) + \sum_{n=1}^{N} w_n \log p_n \tag{3.2}$$

Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric, introduced by Lin (2004), aims to automatically determine the quality of a summary by comparing it to ground truth summaries produced by humans. (3.3) shows the ROUGE-N formula, which is the n-gram recall between a candidate summary and a set of the aforementioned ground truth summaries.

$$\text{ROUGE-N} = \frac{\sum_{S\in\{\text{ReferenceSummaries}\}}\sum_{\text{gram}_n\in S}\text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S\in\{\text{ReferenceSummaries}\}}\sum_{\text{gram}_n\in S}\text{Count}(\text{gram}_n)} \tag{3.3}$$

Diversity

Diversity metrics aim to measure the variety and uniqueness of generated sequences. J. Li et al. (2016) proposed an objective function called Maximum Mutual Information (MMI), which seeks to guide sequence-to-sequence models into producing more diverse,

interesting, and appropriate responses, as opposed to safe and commonplace ones. The parameters of MMI are chosen in order to maximize mutual information between the source sequence S and the target sequence T:

$$\hat{T} = \underset{T}{\operatorname{argmax}}\{(1-\lambda)\log p(T|S) + \lambda \log p(S|T)\} \tag{3.4}$$

where λ serves as a weighting parameter. As the paper is from 2016, the authors only discuss the $p(Y|X)$ function in relation to the Long Short-Term Memory (LSTM) algorithm, but it can be applicable for contemporary language models as well.

Stasaski and Hearst (2022) propose three metrics which leverage the predictions of a Natural Language Inference (NLI) model, that is, a model which seeks to determine if one sentence entails, contradicts, or is neutral toward a second sentence (Stasaski and Hearst, 2022, p. 1). (3.5) shows the *Baseline NLI Diversity* metric

$$\text{Baseline NLI Diversity} = \sum_{u_i, u_j \in u_1, \ldots, u_n} NLI_{score}(NLI_{pred}(NLI(u_i, u_j))) \tag{3.5}$$

where NLI_{score} is 1, 0, or -1 if the sentence is deemed contradictory, neutral, or entails the other sentence, respectively. The two other metrics—the *Neutral NLI Diversity* and *Confidence NLI Diversity*—differ only in how they define the NLI_{score}. Results from experiments show that using these can produce more diverse sets of responses, and that they can be used to investigate a model's ability to produce diverse responses (Stasaski and Hearst, 2022, p. 9).

3.4.2. Benchmarks for LLMs

Benchmarks for LLMs are standardized tests that aim to highlight the strengths and weaknesses of different models, serving as a non-biased way of comparing them. Benchmarks have been developed to assess a model's level of performance on different tasks, such as language understanding, general knowledge, arithmetic, and code generation. This section will introduce some common benchmarks.

HumanEval is a dataset of handwritten problems used to measure functional correctness for synthesizing programs from docstrings (Chen et al., 2021, pp. 2–4). Code generation is one of the most common use cases for LLMs, and HumanEval is therefore arguably one of the more important benchmarks out there.

Multitask Language Understanding (MMLU)—first introduced by Hendrycks et al. (2021)—is a way of testing a Large Language Model's multitask accuracy, covering 57 tasks including mathematics, computer science, and others. MMLU is commonly used to highlight the general knowledge that is embedded within the model.

The BIG-Bench-Hard benchmark (Suzgun et al., 2022) for LLMs is a suite of 23 problems where language models previously were unable to exceed average human performance. Many of the BIG-Bench-Hard tasks are characterized by requiring the rater to use multi-step reasoning, which has traditionally been hard for language models to apply.

HellaSwag (Zellers et al., 2019) is a benchmark designed to measure an LLM's ability to "finish your sentence". By developing a lengthy and complex dataset to see where the LLM starts producing "ridiculous" responses, the HellaSwag benchmark provides a way of testing a model's common-sense inference abilities.

API-Bank (M. Li et al., 2023) is a benchmark designed to evaluate an LLM's ability to use external tools (web APIs, etc.). Through interviews, the author highlights two main requirements for a tool-augmented LLMs (M. Li et al., 2023, p. 2): (1) *Few vs. Many APIs in API Pool.* With only one or a couple APIs in the API pool, one can possibly send entire API schemas with the prompt, simplifying request parameterization and response parsing. This becomes difficult as the number of APIs increases, because the token limit becomes a limiting factor. When this is the case, the LLM needs to reason about which APIs are relevant or not. (2) *Single vs. Several API calls per Turn.* Based on the user's preferences, one might want the LLM to perform several API requests at once, or one might want to gradually guide it through several steps.

4. Testing and Results

Three tests were conducted in this specialization project. Section 4.1 will explain how the tests were conducted, and section 4.2 will present the results.

4.1. Method: Test Planning and Setup

The tests were conducted in order to answer the research questions listed in section 1.2. As section 3.1 shows, there has been a substantial body of work on demonstrating the geospatial abilities of LLMs like ChatGPT, and how these can be used to create larger frameworks for GIS purposes. However, the literature review did not reveal any specific discussions on how LLMs, like ChatGPT, handle different geospatial file formats, such as GML or shapefiles, or manage various data access channels. The tests are designed to address these gaps, with Test 1 focusing on RQ1 and ChatGPT's abilities of performing geospatial analysis using different file formats, while Test 2 and 3 focus RQ2 and RQ3 and on its ability to access external web APIs.

4.1.1. Test 1: Testing ChatGPT's Ability to Perform Geospatial Data Analysis

The approach used in Test 1 is inspired by the work of Roberts et al. (2023) (see section 3.1), who conducted experiments with increasing difficult on GPT-4 to characterize what it knows about the geographical world, highlighting both capabilities and limitations. Roberts et al. focused on GPT-4's general geospatial awareness, and were not concerned with GIS-related tasks. The reference to Roberts et al. (2023) will therefore be made when highlighting the somewhat surprising geospatial awareness abilities of GPT-4, and the focus of Test 1 will instead be on displaying its potential for use in the world of GIS. This will be achieved by constructing various tests that aim to reflect GPT-4's GIS knowledge.

Test 1 will utilize the Elveg 2.0 dataset (The Norwegian Mapping Authority, 2019), along with cadastral data (including building data, etc.). In order to assess ChatGPT's ability to read and understand different data formats, the data will be provided in SOSI, GML, GeoJSON, and shapefile formats. Datasets in the first two formats were downloaded from https://geonorge.no, while the GeoJSON and shapefile datasets were created from the GML dataset using QGIS. The Elveg 2.0 dataset contains a range of different layers for different types of geometries and use cases. In order to simplify the test, only the layer named "Fartsgrense" (English: "Speed limit") was used from 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 were added when needed, in order help the model understand the questions, or when it stopped and asked for permission to go forth with analysis. All conversations are saved in the project's GitHub repository.[1]

4.1.2. Test 2: Comparing File Upload and API Calling in ChatGPT-4

Another important thing to test is the issue of providing ChatGPT with relevant files on which it can perform analyses. ChatGPT Plus users will have access a range of advanced features, including web browsing with Bing, Dall-E Image Generation, and the Code Interpreter (or Advanced Data Analysis). The latter of these allows the user to manually upload files into the chat instance and have it perform advanced analyses on the contents of these, which is what was used to upload the datasets in Test 1. This is of course very powerful, but having to manually upload files poses some limitations. A more flexible system should be capable of accessing web APIs in real time.

A dataset delineating the border of Drammen municipality was utilized to test ChatGPT's capability to analyze manually uploaded data, in comparison to providing it with a URL to an API endpoint. The data conforms to the GeoJSON standard—which is supported by OGC API - Features—and contains a FeatureCollection object with a single Feature, namely the border. When the file/API endpoint had been provided, ChatGPT was simply asked to provide a visual presentation of its contents.

The dataset is available through an OGC API that was created by Norkart's Alexander Salveson Nossum for testing purposes.[2] The API was deployed using `pygeoapi`,[3] a Python library that implements the OGC API standards.

4.1.3. Test 3: Using ChatGPT and LangChain to Make API Calls

A third and final test was conducted to see if there are other, more programmatic ways of performing API calls using LLMs. The LangChain framework (Chase, 2022) was used to

[1] https://github.com/oskarhlm/prosjektoppgave/tree/main/documents/ChatGPT_conversations
[2] https://alenos-tester001.azurewebsites.net/
[3] https://pygeoapi.io/

	SOSI	GML	GeoJSON	Shapefile
Summary	No	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 4.1.: Overview of the ability of ChatGPT's Code Interpreter to handle various geospatial data formats

create OpenAI functions that can be called using OpenAI's Function calling mechanism. The goal of the test was to produce the same plot as requested in Test 2 using the same API endpoint.

A function with signature `make_http_request(url: str, save_file_name: str)` was made to fetch the data from the API and save it to a temporary file on the computer from which the code is executed. Another function, `plot_geojson(geojson_path: str)`, was created to load a GeoJSON file from the temporary folder and plot the contents using the Matplotlib library. The hope is that by using LangChain's `AgentExecutor`, which enables reasoning and chaining responses from an LLM, in conjunction with the Function calling capabilities of the OpenAI GPT APIs, should make it possible for the agent to call these functions in the right order and with the correct arguments. The `gpt-4-1106-preview` model was used for this test.

The test is only meant to be an initial test to demonstrate how LLMs can be adopted to access external tools, thereby supporting RQ3. The current setup is by no means scalable, but is rather meant to demonstrate a technique that can be utilized when developing larger and more capable systems.

4.2. Results from Tests

4.2.1. Results from Test 1

As Table 4.1 shows, 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 to GeoJSON using GDAL's `ogr2ogr` program. For this reason, no further testing was done using the SOSI data format.

Furthermore, 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 instead—which can handle GML data—it managed to plot the file contents and calculate the centroid of the building points. The *Buffer + Intersect* task "was interrupted due to its time-consuming nature", and ChatGPT failed to make an attempt at solving the planning task due to its inability

to flexibly analyze the GML files.

With the GeoJSON data, ChatGPT had difficulties reading the files and could not provide a good summary consistently. It *was* however able to plot the data, but that had to be done in separate responses for each of the two datasets. It was also able to identify the high-speed roads, but could not determine which buildings were located within a 50-meter buffer around these roads. 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. While this did not extract a subset of the building points—which would have been the desired output—it had some minor value in terms of visualization (see Figure 4.1). Although this result is not particularly useful, it is closer to an acceptable response than the outcomes from the attempts made at the other file formats.

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 datasets, colouring the roads differently by their speed limits and the buildings by their building type (see Figure 4.2). While it did manage to filter roads on speed limits, it could not calculate the mean location of the buildings. The *Buffer + Intersect* and planning tasks again proved too complex.

4.2.2. Results from Test 2

Figure 4.3a and Figure 4.3b show the results of providing ChatGPT-4 with the GeoJSON file for the outline of Drammen municipality using ChatGPT's file upload functionality and supplying it with a URL to an API endpoint, respectively. While the Code Interpreter handled the direct file upload with ease, it struggled when provided with the URL to the corresponding web API. When provided with the URL, its first response was that "there was a problem with establishing a connection to the website", after trying to process the request using the Code Interpreter. When guided (twice) to try accessing the URL using its web browsing capabilities instead, it was eventually able to read the data. In the subsequent prompt it was asked to provide a visual representation, but failed to do so successfully as Figure 4.3b shows. The reason for this was its decision to "truncate the dataset for brevity, using a subset of the full coordinate list" (see Listing 4.1).

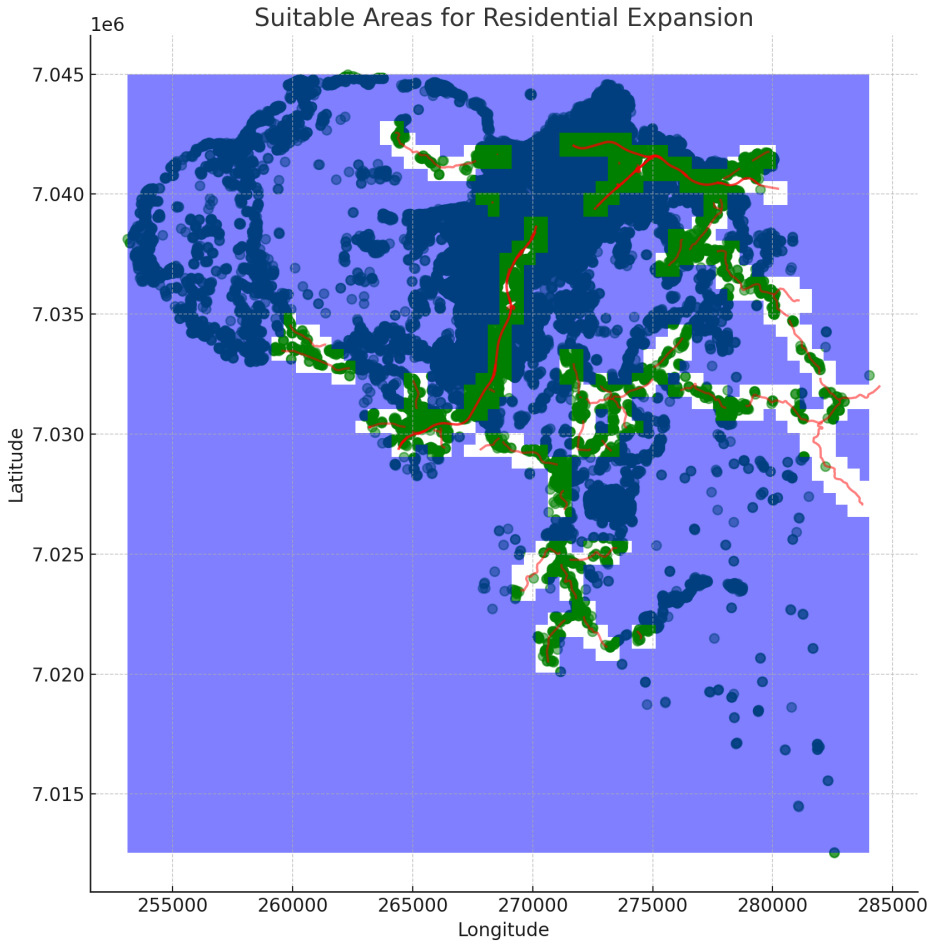


Figure 4.1.: ChatGPT's response when asked to "Find the area best suited for expansion to accommodate residential buildings", using provided GeoJSON datasets. Potentially suitable areas for residential expansion are depicted in blue.

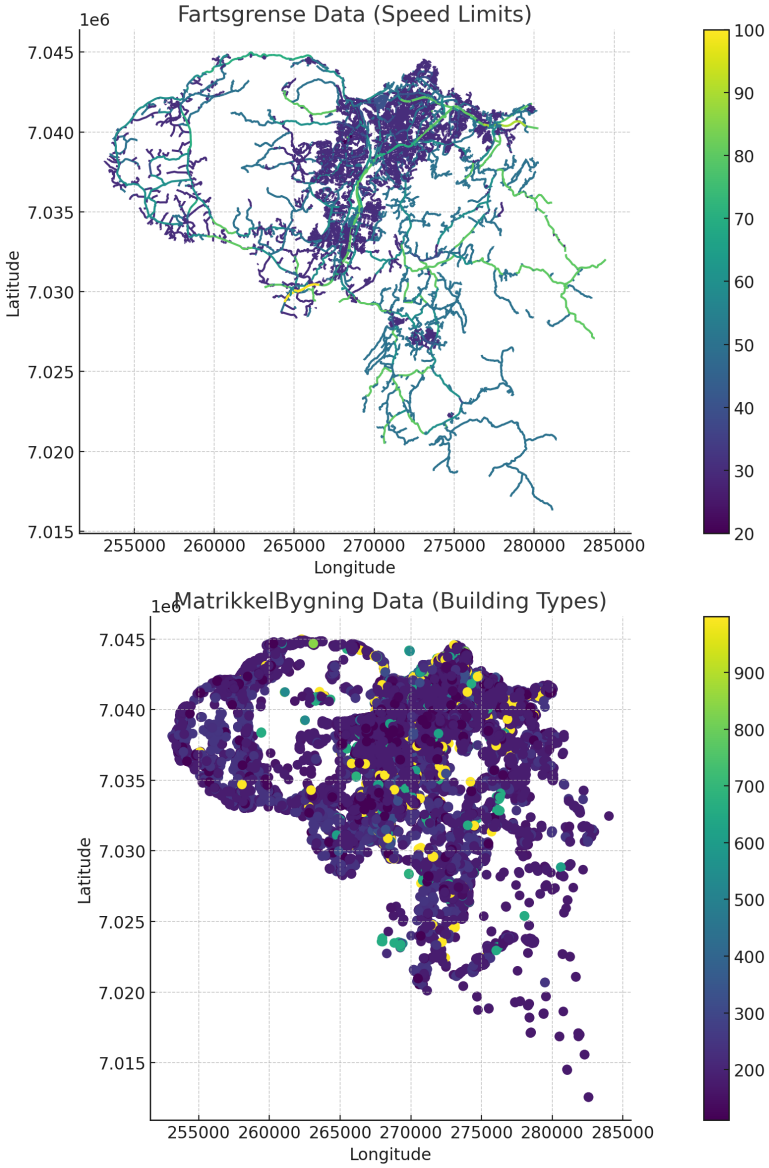
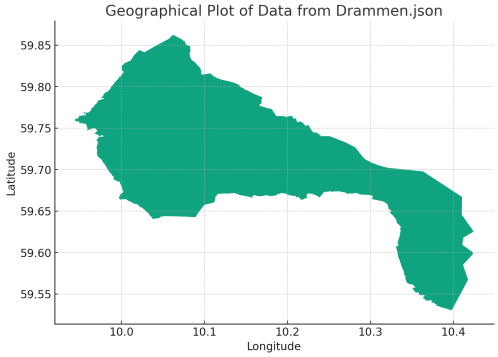
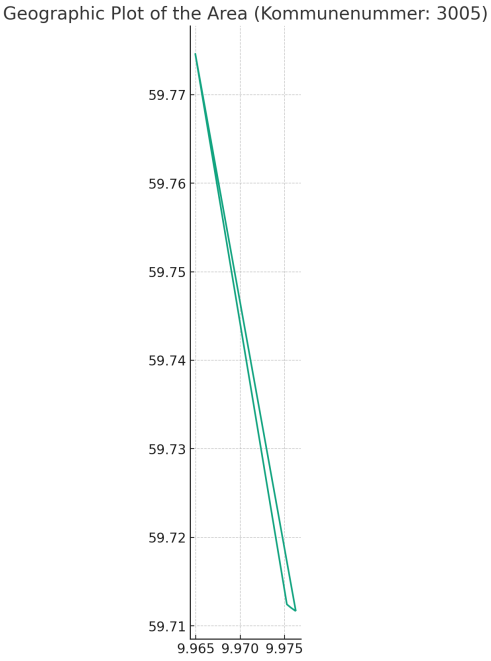


Figure 4.2.: Visual representations of shapefile datasets using ChatGPT's Code Interpreter



(a) File upload



(b) Providing URL

Figure 4.3.: Comparison of the resulting outlines/polygons of Drammen when using file upload (a) and providing an URL to an API endpoint (b) with ChatGPT-4

Listing 4.1: ChatGPT code that truncates coordinates

```
# ...

# Extracted coordinates from the JSON data
coordinates = [
    [[9.976278541184014, 59.71166107645171],
       [9.975715936016496, 59.71206324390201],
       [9.975270250797282, 59.71243147807226],
    # ... Truncated for brevity, using a subset of the full
       coordinate list
    [9.964929990238785, 59.774609947672644]]
]

# ...
```

4.2.3. Results from Test 3

The setup described in subsection 4.1.3, which allows the `AgentExecutor` from LangChain to autonomously make arbitrary calls to the `gpt-4-1106-preview` model, successfully called the two functions in the correct order and with the appropriate arguments. It first decided to call `make_http_request(url: str, save_file_name: str)` with the provided URL to the API endpoint and "kommuner.geojson" as function arguments. As expected, the function call resulted in a GeoJSON file called "kommuner.geojson" being stored at `/tmp/kommuner.geojson`. The file name is likely derived from the URL it was provided with, as it did not have any prior knowledge about the file contents at the time of saving. The `AgentExecutor`—powered by GPT-4—then correctly decided to call `plot_geojson(geojson_path: str)` using "kommuner.geojson" as the function argument. This resulted in Figure 4.4, which correctly shows the outline of Drammen municipality. Since this last function returns the string "File plotted successfully" after having plotted the file contents, the `AgentExecutor` correctly decided to terminate the chain. The notebook used for the test is available on the project GitHub repository.[4]

[4] https://github.com/oskarhlm/prosjektoppgave/blob/main/src/python/examples/drammen_ogc_test/drammen_ogc_test.ipynb

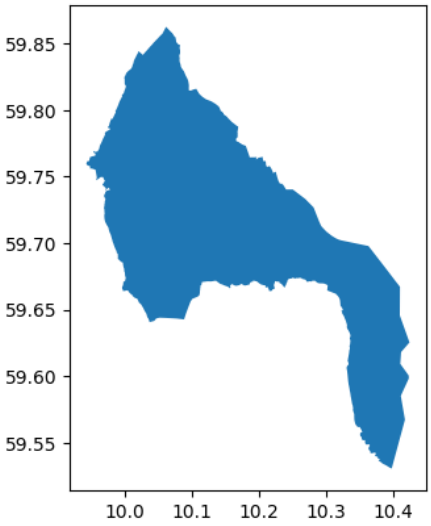


Figure 4.4.: Outline of Drammen using LangChain and OpenAI's Functions calling

5. Discussion and Conclusion

Sections 5.1 and 5.2 will discuss the test results from section 4.2 and provide suggestions as to how the limitations highlighted by the tests can be mitigated. Section 5.3 will conclude this specialization project report, and provide directives for future work on the subject of LLM-powered GIS.

5.1. Using ChatGPT's Code Interpreter for Geospatial Analysis

When using ChatGPT's built-in Code Interpreter with file uploads, it became apparent that it runs in a Linux environment and utilizes a mounted drive in the `/mnt` directory, typically used for temporarily mounted filesystems. In an initial test on the SOSI data format, it tried to execute this 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 certainly not available in the standard Linux environment used for ChatGPT's Code Interpreter. As the SOSI standard is still widely used for Norwegian geospatial purposes (though according to its Wikipedia page[2] 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 of flexibly manipulating the Linux environment used by the Code Interpreter then clearly poses some limitations when developing LLM-based systems. One solution is to create a custom environment on a server that we control ourselves. Having the AI agent run in an environment that we have full control over, gives us greater flexibility, and we can then grant the agent access to powerful GIS tooling, such as the GDAL library. The open-source "Open Interpreter" project (KillianLucas, 2023) could prove useful as an alternative to the closed-source Code Interpreter from OpenAI. Open Interpreter lets us run the interpreter on the computer/server of our choosing, so that

[1] https://github.com/kartverket/fyba
[2] https://no.wikipedia.org/wiki/SOSI-formatet

it can access the file system directly, as well as libraries and programs stored on the computer/server. It also allows for other LLMs than GPT-4, such as Mistral LLMs and Code-LLaMA. Additionally, with it being an open-source project, one can "fork" the repository and make project-specific modifications to the interpreter.

5.2. Mitigating ChatGPT's Inability to Access Web APIs

As the results from Test 2 (see subsection 4.2.2) show, ChatGPT-4 struggles when provided with URLs to external web APIs, even when prompted to use its web browsing abilities and pairing them with its Code Interpreter. These issues are not present when using direct file upload, in which case the model appears to save the uploaded file in a temporary file directory in its Linux environment. Interpreting the inner workings of the Code Interpreter from the code samples in the chat can be challenging, but it appears that it does not do this by default after fetching data from an external web API. As Listing 4.1 shows, it *truncates* the file contents and "stores" them directly in the code, in an attempt to keep the entire file contents within the context window of the LLM. The context window of the ChatGPT-4 model is currently at 32,000 tokens, and the new GPT-4 Turbo has a context length of 128,000 tokens. While these are of significant size, they are not meant to (or able to) store large file. The size of the context window therefore becomes a limiting factor when the file contents grow large, which is not uncommon for geospatial files.

This is a significant limitation of using ChatGPT-4 out of the box, and one should therefore look into other ways of handling web requests and subsequent storing of the received data. Techniques within Retrieval Augmented Generation (RAG), and libraries like LangChain and Open Interpreter, could help solve this issue.

5.3. Conclusion and Future Work

This specialization project report has presented a literature study in the fields of Natural Language Processing, Large Language Models, GIS, planning for LLMs, and Retrieval Augmented Generation, along with three tests that try to demonstrate strengths and weaknesses of LLMs when dealing with geospatial data. The literature study and tests serve to provide a better starting point when attempting to develop LLM-based GIS agents.

One important finding of the literature study is that there have been a substantial body of work concerning the potential of using Large Language Models in GIS analysis (see section 3.1). Studies have been conducted to show that the GPT-4 model has good geospatial awareness, and there have been created a number of prototypes of autonomous GIS systems powered by LLMs. Section 3.2 discussed various planning strategies that have been developed to facilitate better decision making and reasoning in LLM-based systems. Section 3.3 introduced Retrieval Augmented Generation (RAG), the concept of providing LLMs with access to external tooling to help them produce informed and up-to-date responses.

The three tests showcased GPT-4's capability to handle geospatial data using various data formats and access channels. The results from Test 1 and 2 showed that it has *some* understanding of how to perform geospatial analysis, but that it also has substantial limitations, e.g., reading and writing large files and accessing data through web APIs. Test 3 (see subsection 4.1.3) served as an initial test of utilizing tools such as LangChain to extend the capabilities of LLMs like GPT-4 through programmatic methods.

With this being a specialization project that will transition into a larger master thesis, some points of discussion have been reserved for future work. Additionally, the task of developing a proof of concept has been assigned to the master thesis due to time constraints and the intention to acquire more knowledge before proceeding with development. Subsections 5.3.2 and 5.3.1 will elaborate upon potentially important issues that should be addressed when developing LLM-based GIS agents.

5.3.1. Memory and Embeddings

Storing information for future use is important when developing LLM-based agents in order for them to produce consistent responses. Weng (2023) presents three different types of memory in human brains: (1) *Sensory Memory*, (2) *Short-Term Memory*, and (3) *Long-Term Memory*. When translated to LLMs, we can think of *Sensory Memory* as learning embedding representations, *Short-Term Memory* as the memory contained within the limits of the context window of the Transformer, and *Long-Term Memory* as an external vector store that can be attended to by the agent at query time. Such a vector store/database would store the vector embeddings of the data contained within it, and allow for fast and accurate similarity search and retrieval based on the vector distance or similarity between the vector representations (evchaki, 2023). Weng (2023) lists some common approximate nearest neighbours algorithms for fast retrieval speeds, including Locality-Sensitive Hashing (LSH) and Facebook AI Similarity Search (FAISS).

Future work should build upon the research of Unlu (2023) (see section 3.1) and investigate whether vector embeddings can be utilized for long-term storage of geospatial data with textual descriptions, or if a vector database can efficiently retrieve relevant resources such as APIs or other external tools based on their documentation/specifications. Furthermore, these documentations and API specifications can be large is size, and the context length could become a limiting issue. Vector embeddings can help mitigate such issues. By splitting the documents into chunks and indexing them using vector embeddings, one can extract only the relevant parts and pass these to the LLM with the prompt.

5.3.2. Testing 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 section 3.1), results were generally presented in the form of case studies. This type of qualitative testing is entirely appropriate for showcasing the possibilities of the technologies, but it

may be insufficient for comparing the performance of *different* systems. 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 outputs 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 between different user groups (see section 2.4). Outputs could be files with typical geospatial extensions (.shp, .geojson, .sos, etc.), or they could adhere to API specifications from geospatial standards (see section 2.3).

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

- 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 responses to questions that should *not* return geospatial files, e.g., answers to general questions about geo-related subjects?

These questions are outside the scope of this specialization project, and are therefore left out for future work.

Bibliography

Agent Protocol. (n.d.). Retrieved December 11, 2023, from https://agentprotocol.ai/

Ceylan, B. (2023). Large Language Model Evaluation in 2023: 5 Methods. Retrieved December 16, 2023, from https://research.aimultiple.com/large-language-model-evaluation/

Chase, H. (2022). LangChain. Retrieved October 5, 2023, from https://github.com/langchain-ai/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. https://doi.org/10.48550/arXiv.2107.03374

Cleary, D. (2023). Latency Benchmarks and Comparisons for OpenAI, Azure, and Anthropic. Retrieved December 11, 2023, from https://medium.com/@dan_43009/latency-benchmarks-and-comparisons-for-openai-azure-and-anthropic-6f035f1acab6

Datatilsynet. (n.d.). General Data Protection Regulation. Retrieved December 5, 2023, from https://www.datatilsynet.no/en/regulations-and-tools/regulations/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805

evchaki. (2023). Vector Database. Retrieved December 11, 2023, from https://learn.microsoft.com/en-us/semantic-kernel/memories/vector-db

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. https://doi.org/10.48550/arXiv.2310.03533

Firat, M., & Kuleli, S. (2023). What if GPT4 Became Autonomous: The Auto-GPT Project and Use Cases. *Journal of Emerging Computer Technologies*, *3*(1), 1–6. https://doi.org/10.57020/ject.1297961

Gaman, M., Hovy, D., Ionescu, R. T., Jauhiainen, H., Jauhiainen, T., Lindén, K., Ljubešić, N., Partanen, N., Purschke, C., Scherrer, Y., & Zampieri, M. (2020). A Report on the VarDial Evaluation Campaign 2020. *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, 1–14. Retrieved October 16, 2023, from https://aclanthology.org/2020.vardial-1.1

Gemini Team & Google. (2023). *Gemini: A Family of Highly Capable Multimodal Models* (tech. rep.). Retrieved December 7, 2023, from https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring Massive Multitask Language Understanding. https://doi.org/10.48550/arXiv.2009.03300

Holmes, C. (2021). SpatioTemporal Asset Catalogs and the Open Geospatial Consortium. Retrieved October 23, 2023, from https://medium.com/radiant-earth-insights/spatiotemporal-asset-catalogs-and-the-open-geospatial-consortium-659538dce5c7

Hugging Face. (n.d.). Perplexity of fixed-length models. Retrieved December 9, 2023, from https://huggingface.co/docs/transformers/perplexity

KillianLucas. (2023). KillianLucas/open-interpreter. Retrieved December 13, 2023, from https://github.com/KillianLucas/open-interpreter

Kumar, V. (2023). What are people asking to ChatGPT? Retrieved October 25, 2023, from https://varunon9.medium.com/what-are-people-asking-to-chatgpt-f5f324a6cc27

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, *33*, 9459–9474. Retrieved December 6, 2023, from https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Li, J., Galley, M., Brockett, C., Gao, J., & Dolan, B. (2016). A Diversity-Promoting Objective Function for Neural Conversation Models. In K. Knight, A. Nenkova & O. Rambow (Eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 110–119). Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1014

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., & Li, Y. (2023). API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. https://doi.org/10.48550/arXiv.2304.08244

Li, Z., & Ning, H. (2023). Autonomous GIS: The next-generation AI-powered GIS. https://doi.org/10.48550/arXiv.2305.06453

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out*, 74–81. Retrieved December 7, 2023, from https://aclanthology.org/W04-1013

Maeda, J. (2023). AutoGen Agents Meet Semantic Kernel. Retrieved December 11, 2023, from https://devblogs.microsoft.com/semantic-kernel/autogen-agents-meet-semantic-kernel/

Mæhlum, L., & Rød, J. K. (2023). SOSI. *Store norske leksikon*. Retrieved October 23, 2023, from https://snl.no/SOSI

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? Retrieved October 6, 2023, from https://research.ibm.com/blog/retrieval-augmented-generation-RAG

Mistral AI. (2023). Mistral 7B. Retrieved December 7, 2023, from https://mistral.ai/news/announcing-mistral-7b/

Mitra, A., Del Corro, L., Mahajan, S., Codas, A., Simoes, C., Agarwal, S., Chen, X., Razdaibiedina, A., Jones, E., Aggarwal, K., Palangi, H., Zheng, G., Rosset, C., Khanpour, H., & Awadallah, A. (2023). Orca 2: Teaching Small Language Models How to Reason. Retrieved December 12, 2023, from http://arxiv.org/abs/2311.11045

Mooney, P., Cui, W., Guan, B., & Juhász, L. (2023). *Towards Understanding the Geospatial Skills of ChatGPT: Taking a Geographic Information Systems (GIS) Exam*. https://doi.org/10.1145/3615886.3627745

Nascimento, E., García, G., Victorio, W., Lemos, M., Izquierdo, Y., Garcia, R., Leme, L., & Casanova, M. (2023). A Family of Natural Language Interfaces for Databases based on ChatGPT and LangChain.

Norge Digitalt. (2023). *Generelle vilkår for Norge Digitalt-samarbeidet* (tech. rep.). Retrieved October 23, 2023, from https://www.geonorge.no/globalassets/geonorge2/avtaler-og-bilag-norge-digitalt/generelle-vilkar.pdf

OGC. (2023). OGC Standards. Retrieved October 23, 2023, from https://www.ogc.org/standards/

OpenAI. (2022). Introducing ChatGPT. Retrieved October 26, 2023, from https://openai.com/blog/chatgpt

OpenAI. (2023). GPT-4 Technical Report. https://doi.org/10.48550/arXiv.2303.08774

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A Method for Automatic Evaluation of Machine Translation. In P. Isabelle, E. Charniak & D. Lin (Eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Qi, J., Li, Z., & Tanin, E. (2023). MaaSDB: Spatial Databases in the Era of Large Language Models (Vision Paper). https://doi.org/10.1145/3589132.3625597

Radford, A., & Narasimhan, K. (2018). Improving Language Understanding by Generative Pre-Training. Retrieved October 9, 2023, from https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035

Richard, T. B. (2023). AutoGPT: The heart of the open-source agent ecosystem. Retrieved October 5, 2023, from https://github.com/Significant-Gravitas/AutoGPT

Roberts, J., Lüddecke, T., Das, S., Han, K., & Albanie, S. (2023). GPT4GEO: How a Language Model Sees the World's Geography. https://doi.org/10.48550/arXiv.2306.00020

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., & Synnaeve, G. (2023). Code Llama: Open Foundation Models for Code. Retrieved October 30, 2023, from http://arxiv.org/abs/2308.12950

Scherrer, Y., & Ljubešić, N. (2020). HeLju@VarDial 2020: Social Media Variety Geolocation with BERT Models. *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*, 202–211. Retrieved October 16, 2023, from https://aclanthology.org/2020.vardial-1.19

Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., & Yih, W.-t. (2023). REPLUG: Retrieval-Augmented Black-Box Language Models. https://doi.org/10.48550/arXiv.2301.12652

Skjuve, M., Bae Brandtzaeg, P., & Følstad, A. (2023). Why People Use ChatGPT. https://doi.org/10.2139/ssrn.4376834

Sparrow, R. (2007). Killer Robots. *Journal of Applied Philosophy*, *24*(1), 62–77. https://doi.org/10.1111/j.1468-5930.2007.00346.x

STAC Tutorials. (n.d.). Retrieved October 23, 2023, from https://stacspec.org/en/tutorials/

Stasaski, K., & Hearst, M. (2022). Semantic Diversity in Dialogue with Natural Language Inference. In M. Carpuat, M.-C. de Marneffe & I. V. Meza Ruiz (Eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 85–98). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.6

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., & Wei, J. (2022). Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. Retrieved December 9, 2023, from http://arxiv.org/abs/2210.09261

The Norwegian Mapping Authority. (2019). *Elveg 2.0* (tech. rep.). Retrieved December 2, 2023, from https://register.geonorge.no/data/documents/Produktspesifikasjoner_elveg-2-0_v2_sosi-standardisert-produktspesifikasjon-elveg-2_0_1_.pdf

The Norwegian Mapping Authority. (2023). *Håndbok for Geovekst-samarbeidet* (tech. rep.). Kartverket. Retrieved October 10, 2023, from https://kartverket.no/geodataarbeid/geovekst/veiledningsmateriell-geovekst/

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., & Scialom, T. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. https://doi.org/10.48550/arXiv.2307.09288

Unlu, E. (2023). Chatmap : Large Language Model Interaction with Cartographic Data. https://doi.org/10.48550/arXiv.2310.01429

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. Retrieved October 10, 2023, from https://arxiv.org/abs/1706.03762v7

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. https://doi.org/10.48550/arXiv.2201.11903

Weng, L. (2023). LLM Powered Autonomous Agents. *lilianweng.github.io*. Retrieved December 10, 2023, from https://lilianweng.github.io/posts/2023-06-23-agent/

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. https://doi.org/10.48550/arXiv.2308.08155

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. https://doi.org/10.48550/arXiv.2305.10601

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). HellaSwag: Can a Machine Really Finish Your Sentence? https://doi.org/10.48550/arXiv.1905.07830

Zhang, Y., Wei, C., Wu, S., He, Z., & Yu, W. (2023). GeoGPT: Understanding and Processing Geospatial Tasks through An Autonomous GPT. https://doi.org/10.48550/arXiv.2307.07930

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. Retrieved October 30, 2023, from http://arxiv.org/abs/2310.04406

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

NTNU
Faculty of Engineering Science and Technology
Department of Civil and Environmental Engineering

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)

Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

AWS Amazon Web Services.

BERT Bidirectional Encoder Representation from Transformers.

BLEU BiLingual Evaluation Understudy.

CLI Command Line Interface.

EU European Union.

FAISS Facebook AI Similarity Search.

FKB Felles KartdataBase.

GAN Generative Adverserial Network.

GDAL Geospatial Data Abstraction Library.

GDPR General Data Protection Regulation.

GIS Geographic Information System.

GML Geography Markup Language.

GPT Generative Pre-trained Transformer.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

ISO International Organization for Standardization.

JSON JavaScript Object Notation.

LATS Language Agent Tree Search.

LLaMA Large Language Model Meta AI.

LLM Large Language Model.

LSH Locality-Sensitive Hashing.

LSTM Long Short-Term Memory.

MBPP Mostly Basic Python Programming.

MLM Masked Language Modelling.

MMI Maximum Mutual Information.

MMLU Multitask Language Understanding.

NLG Natural Language Generation.

NLI Natural Language Inference.

NLIDB Natural Language Interfaces for Database.

NLP Natural Language Processing.

NLU Natural Language Understanding.

NSDI National Spatial Data Infrastructure.

NSP Next Sentence Prediction.

OGC Open Geospatial Consortium.

OSM OpenStreetMap.

PPO Proximal Policy Optimization.

RAG Retrieval Augmented Generation.

RLHF Reinforcement Learning from Human Feedback.

RNN Recurrent Neural Network.

ROUGE Recall-Oriented Understudy for Gisting Evaluation.

SDK Software Development Kit.

SOSI Samordnet Opplegg for Stedfestet Informasjon.

SQL Structured Query Language.

STAC SpatioTemporal Asset Catalog.

TC Technical committee.

UI User Interface.

WFS Web Feature Service.

WMS Web Map Service.

XML Extensible Markup Language.