Bachelor Thesis

DataViz: A Business Data Visualization System Using LLMs

Author: Maja L. Bińkowska (2761364)

1st supervisor	Prof. dr. ir. Alexandru Iosup
daily supervisor	Msc. Xiaoyu Chu
2nd reader	Prof. dr. ir. Jiahuan Pei

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

July 1, 2025

Abstract

The rapid growth of data is expected not only to accelerate but also reshape business transformation across established enterprises. Meaningful data handling and visualization allows companies to progress, drive innovation, and maintain market position. However, current business workflows have significant bottlenecks, such as inter-departmental dependencies, limited knowledge in business intelligence tools, and inefficient data visualization processes. In this context, the emergence of Natural Language Interfaces (NLIs) and Generative Artificial Intelligence (GenAI) has accelerated the applications of Large Language Models (LLMs) for data analytics. This thesis aims to design, implement, and evaluate a business data visualization system (`DataViz`) that automated processing large-scale data, and generates accurate and useful visual data insights. We first analyze the stakeholders, use cases and requirements of the system, provide the design overview, and core modules. Then, we implement `DataViz` as a prototype system, including the implementation of front- and back-end with API calling modules. We showcase the data-processing and visualization pipeline based on an real-world example. We design and conduct evaluation experiments to assess the accuracy, performance, and usability of the system. `DataViz` demonstrates high accuracy, with 3,7% visualization error rate and highly positive user feedback, 80% *very likely* responses, on LLM and data analytics integration based on a usability study. The system allows for large-scale visualization generation, with average script execution latency of 11.29 seconds across 10 KB-10 GB data files. This LLM-based analytics system can pave the way into the everyday business workflows of data-driven companies.

Keywords: Data Analytics, Large Language Model, Natural Language Interface, Human-Computer Interaction, Natural Language Processing, System Design, Prompt Engineering, Case Study

Contents

1

Introduction

The big data movement is rapidly reshaping business transformation across established business enterprises. Various sectors face the impact of increasing velocity of data; the US healthcare system alone reached, in 2011, 150 exabytes and is predicted to reach zettabyte (1021 gigabytes) soon [21]. GenAI solutions have shown to improve time inefficiencies, outcomes, decision-making of various business models, and transform workflows to fit the latest technological solutions [36]. The McKinsey Global Institute predicts that Gen-AI usage has the highest profit potential for automation of operations in companies, and it is estimated that such implementations could add the equivalent of $2.6 trillion to $4.4 trillion annually in value [9]. Reliance on mass data allows shifting from the predictive view of operations to visualizing mass growing data, promotes successful data analytics in real-time, and reduces the cognitive burden of extracting insights from large tabular datasets [13, 29, 60]. The current competition within companies is to deploy LLMs to overcome the visualization paradigm. It represents the difficulty of users in translating their intentions for analysis into tool-specific operations, by natural language instructions [4, 77]. This is strongly motivated by visualization authoring being a complex task, involving multiple consecutive actions, all of which require expertise and are error-prone for users with limited visualization experience [13]. For instance, Unilever's digital infrastructure is complex and its network handles on average 240 TB of data a week, for more than 3 billion transactions, and is actively implementing AI-driven solutions, as 500 projects have been implemented around the world over the last decade [39, 75]. Unilever, among the world's leading consumer goods companies, has taken action to include Gen AI solutions in their business operations [68]. The research of NLIs is prominent and allows users to have simplicity, interaction with data using natural language questions (NLQs) [33, 35,

43, 44, 77]. However, implementation of NLIs includes the inherent ambiguity of natural language and the common underspecification of user queries [13, 35].

The integration of NLI is becoming apparent (IBM Watson Analytics, Microsoft Power BI and Tableau), however, there is a gap in research for systematic system design and implementation and evaluation of AI-enabled data analytics systems tailored to enterprise-specific needs [22, 35, 41, 43, 44, 70, 77]. It is challenging to create a fully-automated data visualization NLI, which will be able not only to process large-scale data but also seamlessly blend into current business workflows of world-leading companies. This paper demonstrates how effective Prompt Engineering (PE) can be conducted in order to extract from LLMs the desired code for visualizations. This approach demonstrates gains in accuracy, performance, usability, and resource reduction in the development of an LLM-driven data visualization system [35].

1.1 Problem Statement

Despite efforts from the research community to provide open source LLM-driven implementations and architectures for fully automated data analytics systems, we identify the following bottlenecks [75].

The complexity of extracting, aggregating, and wrangling information from large-scale datasets affects the time efficiency of performing data analytics. It was noticed that the pace of workforce needed to prepare PowerBI dashboards from singular or/and multiple datasets is rather time consuming and does not ensure elimination of a human error. Partial or complete transformation of this process could enable for an accelerated pace due to technological automation. GenAI was estimated to improve the growth of labor productivity from 0.1 to 0.6 % annually through 2040, depending on the rate of adoption of technology and the redistribution of worker time [9]. Furthermore, combining AI-driven solutions with other technologies currently used could add 0.5 to 3.4 percentage points annually to productivity growth [9].

The need for tool-specific knowledge of BI tools and functions is vital to create meaningful visual data insights. BI graphical interface expects the user to have a deep understanding of single or multiple datasets and be able to translate the visualization needs into tool-specific functions [17, 58, 87]. This affects the validation processes of the data extracted, processed, and represented. Defective mapping to unexplored areas due to a large number of attributes and values, insufficient navigation to useful tools to create meaningful insights, and limited understanding of tool preferences for certain data visualizations can enhance

the risk of error-prone analytics [58]. This introduces the possibility of human error, as the data analyst must engage in a trial-and-error process, resulting in fluctuations and limited precision of data prediction in the declaration of the strategy [74]. Although it is challenging to automate the data visualization process, this paper will try to design a visualization tool that allows for fidelity, assuring faithfulness of the visuals that are relevant to the user prompt and dataset.

The reliance of non-technical users on the data visualization development team to create new or alter existing data visualizations creates further time inefficiencies. In this thesis, we refer to non-technical users as enterprise workers who have limited or no background in dashboard and data visualization creation. We refer to technical users as enterprise workers who are advanced in dashboard creation, maintenance, and data analytics. These dependencies make time-sensitive decisions based on the data without having to contact the technical team, which affects the time-inefficiency bottleneck. This calls for the research and deployment of a fully automated LLM-driven data visualization system based solely on unrestricted NLQ. The acceleration in the potential for technical automation was said to be mainly due to the ability of Gen-AI to process NL and reason about it. This accounts for approximately 25% of the total work time spent [9].

1.2 Research Questions

To address the aforementioned challenges, we propose the main research question **(MRQ)**: **How to design, implement, and evaluate a LLMs-based business data visualization system?** The **MRQ** can be divided into four sub-questions **(RQ)**:

RQ1 What are the state-of-the-art LLMs applications in data analytics and visualization? The observed gap in research for systematic design, implementation, and evaluation of LLM-driven data visualization systems adapted to the enterprise is worth noticing. Although, advancements in LLM-driven data analytics demonstrate automation of data visualization generation, enterprise employees, of any data science expertise, often work on large-scale, sensitive data and require domain-specific solutions tailored to their business workflows. Nevertheless, it is challenging to propose a fully-flexible system for visual data analytics, which allows large-scale data processing due to varying system requirements. This question aims to explore the current state-of-the-art and novel approaches to data analytics automation.

RQ2 How to design a business data visualization system driven by LLMs? The bottlenecks in the enterprise workflow described in Section 1.1 call for LLM-driven approaches to independent data visualization. This poses a challenge of aligning the design requirements of the LLM-driven large-scale visualization system with the enterprise needs. This highlights the need to adapt established workflows without disrupting existing data practices but enhancing them with the usage of GenAI. The design should not allow for secure and large-scale data analytics, but also try to maximize accuracy, performance, and usability of the system.

RQ3 How to implement a business data visualization system driven by LLMs? The implementation process should allow the system to be accessible to company employees. This is challenging due to alignment with domain-specific workflows of companies and needed authorizations of API usage. This system should be implemented in a way to be deployed in a hosting environment, for instance DataLab and allow LLM API authorization without data privacy infringements [74]. It should try to ensure usability with dynamic interface and allow functionalities useful for users and allow for time-sensitive data visualization.

RQ4 How to evaluate a business data visualization system driven by LLMs? The final complex task in developing an LLM-driven large-scale visualization system includes the implementation evaluation process. With the understanding provided by **RQ3**, the system would need to be evaluated in the fields of Human-Computer Interaction (HCI), Natural Language Processing (NLP), and data visualization generation. This is challenging due to the vast scope of varying fields involved, and ensuring accurate, efficient, low-resource, and useful visual insights. The evaluation should be designed and performed on the basis of the existing literature and the baseline generated using **RQ1**.

1.3 Research Methodology

In addressing the research questions (Section 1.2), we employ the following research methodologies **(RM)**:

RM1 To address **RQ1**, we perform and document a time-constrained systematic literature review of state-of-the-art and novel systems for AI-driven analytics. We begin by explaining the methodology of the study and present the systems and their subsequent evaluation approaches. We summarize and analyze the most relevant work, the PE approaches, the terminology, and the metrics, which will all be essential to answer the next research questions.

RM2 To address **RQ2**, based on the knowledge derived from the systematic review of the literature, we will begin the design process by in-depth requirements analysis for the large-scale data visualization system driven by LLM. We will design the NLI based on the approaches recommended in the literature to ensure the usability of the proposed User Interface (UI) and Experience (UX) [18, 46]. We will design the data visualization pipeline to ensure the accuracy of the generated plots and well-performing large-scale data processing.

RM3 To address **RQ3**, the process will involve a series of implementation and prototyping of the LLM-driven data visualization system. The development process will be conducted in an agile approach and progress updated during bi-weekly sprint runs. This ensures that progress is visible and that the learning process was successfully documented. We want to ensure reproducibility; therefore, both source code and experimental material will be made public.

RM4 To address **RQ4**, a series of experiments will be designed and conducted to evaluate the accuracy, performance, and usability of the system. The studies will be based on an anonymized case study dataset derived from Unilever, which will be transformed and scaled accordingly and will have adequate metrics. Core experiment choices, such as NLQ preparation, will be motivated by research, and finally to ensure usability, a user study will be conducted within the business enterprise setting to evaluate the HCI of the system. This will allow for a comprehensive assessment and limitation, as well as future work recommendations of the proposed system.

Throughout all stages of answering the aforementioned research questions, we impose a strong emphasis on open and reproducible science. The proposed system follows standard practices and is accompanied by documentation that includes the latest LLM-driven solutions and uses the acquired knowledge base for the design, implementation, and evaluation processes. The documentation will be open source in hopes of encouraging the community to expand on it, as research on the deployment of AI-driven data analytics solutions in business settings is rapidly evolving fields [3].

1.4 Research Contributions

Addressing the research questions, our study makes five research contributions **(RC)**:

RC1 (Conceptual) We conduct a systematic literature review on the state-of-the-art LLMs applications for data analytics and visualization. We learn that existing LLM-driven visualization systems often understate large-scale data processing (**F2.1**) and that prompt engineering techniques, such as SCOT, can improve LLM's code generation capabilities (**F2.3**).

RC2 (Conceptual) We design a business data visualization system (`DataViz`) that automatically performs data analytics using LLMs. We systematically analyze 6 key stakeholders, 3 use cases, and define 5 functional and 5 non-functional requirements for the system. We further propose different system components, including critical modules for natural language interface, error handling, and prompt engineering. We also design an evaluation framework for assessing accuracy, performance, and usability of the system.

RC3 (Technical) We implement and evaluate the system for real-world business data visualization. We apply different evaluation metrics to assess the accuracy, performance, and usability of `DataViz`. For accuracy and performance, we design user prompts covering common data visualization use cases and data combinations. For usability, we conduct a real-word user study and collect feedback from exemplary company employees. We demonstrate that a fully-automated, visualization-first system driven by LLMs can be implemented deploying a lightweight architecture (**F4.1**). We evaluate that it can produce accurate and timely data analytics for varying-scale datasets with 3.7% of visualization error rate and an average execution latency of 11.29 seconds across 10 KB-10 GB files (**F5.1**, **F5.4**).

RC4 (Open Science) We follow the FAIR principles and will release the data and code as a contribution to open science[1].

RC5 (Personal Development) This thesis has a significant impact on the author's personal development and help the author build scientific and technical skills in the research of LLM-driven solutions for data science and analytics.

1.5 FAIR Data and Software

We strive to adhere to the FAIR principles of scientific data and software. FAIR, which stands for Findability, Accessibility, Interoperability, and Reuse of digital assets, is a set of guidelines to ensure open science and reproducibility of the used dataset [3]. This thesis is evaluated based on anonymized case study data (2.2 MB). The implementation and experiment documentation are available on GitHub [2] [3] (**RC4**).

1.6 Statement of Non-Plagiarism

I confirm that this thesis is my own work, is not copied from any other source (person, Internet, or machine), and has not been submitted elsewhere for evaluation. The work, findings, and formulations that do not represent my contribution are explicitly acknowledged by citations.

1.7 Thesis Structure

In Chapter 2, we address **RQ1** and conduct a literature review and present relevant background information (**RC1**). Chapter 3 relates to **RQ2** and specifies the core design choices made based on the summarized and documented research (**RC2**). In Chapter 4, we answer **RQ3** and describe the implementation choices behind `DataViz` (**RC3**). Chapter 5 we address **RQ4** and summarize the experimental setup and present the core findings to ensure that all system requirements were covered (**RC3**). Chapter 6 conveys the design, implementation and evaluation limitations and consequent future work. Chapter 7 summarizes the thesis by answering the research questions.

[1] The data and code will be accessible after the evaluation of Unilever.

[2] `https://github.com/mmyalen/DataViz`

[3] `https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz`

2

Literature Review

For addressing **RQ1**, this chapter conducts a systematic literature review on LLMs applications in data analytics. We begin by explaining the key technical terminology in Section 2.1 for ensured understanding of the following chapter and thesis.; We derive the context and scope of LLMs approaches in data analytics and visualization (Section 2.2); We describe the methodology of conducting a systematic literature review (Section 2.3), and is followed by a synthesis of core research findings with a focus on LLM-driven analytics and novel PE approaches; The summary of the key findings of the literature review is included in Section 2.5.

2.1 Terminology in LLM-Driven Analytics Systems

Data Analytics System is a framework for data collection, aggregation, processing, and visualization, which is highly automated, friendly to non-experts and generic, enhanced by error handling [85, 89].

LLM *are computational models that have the capability to understand and generate human language. LLMs have the transformative ability to predict the likelihood of word sequences or generate new text based on a given input* [7].

Generative Pre-trained Transformer (GPT) refers to the first Transformer-based pre-trained LM that can effectively manipulate the semantics of words in terms of context [45]. It is pre-trained on a massive corpus of text, have acquired a vast amount of knowledge, and can be utilized for various downstream tasks, including language translation, text classification, and data parsing [78]. Exemplary GPT is the OpenAI Transformer, which uses the decoder of the transformer to model the language as it is an auto-regressive model

where the model predicts the next word according to its previous context. GPT is unidirectional, that is, the model is only trained to predict the future context from left to right, which assures its content generation capabilities [45].

Hallucination Although language models can generate fluent and informative answers to human questions, they may not always be accurate [8, 71]. Online models such as Codex, GPT-3, and GPT-4 are not fully controllable and stable, as they suffer from hallucination problems, and occasionally provide unstable output with incorrect answers, leading to failure to follow the designed pipeline [71].

Prompt Engineering (PE) (zero-shot and few-shot) allows for communication and interaction with LLMs. PE can be leveraged in enforcing rules for improved output from LLMs [37]. Zero-Shot Learning directly feeds the requirement into LLMs without examples. Then, it extracts a program generated from LLM outputs [28]. Few-Shot Learning randomly selects several requirement and code pairs as examples. Given a requirement, it concatenates the examples and the requirement together, making a prompt. Then, the prompt is sent to LLM, and LLM predicts a new program [28].

Natural Language to Visualization (NL2Viz) aims to transform natural language descriptions into visual representations for a grounded table, allowing users to gain insights from vast amounts of data [82]. Implementing NL2Viz is particularly challenging due to the ambiguity and underspecification of the requirements on the prompts, which makes accurate data processing and visualization prone to failure [35, 66].

Natural Language Interface (NLI) serves as a complementary input modality for visual analytics. It can provide users with an engaging experience and generate data insights in a time-efficient manner [63].

User Study is a cost-effective approach to gather data on stakeholders, their needs, and intentions. Consequently, to translate them into user requirements that support the development of useful and usable products [25, 64]. Conducting a user study with enterprise data simulates real-world behaviors and enhances documentation with a realistic evaluation.

2.2 Context and Scope of LLMs for Data Analytics

Existing studies demonstrate a notable focus on LLM-driven data visual generation (both with and without accompanying text data analysis, as well as prompt optimization and engineering techniques [31, 81]. However, this leaves a critical gap to systematically explore broader applications of LLMs that are suitable for visualization-first lightweight system

that allows for large-scale, independent, efficient and context-aware data processing, aggregation, and visualization. This is imperative in today's market, enabling the adoption of current solutions and the exploration of new opportunities to grow and remain competitive, especially for leading industrial and business enterprises.

In response to these gaps, our literature review adopts a systematic approach to examine the LLM-driven data analytics [5, 27, 83]. It involves a comprehensive search in multiple academic databases using well-defined keywords related to this area. The process follows a comprehensive literature review methodology, ensuring that only the most relevant studies are included in the analysis.

2.3 Methodology for Systematic Literature Review

To perform a systematic review to explore state-of-the-art LLM-driven systems for data visualization analytics, we depict the process in Figure 2.1, following the systematic literature review steps from the study [5, 83]. This methodology was used to ensure a rigorous and reproducible literature synthesis, providing a comprehensive understanding of the current applications of LLMs in the field of data analytics, specifically context-aware data visualization generation from the free-text form prompt. This structured approach establishes the foundational framework for identifying, categorizing and evaluating relevant contributions in this emerging area.

Our systematic literature review process consists of 8 steps, organized into 3 stages. Steps 1 and 2 produce the *planning stage*. Step 1 defines the scope of the study, Step 2 describes a preset plan to use in conducting the review, which should be strictly followed to ensure a systematic and reliable search. We also describe the keywords for searching literature. For broader LLM based data visualization systems, we use keywords such as: "LLM data visualization", "LLM data insight generation", "AI Data Analytics", and "AI data visualization". In *conducting stage*, Step 3, 4, and 5 depict the search strategy of the literature, which was conducted using the listed keywords with the use of multiple academic databases, including Google Scholar, arXiv, IEEE Xplore, and ResearchGate. The articles found were searched using the simplified method, which narrows the body word by first scanning for relevance in the title, then in the abstract with full text review when the steps done before were applicable and relevant to the stated research questions. In Steps 6 and 7 relevant data from the papers were extracted. The findings were documented in Step 8 as the *documenting stage*.

The key findings are documented in Table 2.1, which describes, for each review article, its application to the system, the models deployed, and the datasets used for evaluation.

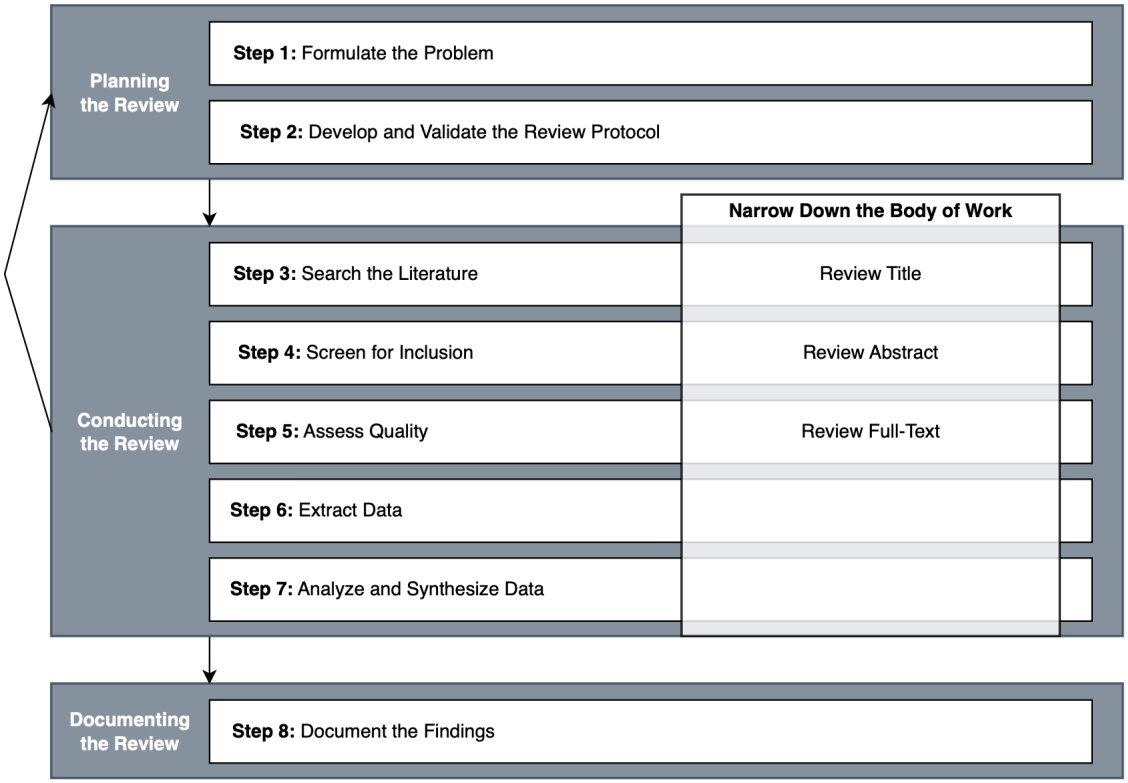


Figure 2.1: Overview of the systematic literature review process for this thesis, reproduced from [83].

2.4 Applications of LLMs for Data Analytics

The methodology outlined in Figure 2.1 facilitated the identification and comparative analysis of recent LLM-driven data visualization frameworks, summarized in Table 2.1. It is worth noting that the provided table only includes core reference systems, which are not exhaustive.

These systems employ advanced language models including GPT-4, and fine-tuned variants, which support natural language interfaces for data processing and visualization generation. Each one proposes a tailored and varying set of functionalities and architectures used from schema filtering and in-context learning to modular pipelines. The reviewed applications are benchmarked on heterogeneous datasets, ranging from domain-specific collections like NVBench and IMDb to custom or open-source datasets [32]. NL4DV is

frequently adopted as evaluation baseline due to its rule-based design [44]. This literature documentation stresses the rapid development and varied approaches in integrating LLMs for visualization tasks.

The gap identified through this review refers to the lack of lightweight solutions capable of data visualization for large-scale datasets, specifically those that exceed size of 30MB.

2.4.1 Applications of LLMs for Data Visualization

Within recent significant advancements in LLMs with online models, such as Codex, GPT-3, and GPT-4, Gemini, as well as open-source models Flan-T5 and LLaMa, it is worth using such tools and strengthening research toward their deployment in business and enterprise context [8, 10, 71, 72]. These models were proved to be effective in the analysis and extraction of relevant information from structured and unstructured data, as well as in the generation of code and web design [8, 58, 71]. Despite having shown effectiveness and improved performance LLM-based systems have documented limitations, such as providing insufficient explanations for the system's generated output and being inconsistent in generating data visualizations [32, 58]. These unexplainable, uncertain systems impact transparency and trust, making it difficult for users to find and fix errors. Several systems pretrained LLMs to generate Python-based visualization scripts. For example, Chat2VIS leverages prompt-engineered interactions with ChatGPT and Codex to generate code for user-uploaded CSV files, supporting free-form natural language queries [35]. LIDA extends this by supporting code generation for summarization, exploration, and infographic rendering using a modular LLM pipeline [13]. While the systems reviewed in Section 2.4 do not explicitly address large-scale data visualization, the study done by *Cheng et al.* evaluates the capacity of GPT-4 for large-scale data analysis by processing data files in smaller, manageable chunks to overcome the limitations of the context window. The reliance on token-limited contexts, output inconsistencies, and edge cases affect the fidelity and accuracy of LLM-generated code and subsequent data visualizations [32, 58].

Reference	Description	Model	Evaluation	Datasets
Prompt4Vis (2024) [29]	In-context learning using multi-objective example mining and schema filtering for simplified database search.	GPT-3.5-Turbo	Compared against ncNet, Seq2Vis, Transformer, RGVisNet using Vis Acc, Axis Acc, Data Acc, Overall Acc.	NVBench
Chat2VIS (2023) [35]	Streamlit-based web app using PE, description and code primers. Supports free-form NL queries and CSV uploads up to 30MB.	ChatGPT, Codex and GPT-3	6 case studies compared against NL4DV and ADVISor; robustness tested with ambiguous prompts.	nvBench, IMDb, Colleges, Energy, Products
ChartGPT (2023) [71]	Six-step reasoning pipeline powered by a fine-tuned LLM including multi-view interface, enabling users to inspect and modify intermediate outputs.	Fine-tuned Flan-T5-XL	Evaluated against NL4DV and ncNet using consistency, similarity (ROUGE-L, BLEU), and user study.	Custom dataset based on NL4DV
LIDA (2023) [13]	A novel tool using modular pipeline for data summarizing, goal exploring, code generating and a infographic module.	GPT-4, ChatGPT, PaLM, Cohere	Integrated into NL4DV toolkit for conversational interaction and ambiguity detection.	Vega datasets

Table 2.1: Summary overview of LLM-Based Data Visualization Applications.

2.4.2 Applications of Prompt Engineering for Code Generation

Methodologies that improve context awareness of LLMs and are domain-specific include prompting and fine-tuning [71] (Figure 2.2). Although one does not exclude the other, for the scope of this paper we will primarily focus on PE methods in hopes of creating a visualization-first system for large-scale data processing.

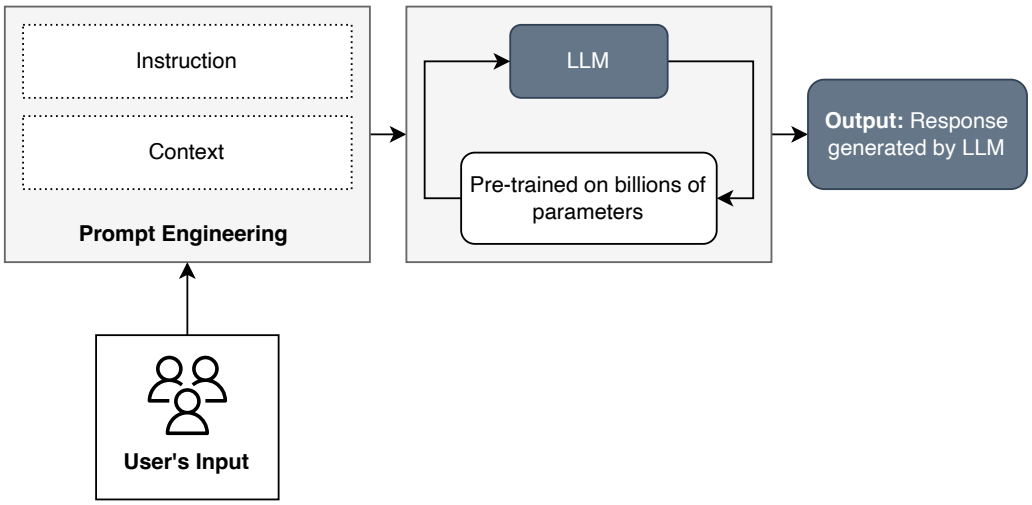


Figure 2.2: Visual breakdown of prompt engineering process. The user's input is provided as an instruction prompt through a natural language interface, combined with a context prompt, and then fed into the LLMs as a whole, reproduced from [59].

The term prompting refers to providing the model with a text that includes the context of domain tasks and expected outputs [71]. A prompt is a set of instructions provided to an LLM that is programmed by customizing it and refining its capabilities [80]. PE is a relatively recent discipline that focuses on developing and optimizing prompts to effectively utilize LLMs in a wide range of applications and research [19]. *Tian et al.* claim that context-aware models show more efficient results using only PE methods due to the simplified approach when the model does not need to rely on large amounts of domain-specific knowledge [71]. *Maddigan et al.* research that the most effective method to obtain the product tailored to the desired product is by using the "show-and-tell" technique. This is achieved by providing the API call with a Description Primer, which includes, i.e., table schema, column types or dataset preview, and a Code Primer, which serves as a starting point and guidance to LLMs output of the Python visualization script [35]. The limitation of this approach is that both of the prompt primers lead to extensive token usage, which makes the use of the API call more costly and less time-efficient. Novel solution of *V. Dibia* highlight latency limitations, as LIDA's integrates computationally expensive GPT-3.5, which requires significant compute resources to deploy at low latency. All the aforementioned systems are given in Table 2.1.

One of the most used methods is Chain-Of-Thought (CoT), which can be applied for zero-shot and few-shot prompting [71, 79, 88]. CoT improves the output of LLM by

structuring the prompt into several intermediate steps of natural language reasoning and can minimize token usage for online LLM pipelines [28, 49, 79]. It shows remarkable performance in varying natural language tasks and minimizes the risk of hallucination, as explained in Section 2.1. However, it tends to perform with lower accuracy when solving problems more often than the exemplars shown on the prompts and when applied to code generation tasks.[28, 88]. *Zhou et al.* propose a technique of least-to-most prompting for solving complex LLM-based tasks, yet for code generation tasks, determining problem decomposition can be non-trivial. *Li et al.* explore an enhanced version of CoT specifically designed for code generation [28]. Structured Chain-of-Thought (SCoT) prompting (Figure 3.7), which is an innovative prompting technique tailored for source code production due to incorporation of program structures (sequence, branch, and loop structures) into the reasoning steps [28]. *Li et al.* evaluated the effectiveness of SCoT on ChatGPT and Codex and demonstrated a superior performance over CoT induced by up to 13.79% [28, 59].

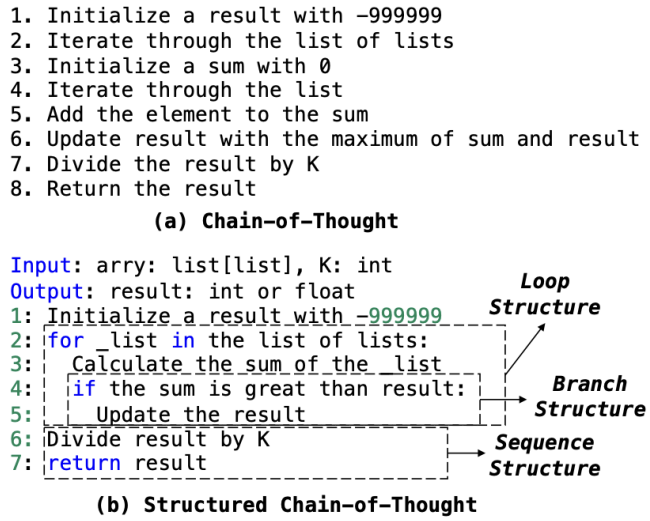


Figure 2.3: The comparison of a Chain-of-Thoughts (CoT) and Structured Chain-of-Thought (SCoT), taken from [28].

2.5 Summary of Literature Review

In this chapter, we addressed **RQ1** and conducted a systematic review of the literature on LLMs for data analytics. We systematically explored LLMs and prompt engineering techniques for the generation of data visualization scripts, summarizing current studies and gaps (as **RC1**). We begin by defining the context and scope of LLMs in data analytics, clarifying key terminologies used throughout this work. We clearly present our literature

review methodology and process in detail. Then we focus on two main application areas: (1) The use of LLMs for data visualization, and (2) Prompt engineering for code generation in data analysis. The summary of key work is listed below:

F2.1 The majority of the explored LLM-based systems prioritize usability over scalability, reducing the adoption of software to growing large and mass-scale data.

F2.2 Available LLMs, although allow effective natural language processing and data visualization efforts, are limited by API latency and the token context window.

F2.3 Prompt engineering methods such as CoT and SCoT show effective ways of providing context and improving accuracy for LLMs such as GPTx and Codex.

F2.4 The lack of systematic evaluation metrics and benchmarks reduces comparability between various LLM-based systems and limits reproducibility.

F2.5 The popular topic in this research area is focusing on resource minimization (token usage, latency minimization, and API utilization) of GPT models.

Due to rapid evolution of LLM integration and business workflow automation, future directions should focus on: (1) Structured evaluation of data visualizations, and (2) Accurate generation of visualization scripts.

3

Design of DataViz

For addressing **RQ2**, this chapter designs a system for the visualization of large-scale business data. We summarize and describe the core stakeholders, use cases and follow this by functional and non-functional requirements analysis in Section 3.1; We inspect the components of the design overview in Section 3.2 and explain in detail the design choices behind the NLI, PE, the data visualization pipeline and error handling in Section 3.3; Furthermore, we describe the metrics in Section 3.4, for the DataViz evaluation; The summary of the key work of the design is included in Section 3.5 and limitations in Section 6.1.

3.1 Requirements Analysis

To ensure that the large-scale data visualization system is aligned with the objectives of its intended stakeholders and the enterprise needs, we perform a requirements analysis. We identify key stakeholder groups (Section 3.1.1), their representative use cases (Section 3.1.2), and the corresponding functional requirements (Section 3.1.3) and non-functional requirements (Section 3.1.4). This matches stage (1) of the AtLarge Design Process [23].

3.1.1 Stakeholders

We distinguish the following stakeholders, the list of which is summarized in Table 3.1.

Tag	Stakeholder
S1	Non-technical Enterprise Employees
S2	Technical Enterprise Employees
S3	Enterprise Decision-making Units
S4	Researchers
S5	Students
S6	General Public

Table 3.1: Stakeholder Summary.

(S1) Non-technical Enterprise Employees are individuals without the technical skills to create data visualizations using BI-specific functions and tools from a singular or multiple datasets. They are responsible for the analysis of the generated charts for decision-making efforts or want to expand their understanding of data independently.

(S2) Technical Enterprise Employees are individuals who are proficient in using BI tools to create visual data insights. They seek to deepen their understanding of domain-specific data more time-efficiently, improve their data visualization skills further, or gain inspiration for data handling.

(S3) Enterprise Decision-making Units are divisions responsible of analyzing data visualization from large-scale data files for decision-making efforts that are substantial to enterprise evolvement and persistence on the market. They require accurate data representation for well-guided and timely decisions.

(S4) Researchers give us invaluable knowledge about the past, present, and future of LLM-driven visual data insights. Their job is to seek deep and reliable data analytics efforts and structured solutions that enterprises could benefit from. They investigate reliable applications of various models and techniques for large-scale data visualization pipelines.

(S5) Students represent not only individuals interested in the topic of data analytics, but also the future of research and efforts for AI-enabled automation solutions. They are the LLM systems architects and data science researchers of tomorrow. Therefore, a comprehensive understanding of LLM-based analytics systems is essential for them and should be made more accessible.

(S6) General Public may begin to look for more environmentally friendly solutions to the increasingly growing mass data. The comprehension of such assets is essential not only in their daily lives, but also in their understanding of current AI-based solutions and

independent data analytics. The general public needs knowledge to be intelligible to new approaches, with an explanation of new concepts and technicalities readily available.

3.1.2 Use Cases

We distinguish the following use cases, the list of which is summarized in Table 3.2. These are identified based on the stakeholders depicted and summarized in Table 3.1. This is essential for further investigation of the usefulness and reproducibility of the research efforts on LLM-driven data analytics.

Tag	Use Case
U1	Time-efficient and Large-scale Data Visualization
U2	Fully-automated Data Visualization
U3	System Extension and Improvement

Table 3.2: Use Cases Summary.

(U1) Time-efficient and Large-scale Data Visualization should be accessible by enterprise employees of various expertise backgrounds, both non-technical **(S1)** and technical **(S2)** who are enabled to derive analysis from time-efficient data visualizations based on large-scale files. This is substantial for solving bottlenecks mentioned in Section 1.1, which describe the inefficiencies caused by time constraints to create data visuals. Furthermore, the beneficial elements of this system are the individuals and units that make decisions **(S3)**, whose actions can be based on reliable insights generated without a possible misuse of BI tools. LLM-driven solutions for business workflows are currently a highly researched subject; therefore, researchers **(S4)** should have access to reproducible documentation of this thesis.

(U2) Fully-automated Data Visualization should allow specifically non-technical enterprise employees **(S1)** and decision-making units **(S3)** to analyze large-scale data in an independent manner. This could solve the bottleneck of dependencies on the technical divisions and allow fully-automatic data visualizations using free-text. This solution could also be beneficial for researchers **(S4)** who specialize in prompt engineering and visualization efforts, students **(S5)** and the general public **(S6)**, who could create customized visualizations based on their personal needs.

(U3) System Extension and Improvement Researchers **(S5)** and students **(S6)** should be able to fully dissect and extend the basic capabilities of the system to enhance the vi-

sual generation capabilities of the system. This could include changes that are raging from those concerning user interface and experience design to testing other prompt engineering techniques other than those mentioned in Section 2.4.2. This is essential as the research towards LLM-based solutions is prominent nowadays and is significantly evolving in enterprise adoption and research capabilities.

3.1.3 Functional Requirements

We distinguish the following Functional Requirements (FR), the list of which is summarized in Table 3.3. These are identified based on the stakeholders depicted in Table 3.2. Based on the identified envisioned users and their consequential user cases, we can establish the core design requirements of the DataViz system for large-scale data visualization. We identify the Main Functional Requirement (MFR) in the following manner:
MFR: Accurate data visualization generation from natural language prompts and CSV type datasets, without the requirement of technical expertise.

Tag	Requirement	Category
FR1	Prompt Alignment with Data Visual Intent	Accuracy
FR2	Correct Utilization of Dataset sample	Accuracy
FR3	Fully-automated Natural Language Interface	Usability
FR4	Secure Data Handling	Usability
FR5	Error Detection and Handling	Usability

Table 3.3: Summary of Functional Requirements of the DataViz system.

(FR1) Prompt Alignment with Data Visual Intent is necessary to obtain accurate visual insights of the generated data based on the prompt provided by the user, so that the created image is tailored to the personal needs of the user and the request. The system shall handle ambiguous and misspelled prompts.

(FR2) Correct Utilization of Dataset Sample stands for correct extraction of column names in the data file provided by the system user. This is essential for accurate code generation and execution for accurate data insight, which is aligned with the file.

(FR3) Fully-automated Natural Language Interface should enable the user to navigate the user interface and be easy to operate with. It should convey a prompt field for the user to submit to the LLM, as well as other input needed to generate a visualization aligned with the analysis intention. The interface should not be constrained by any column

disclosures or modular pipelines to ensure non-technical users could freely generate data visualizations.

(FR4) Secure Data Handling stands for the system's ability to process the business enterprise, sensitive data in a way that aims to maximize security and minimize privacy breaches.

(FR5) Error Detection and Handling stands for the system's ability to notice and inform the user about the erroneous behavior both by the system and by the LLM. This can be caused by misalignment of the user prompt and the provided dataset or any occurrence issues with data visualization script execution.

3.1.4 Non-Functional Requirements

In addition to the functional requirements of Section 3.1.3, we determine five Non-Functional Requirements (NFR) for the `DataViz` system for large-scale data visualization. We summarize the list in Table 3.4.

Tag	Requirement	Category
NFR1	LLM-driven Visualization Accuracy	Accuracy
NFR2	Low-latency API Responses	Performance
NFR3	Large-scale Data Processing	Performance
NFR4	Efficient Code Execution	Performance
NFR5	Reproducibility	Usability

Table 3.4: Summary of Non-Functional Requirements of the `DataViz` system.

(NFR1) LLM-driven Visualization Accuracy is substantial for the correct execution of the script. This requirement is necessary to validate that the LLM response is aligned not only with the prompt and the provided dataset, but also visually and semantically accurate to the visualization intent of the user.

(NFR2) Low-latency API Responses is aimed at time-efficiency value of the system. This requirement is needed to ensure the errors connected to context window and the maximal number of tokens needed for input and output chat completion. This limitation should be targeted by the system by proposing a lightweight solution.

(NFR3) Large-scale Data Processing is crucial to the performance of the proposed system. Based on varying dataset sizes, the system should process the visualization script in a time-efficient manner.

(NFR4) Efficient Code Execution is needed for large-scale execution of the LLM-generated visualization script. This requirement is motivated by the need for an efficient and effective solution for processing large data files and code execution in a timely manner.

(NFR5) Reproducibility is needed for the visualization system to be well documented to ensure reproducibility of its design, implementation, and evaluation. The system should be structured and allow for understanding of the open-source community [69].

3.2 Design Overview of DataViz

The high-level architecture overview of the DataViz for large-scale data visualization through prompt engineering is shown in Figure 3.1. The system consists of an NLI, which enables user interaction through user free-text input, explained in detail in Section 3.3.1. The interface is integrated with the data visualization pipeline, which serves as the core data processing, as well as the code execution engine. Both of these components form a system that allows natural language processing to visualization-first data insights, ensuring an intuitive user interface and experience by incorporation of error handling mechanisms.

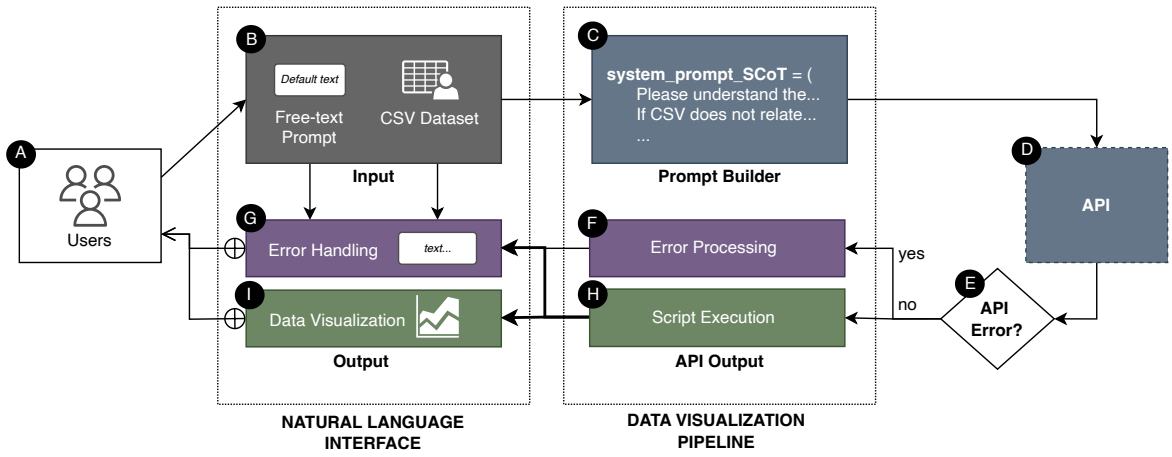


Figure 3.1: Conceptual Design Overview of the DataViz system.

The conceptual design begins with the inclusion of user interaction (A) with the core components of the NLI. The input modalities are required to generate a data insight, a free-text NL prompt, and a CSV Dataset (B) **(FR3)**. If either of the required inputs is missing, the system should inform the user by displaying an error message (G) **(FR5)**. When the required input modalities are provided, they are pre-processed and included in the prompt builder (C) **(FR1)** **(FR2)**, which is sent to the OpenAI API. By the instructions provided

by the NLQ, the LLM shall return either an error message which contains information on why the visualization cannot be generated or a Python data visualization script. In the former case, when LLM detects a message (F) it is displayed to the NLI (G). In the latter case, when a dataset error is not raised by an LLM, the script is executed locally (H) and the image dynamically displayed to the user (I).

3.3 Design of Core Modules in DataViz

Based on the understanding of DataViz provided by the design overview in Section 3.2, we provide an in-depth analysis of core system components. We describe the design process and motivation for the choices made behind the NLI in Section 3.3.1, error handling in Section 3.3.2, prompt engineering (Section 3.3.3) and data visualization pipeline in Section 3.3.4; These are necessary for a structured documentation and reproducibility (**RC4**).

3.3.1 Natural Language Interface

The NLI design process involved multiple iterations, evolving from simplified digital prototypes created using Figma software to functional prototypes [18]. Each redesign was carried out according to the feedback generated from the stakeholders mentioned in Table 3.1. We present the final DataViz NLI, which went through several design phases, from digital sketches to code implementations. The final graphical user interface shown in Figure 4.2, which includes applied feedback on the redesign gained after the user study (Section 5.1.5). We wanted the interface to be minimalist, yet accessible and in accordance to HCI principles, hence the high color contrast [15, 42]. The letter annotations in Figure 4.2 relate to those in Figure 3.1.

The code modules required for user input include (B), which are the upload of the data set, and a concise preview of the data, as well as the user prompt input field. Both are necessary for the successful output of the data visualization, which is related to both of these variables. An exemplary successful data visualization output in the NLI is provided in the figure by the (I) tag. Furthermore, the interface includes additional functionalities in addition to the core ones mentioned to ensure an interactive user interface. The uploaded files are stored during a session in *Recent Files* for easier accessibility of different datasets, as well as the possibility to refresh the short-term storage, *Clear Recent Files* (a). The user can also decide to download the data visualization image to their local machine, as well as to remove the picture from the display in *More Actions* (b). The choice of which LLM model is used is also available for the user to manipulate with (c), so that they can

alternate between varying levels of text comprehension and token usage. In the footer of the sidebar project documentation is included as well as visualization tool instructions. The choice of the LLM model that is used is also available for the user to manipulate with ⓓ. The former opens a new window with the pdf documentation of the system, and the latte displays an introductory message to the user about the usage of DataViz.

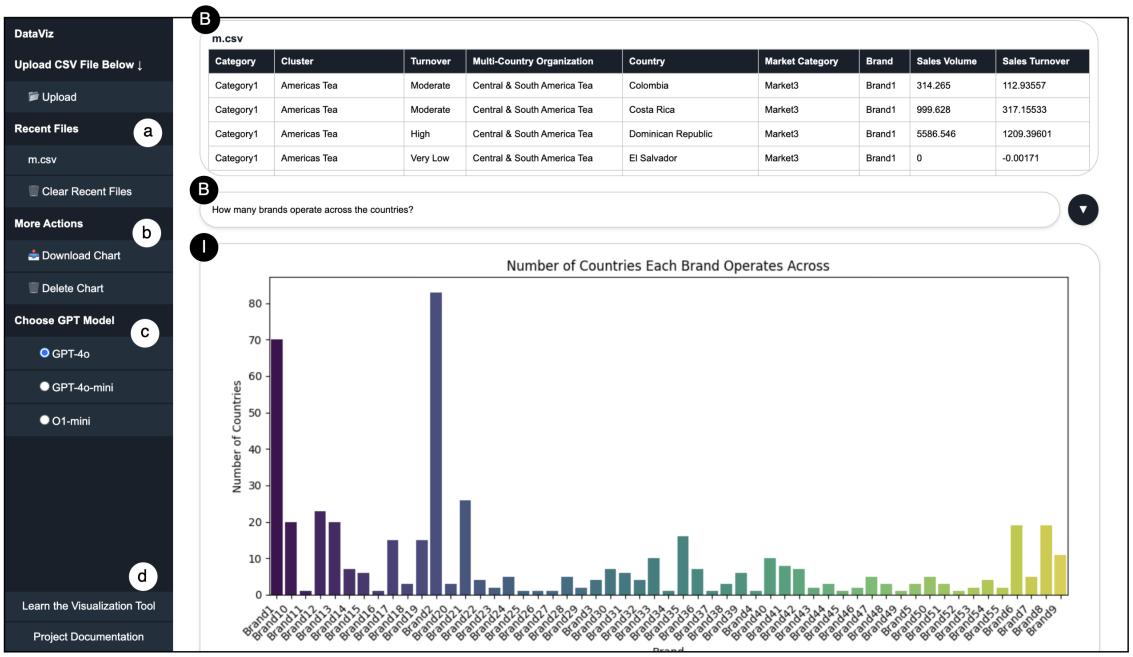


Figure 3.2: DataViz snapshot displaying a data visualization from NLQ, *How many brands operate across the countries?*.

3.3.2 Error Handling

The NLI is closely integrated with error handling to facilitate user navigation through the DataViz system [46]. The NLI snapshot that displays an error message is shown in Figure 3.3, which is consequent for each error case. The error messages can be organized into three error categories, based on Figure 3.1 tags: *Incomplete input error* Ⓑ, *LLM-facilitated error* Ⓕ and *Script execution error* Ⓗ. All of which are shown in Figures 3.4 3.5 3.6, respectively.

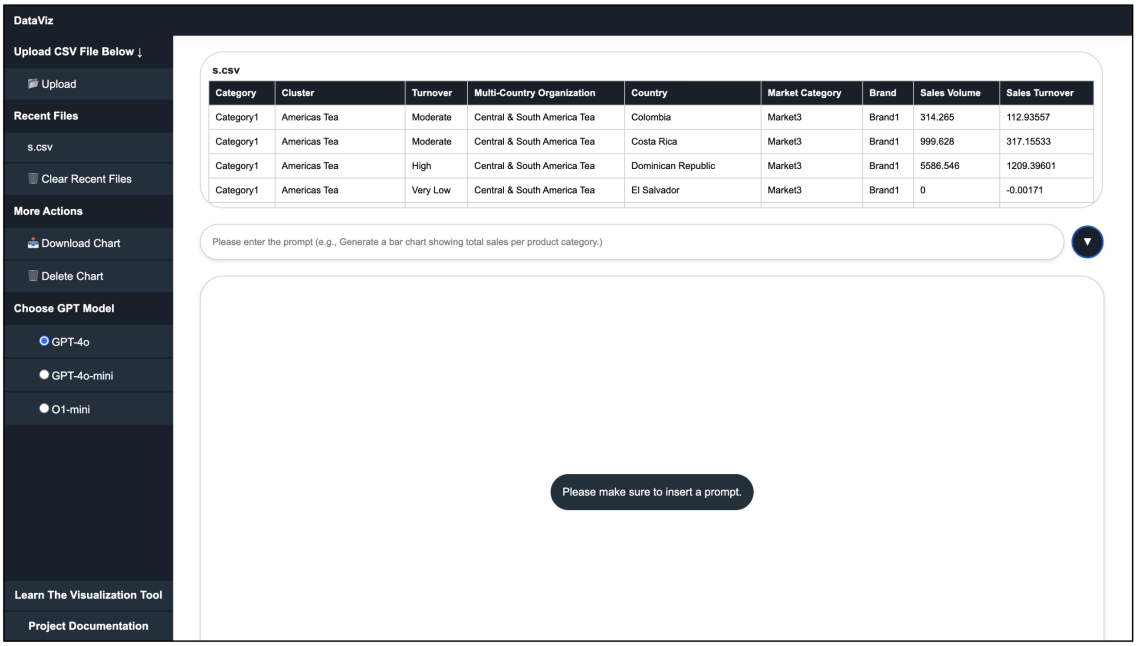


Figure 3.3: NLI displaying an Error Message.

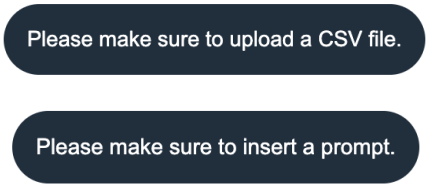


Figure 3.4: NLI showing the *Incomplete input error*.

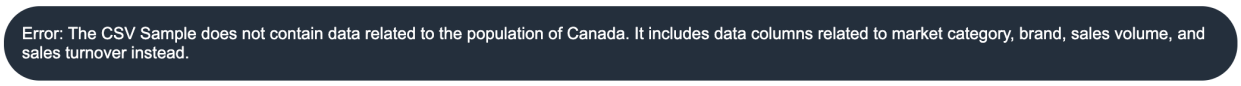


Figure 3.5: NLI showing the *LLM-facilitated error*.

Figure 3.6: NLI showing the *Script Execution Error*.

3.3.3 Prompt Engineering

Based on the systematic review conducted in the literature (Section 2.3), we implement the novel Structured Chain-of-Thought prompting approach in the design of DataViz [28]. By using such a method we want to ensure the accuracy of the generated data visualization script by a structured, defined approach to limit the possibility of LLM's hallucination and handle ambiguity **(NFR1) (FR1) (FR2)**. The *Prompt Builder* from Figure 3.1 is structurally decomposed and shown in Figure 3.7.

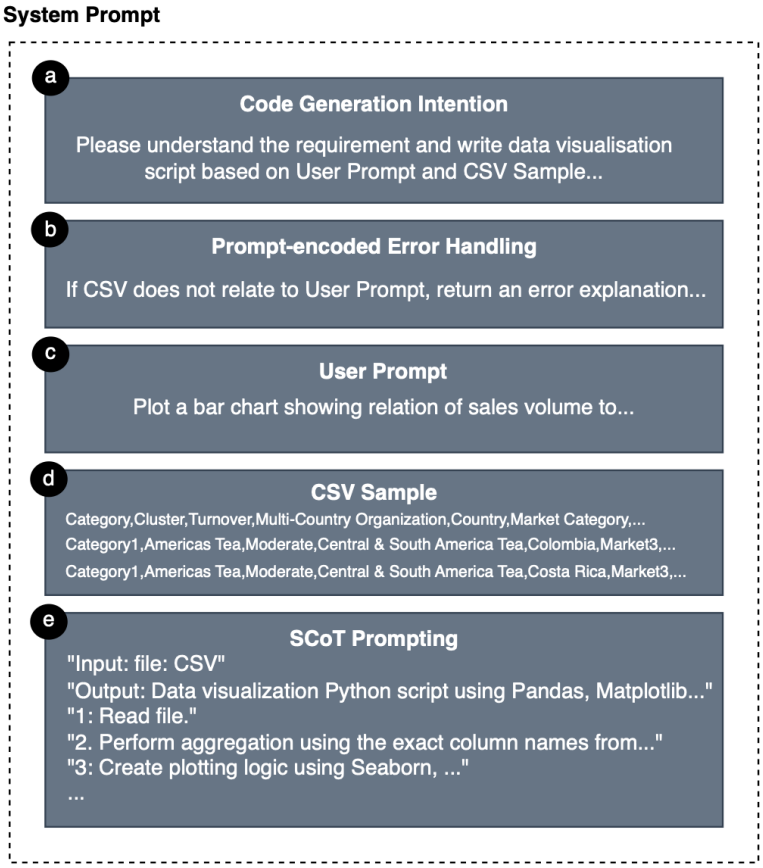


Figure 3.7: Structured Chain-of-Thought Prompting.

Firstly, the LLM is informed and the script generation intent is disclosed (a), where the model learns that it will be writing a data visualization script based on the user prompt and dataset. Then, the prompt-encoded, *LLM-facilitated error* (Figure 3.5) is handled in the instructions (b). The user input, the prompt (c) and the dataset sample (headers with two exemplary rows) (d) are provided to the LLM as context. This is necessary for domain-specific and context-aware data insights, not to produce extensive API calls,

which are prone to timeouts and token use **(NFR2)**. This approach allows for large file processing, since the model is instructed to create plotting logic, knowing that the data set file is already provided locally, using specific columns and their respective data types instead of raw data **(NFR3)**. Therefore, data visualization script execution is done locally using local architecture, minimizing token usage for output generation and the possibility of API timeout errors **(NFR5)**. Moreover, by not including the full dataset in the API call, we minimize the security risks of a sensitive data publication. Finally, we include structured code generation rules for general-use Python visualization script ⓔ.

3.3.4 Data Visualization Pipeline

The simplified data visualization flow is shown in Figure 3.8 and is based on the work done by *Li. et al.* [29]. It provides a concise overview of the visualization pipeline from a NL question ❶ to a data visualization image ❹. The intermediate steps include the system prompt building ❷, which is explained in detail in Section 3.3.3 and a data visualization script returned by the LLM ❸. An image generated can be used for effective decision-making processes and dashboard creation due to download functionality. When desigining the final version of DataViz we prioritize simplicity of the offered system, which allows efficient and simplified script execution.

The choice of the particular API was influenced by interviews and peer-review sessions with stakeholders, who were primarily interested in a visualization-first system, rather than training and fine-tuning an LLM. The use of OpenAI APIs is accepted in large-scale companies for app development, such as Unilever, but it needs to pass authorization checks to be accepted for production in DataLab [2, 49, 74]. However, due to the possible fragility of an external API call, sending sensitive data could result in sensitive data breaches and calls for security enhancement research [2, 50]. Currently, the system allows API usage of the following models: GPT-4o, GPT-4o-mini and O1-mini [49].

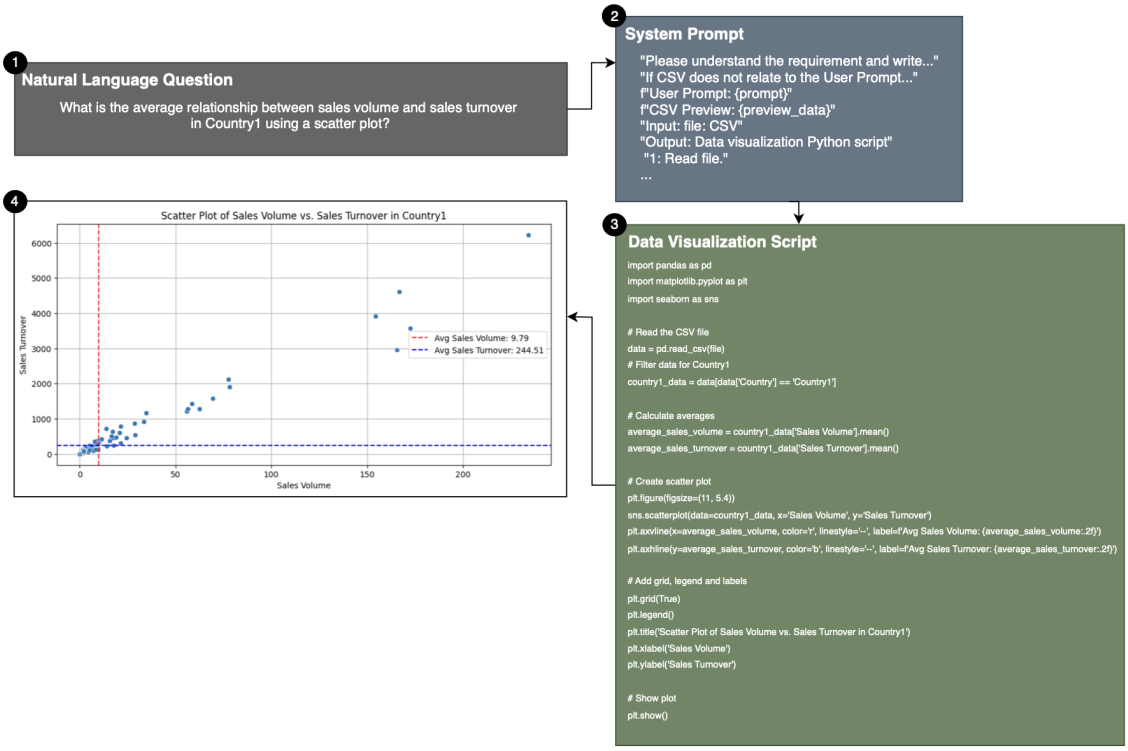


Figure 3.8: Pipeline of LLM-driven Tabular Data Visualization, based on [76].

3.4 Design of Evaluation Metrics

The evaluation metrics for DataViz are presented in the following section. Accordingly to the systematic review of the literature (Section 2.5), we acknowledge the gap in research for structured accuracy assessment approaches of LLM-driven data scripts and visualizations. According to *Li et al.* research on code generation evaluated using text similarity-based metrics, such as BLEU, shows poor performance in measuring [8, 28, 52]. Therefore, we choose not to include this metric in our system's evaluation. We organize the metrics into evaluation categories: accuracy, performance, and usability. For accuracy, we deploy the human evaluation metric and Visualization Error Rate (VER). For performance, we measure latency for visualization script execution and input and output token usage. For usability study, we gather user feedback using a Likert scale.

3.4.1 Human Evaluation

We identify a human evaluation metric is sufficient for the start stage of DataViz development. *Dibia* proposes a summary of the evaluation dimensions, but for the sake of

experiments, we restructure it so that the experiments will rely on answers *Correct* (positive result) or *Incorrect* (negative result) for a more straightforward interpretation of the results [8, 14]. This metric allows for a more structured and comprehensive approach to evaluate the quality of data insight using six diverse questions.

Dimension	Prompt
Code accuracy	Is the visualization script free from bugs, logic errors, syntax errors or typos?
Data transformation	Is the data transformed and appropriately used from the dataset Sample for the visualization type?
Goal compliance	Does the script meet the specified visualization goals?
Visualization type	Considering best practices, is the visualization type appropriate for the data and intent?
Data encoding	Is the data encoded appropriately for the plot type?
Aesthetics	Are the aesthetics of the visualization appropriate and effective for the visualization type and the data?

Table 3.5: Summary of the evaluation dimensions and the corresponding question sketches, based on [14].

3.4.2 Visualization Error Rate (VER)

VER is computed as a percentage of created data visualizations that result in code compilation errors [13]. This metric allows us to assess the precision of script execution and critical insight into the reliability of `DataViz` outputs and how prompt-engineering changes can impact and affect the system. In the mathematical formula below, **E** stands for the number of scripts generated with code compilation errors and **T** for the total count of the visualizations generated [13].

$$\text{VER} = \frac{E}{T} \times 100 \tag{3.1}$$

3.4.3 Token Usage

The *inference speed* refers to the actual rate at which the LLM processes tokens, and is often measured in TPM (tokens per minute) or TPS (tokens per second) [48]. To ensure time-efficient data visualization cutting it is said that 50% of the output tokens may cut

around 50% of the latency needed for the API call completion [48]. Therefore, when evaluating DataViz performance we measure the amount of input and output tokens per each call. We do this by inclusion of `completion.usage.prompt_tokens` for measuring the count of the input tokens and `completion.usage.completion_tokens` for measuring the count of the output tokens.

3.4.4 Latency Measurement

To evaluate the system's performance across varying-scale datasets we measure the latency, time (MS) needed for data visualization script execution. We apply this metric by including timestamps before and after the script execution function and we round up the values to 4 places after the decimal to ensure precision. This is crucial for DataViz documentation and sets the ground for future work.

3.4.5 Likert Scale

The original Likert scale is a set of statements (items) offered for a real or hypothetical situation under study. Participants are asked to show their level of agreement (from strongly disagree to strongly agree) with the given statement (items) on a metric scale. Here, all the statements in combination reveal the specific dimension of the attitude towards the issue, hence, necessarily inter-linked with each other [24].

3.5 Summary of the DataViz Design

In this chapter, we addressed **RQ2** and designed an LLM-driven system for large-scale business data visualization. We summarized 6 key stakeholders, 3 use cases, and defined 5 functional and 5 non-functional requirements, and consequently addressed them in the system design (**RC2**). The summary of key work is listed below. The design limitations are discussed in Section 6.1.

F3.1 A comprehensive analysis of the stakeholders, use cases and functional and non-functional requirements of the DataViz design was conducted and documented for reproducibility and comprehension of the topic.

F3.2 The design reiterations and redesigns based on stakeholder feedback and literature review resulted in code-implied NLI of DataViz.

F3.3 A responsive NLI enhanced by error handling and website functionalities improves navigation and interaction.

F3.4 The designed data visualization pipeline in its simplified form is sufficient for the visualization-first system and its further evaluation of its usefulness in an enterprise setting.

F3.5 The metrics used for evaluation of LLM-driven data visualization system is limited but is sufficient to assess accuracy of the charts.

F3.6 Multiple design limitations were identified and consequent future work was proposed.

4

Implementation of DataViz

For addressing **RQ3**, this chapter implements a system for large-scale business data visualization (**RC3**). We begin by explaining the implementation choices behind the core components, the programming languages, libraries, and tools in Section 4.1; Furthermore, we motivate the in-detail choices behind the dynamic interface, and the Flask backend; The summary of the key findings of the implementation is included in Section 4.4 and limitations in Section 6.2.

4.1 Implementation Overview

The DataViz implementation is avaliable publicly on GitHub[1]. According to the design requirements outlined in Section 3.1 and the design overview, the DataViz system was implemented to support large-scale data visualization through a full-stack web architecture. The core components of the system and the data flow between the modules are summarized in Figure 4.1. The motivation behind using Flask was due to its compatibility with enterprise workflows, such as DataLab [74]. In GitHub we include Pipfile.lock to ensure exact dependency versions [2].

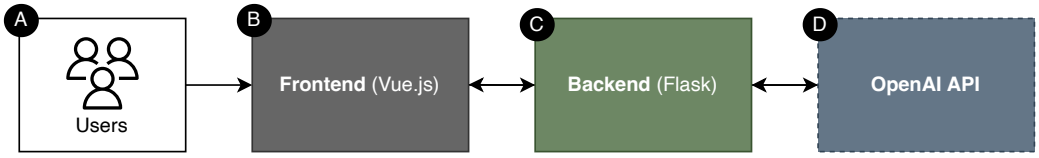


Figure 4.1: High-level Overview of DataViz implementation.

[1] https://github.com/mmyalen/DataViz
[2] https://github.com/mmyalen/DataViz/blob/main/Pipfile.lock

The user (A) interacts with the dynamic interface (B), which is implemented using the JavaScript framework, Vue.js, which enables the extension of HTML attributes in a way accessible [26, 34, 84]. HTML was used for the website structure and CSS for the styling aspects of the document [6].The backend (C) was built using Flask, a lightweight WSGI-compliant Python framework suitable for the integration of RESTful APIs, which allows the incorporation of LLM capabilities into applications [51]. The code base is implemented in Python 3.11, which choice is motivated by its wide-functionality in data science and analytics, as well as accessibility by the open source community [69]. The choice of languages is rationalized by their interactivity with specific frameworks and simplicity to ensure the reproducibility of the project **(NFR5)**. The backend consists of two files `app.py`, the main file, and `processing.py` for all helper functions used to generate a data visualization [1] [2]. This ensures transparency and aligns with the principles of clean code. The list of packages used by the backend is summarized in Table 4.1. The backend closely interacts with the OpenAI API (D) , which is described in more detail in Section 4.3.

Package	Version	Brief Description
`flask`	3.0.3	Web development framework.
`openai`	1.54.3	Python SDK for OpenAI integration.
`python-dotenv`	1.0.1	Secure usage of environment variables.
`pandas`	2.2.3	Python data processing library.
`matplotlib`	3.9.2	Python plotting library.
`seaborn`	0.13.2	Python plotting library.

Table 4.1: Data dependencies `DataViz` implementation.

4.2 Dynamic Interface

Vue.js was chosen for its widespread adoption among full-stack app developers and its lightweight architecture, which supports dynamic and rapid changes in performant web applications [34] . It allows for reactivity and interaction with the application as a result of its wide list of varying binding methods. In the `DataViz` system, Vue.js methods and varying binding types are used for real-time, conditional responsive output, error handling, and data visualization display **(FR3) (FR5)**. It allows for a dynamic field preview of the

[1] `https://github.com/mmyalen/DataViz/blob/main/app.py`
[2] `https://github.com/mmyalen/DataViz/blob/main/processing.py`

uploaded dataset and choice of available LLMs for data insight generation. This facilitates a seamless user experience by updating the interface without full page reloads, improving usability [1].

Integration of Vue.js with the backend is possible due to the `async plotGen()` method, which automatically fetches the data and awaits their response. Alongside usability features such as a loading icon or error handling if a part of the required is missing, this function comprises the input, user prompt, and dataset, and sends a `POST` request to the `'/response'` Flask endpoint. Based on the specific type of the response, we await either `await response.json()` or `await response.blob()` to display to the user a text message or a PNG image to ensure efficient efficient and accurate feedback.

4.3 Data Processing and Visualization

The backend module, `app.py`, deploys built-in Flask functions, such as: `render_template`, `request`, `jsonify`, and `send_file` to render HTML pages, handle incoming requests, return JSON-formatted text messages, and send a visualization PNG to the frontend, respectively. When a `POST` request is recieved, Flask processes the uploaded CSV file and the NLQ using `request.files` and `request.form`. The data is then read and processed using pandas, so that a data sample is extracted (headers with first two rows) to be provided to the LLM. The LLM receives the data in a CSV format separated by comas without indexing to ensure the accuracy of the generated visualization script and limit the possibility of errors based on the misunderstood dataset structure by the model. Although it serves as a good example for context-aware visualization **(NFR1)** and low-latency API calls **(NFR2)**, it is a static solution. Without sending the complete data file, we omit possible security concerns **(FR4)**. The prompt structure is available on GitHub [2].

Interaction with OpenAI models is possible with a secret API key, which must be securely provided through a `.env` file and accessed through `os.environ` as shown in the code sample below [47]. The Python library `python-dotenv` is essential and is provided in the `Pipfile`[3].We implement the functionality to alternate between varying models: GPT-4o, GPT-4o-mini, and O1-mini. At the time of the research was conducted, these were the most

[1] https://github.com/mmyalen/DataViz/blob/main/templates/base.html
[2] https://github.com/mmyalen/DataViz/blob/main/processing.py
[3] https://github.com/mmyalen/DataViz/blob/main/Pipfile

recent OpenAI publications with low token usage [49]. The selection of models is dynamically determined on the basis of the user's input and provided in the `model` variable. Models such as O1-mini do not have configurable parameters for temperature or token limits, therefore the default values were used [16]. The system prompt, which combines the user prompt **(FR1)**, the dataset sample **(FR2)**, and the structured chain-of-thought structure, is combined as `system_prompt_SCoT`. The resulting output, `chat_completion`, from the OpenAI API, is either a text message which is returned to the NLI or a Python visualization script, which is checked for any GPT-specific symbols that can fail code execution, for instance the triple backticks `"‵‵‵"` common for the API's code generation practices. The count of input and output tokens can be detected using `completion.usage.prompt_tokens` and `completion.usage.completion_tokens`, respectively [2, 49, 62].

```
client = openai.OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
completion = client.chat.completions.create(
    model = model,
    messages = [
        {"role": "user", "content": system_prompt_SCoT}
    ]
)
chat_completion = completion.choices[0].message.content
```

Figure 4.2: Chat Completion for OpenAI API, based on [49].

All scripts returned by the LLM are written in Python and rely on standard Python libraries for their functionality. Efficient execution **(NFR4)**, specifically for large files, is achieved by script execution on locally stored data files , ensuring its security and low-latency API connection [49]. For visualization, the Anti-Grain Geometry (AGG) backend, used by Matplotlib, is essential; it enables the rendering of static PNG images [38]. We opted for using Pandas, Matplotlib and Seaborn libraries for visualization generation due to its wide use among the data science community. We considered using Plotly, yet the aesthetics of the charts with the libraries mentioned above were more suitable for the initial stage of the `DataViz` system [54]. The LLM-driven script is executed using the Python build-in `exec()` function according to *Ramos* approach, which creates an execution environment with safe global and configuration dictionary for the plotting data processing functions [55]. The image is sent in PNG format to the NLI using the `send_file()` flask function, ensuring it fits in the visualization fields. This enables in-memory image transmission without writing to disk, without comprising performance or memory usage. Processing of the script in a visualization was done in a PNG format rather than as a

SVG (Scalable Vector Graphics). This was motivated by the fact that PNGs allow efficient and consistent rendering across various browsers and platforms, making them particularly suitable for detailed images [1]. Figures 4.3 and 4.2 show the system with exemplary data visualizations displayed.

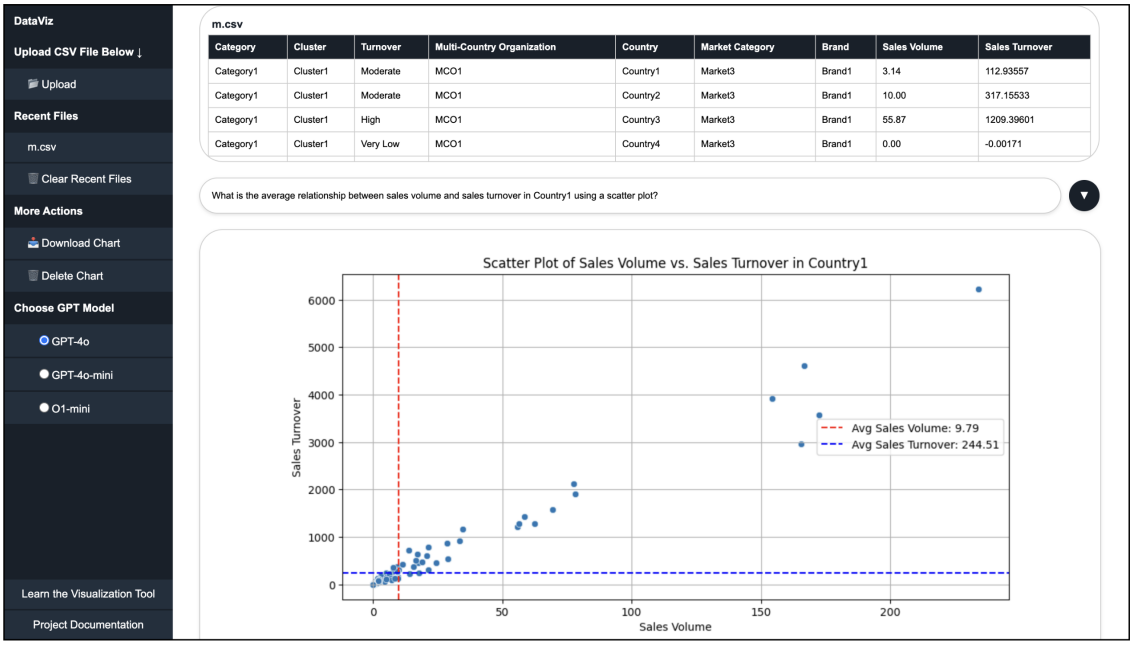


Figure 4.3: DataViz snapshot displaying a data visualization from NLQ, *What is the average relationship between sales volume and sales turnover in Country1 using a scatter plot?*.

4.4 Summary of the DataViz Implementation

In this chapter, we addressed **RQ3** and implemented an LLM-driven system for large-scale business data visualization. We analyze and motivate the core implementation choices and provide an in-depth explanation of the NLI and the data processing and visualization pipeline (**RC3**). The summary of key work is listed below. The implementation limitations are discussed in Section 6.2.

F4.1 A fully-automated, visualization-first web application driven by OpenAI LLMs can be effectively implemented using a lightweight architecture.

F4.2 Resource-efficient prompting, which combines the SCoT structure, the NLQ, and the sampling of the dataset, enables the generation of secure and low-cost visualizations, with the aim of maintaining consistent API latency and minimizing token usage.

F4.3 Dataset sampling combined with in-memory execution enables memory-efficient processing and large-scale data file handling without disk storage.

F4.4 The dynamic, responsive NLI, enhances user experience and ensures reproducibility through a simplified fronted framework.

F4.5 Time-efficient data visualization is achievable, allowing stakeholders to gain understanding of the provided datasets and generate data insights efficiently and effectively.

F4.6 Multiple implementation limitations were identified and consequent future work was proposed.

5

Evaluation of DataViz

For addressing **RQ4**, this chapter conducts an evaluation of DataViz, a system designed to enable AI-driven data insights. The evaluation aims to validate the code implementation, explained in Chapter 4; This chapter begins with an explanation of the experimental setup, which is followed by the preparation of the case study dataset and prompt engineering; This is necessary for reproducible evaluation and enterprise data privacy; We ensure a comprehensive assessment by structuring the evaluation among key dimensions: accuracy, performance, and usability, which all reflect the indispensable aspects of the system: HCI, NLP, and data visualization; The summary of the key findings of the evaluation is included in Section 5.5 and limitations in Section 6.3.

5.1 Experiment Setup

The DataViz experiments and materials are available publicly on GitHub[1]. All experiments described in the following chapter were conducted on a local server. The technical and hardware specifications are detailed in Table 5.1. To ensure a structured, reproducible and comparable experimental setup, we first prepared the user case datasets (Section 5.1.1) and the corresponding prompts (Section 5.1.2) to test for a variety of NLQs. This setup facilitates structured evaluation, where design choices are motivated by research and enterprise requirements. In the accuracy and performance studies we only use the GPT-4o model, while for the usability study we used a selection of differing available models.

[1] https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz

Component	Specification
Model	MacBook Pro (13-inch, M2, 2022)
Chip	Apple M2 (8-core CPU, 10-core GPU)
Memory (RAM)	16 GB Unified Memory
Storage	494.38 GB SSD
Display	13.3-inch Retina (2560 × 1600)

Table 5.1: Technical Specifications of the Experiment Infrastructure.

5.1.1 Dataset Preparation

The prompt preparation (Section 5.1.2) is based on a dataset derived from Unilever as case study data during the course of internship research. This enhances the evaluation by including exemplary real-world company data, which contains domain-specific dependencies, structured relationships, varying data types, and details, which synthetic datasets could not provide. However, to address privacy concerns and avoid infringements of corporate confidentiality, all data were thoroughly anonymized prior to use (Figure 5.1). The data from the case study allow the system to generate data visualization, which serves as a simplified abstraction of real-world business data. The original dataset took approximately 2.2 MB of storage and included the following data types: **Categorical (C)** and **Quantitative (Q)**. To ensure that the core data types were covered in the evaluation, a column was added, *Turnover* based on the *Sales Turnover* column to ensure incorporation of **Ordinal (O)** data, which preserves the dependencies of the dataset [56]. The diverse data types and columns of related datasets shown in Figure 5.1 are summarized in an overview table 5.2. The anonymized case study data are publicly available on GitHub [1].

Category	Cluster	Turnover	Multi-Country Organization	Country	Market Category	Brand	Sales Volume	Sales Turnover
Category1	Cluster1	Moderate	MCO1	Country1	Market3	Brand1	3.14	112.93557
Category1	Cluster1	Moderate	MCO1	Country2	Market3	Brand1	10.00	317.15533
Category1	Cluster1	High	MCO1	Country3	Market3	Brand1	55.87	1209.39601

Figure 5.1: Case Study Dataset Sample.

[1] https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz/tree/main/case_study_data.csv

Data Type	Related Column from Dataset Sample
C	Category, Cluster, Multi-Country Organization, Country, Market Category, Brand
O	Turnover
Q	Sales Volume, Sales Turnover

Table 5.2: Data Types and Related Columns from the Dataset Sample.

5.1.2 Prompt Engineering

The structured literature review (Section 2.4) allowed investigation of varying exploration methodologies used in LLM-driven visualization-first systems. *Liu et al.* propose a structured approach to encode data attributes according to data and visualization plot types (Table 5.3) and related question types [30].

Aggregation	C	O	Q	Visualization	Annotation
None	1	0	1	Bar Chart	
	0	1	1	Line Chart	Highlight
	0	0	≥1	Scatter Plot	
Average	1	0	1	Bar Chart	
	0	1	1	Line Chart	Highlight + Line
	0	0	2	Scatter Plot	
Extreme	1	0	1	Bar Chart	
	0	1	1	Line Chart	Highlight + Line
	0	0	2	Scatter Plot	
Count	1–2	0	0	Bar Chart	Highlight
	0	1	0	Line Chart	
Sum	1–2	0	1	Bar Chart	Highlight
	0	1	1	Line Chart	

Table 5.3: Rules for Encoding Data Attributes (C for Categorical, O for Ordinal, and Q for Quantitative), reproduced from [30]

They diverse two categories of questions: *exploration questions*, e.g. What is the relationship of oil and gas?, and *reasoning questions*, e.g. What is the total gold medals won by the top3? [30]. This further allows for verification of system capabilities across varying data-type compositions, plot and question types, and aggregations. Due to time constraints and the vast nature of the study in a rapidly developing field, where there

are little systematic evaluation approaches, we primarily focus on the *reasoning questions*. Such question types allow for more systematic assessment of system's abilities to handle diverse data aggregations. The specific NLQs used for the accuracy (Section 5.1.3) and performance (Section 5.1.4) evaluations are shown in Table 5.4 and are based on the questions proposed by *Liu et al.* [30]. Due to the fact that we simultaneously want to test for the abilities of DataViz to create diverse plot types (bar chart, scatter plot, line chart), we specify in the prompts which plot due to the fact that such data could be visualized also using bar charts [20]. In contrast, the prompts used in the user study differ as they are designed to assess system usability.

Aggregation	Data Types	Plot Type	Question
Average	2Q	Scatter Plot	What is the average relationship between sales volume and sales turnover in Country1 using a scatter plot?
Extreme	1C+1Q	Bar Chart	Which country has the highest sales turnover in MCO1 Multi-Country Organization?
Count	1–2C	Bar Chart	How many brands operate across the countries?
Sum	1O+1Q	Line Chart	What is the total sales volume by turnover level for each market using a line chart?

Table 5.4: Proposed Reasoning Questions for DataViz Evaluation, based on [30]

5.1.3 Accuracy Evaluation

Based on the prompts listed in Table 5.4, we design an accuracy evaluation of the DataViz system. To assess data visualization and script accuracy, we introduce a human evaluation metric based on the current literature (Table 3.5 in Section 3.4) [14]. This evaluation is carried out in parallel with the performance study described in Section 5.1.4. Due to time and resource constraints, the human evaluation will be done by the researcher instead of data analytics experts. Although this poses a limitation to the accuracy evaluation, the proposed metric acts as a reliable assessment tool. Alongside human evaluation, we will note script execution failures to apply the VER metric explained in Section 3.4 to further test system accuracy. Therefore, the accuracy findings will include data from all the experiment runs (successful and unsuccessful) with a minimum of 10 successful data

visualization outputs per datasize and NLQ, so that we can capture variability in results and allow comparability of the findings.

5.1.4 Performance Evaluation

The performance evaluation of the `DataViz` system is conducted simultaneously with the accuracy study. To simulate varying workloads, we scale the case study dataset to range from 10 KB to 10 GB using functions included in GitHub [1]. All data sizes used and their labels are shown in Table 5.5 for reproducibility efforts. We will test for latency, time needed for the system to execute the LLM-generated data visualization script to document the system's capabilities and to mark possible limitations. We capture the time needed for the OpenAI API call, yet we do not discuss it further, as it is primarily dependent on the provider server infrastructure and the network latency [49]. Due to the prompt engineering approach described in Section 2.4.2, the structure of all system prompts is the same, despite varying dataset sizes. The performance findings will include a maximum of 10 successful runs of the experiment per each datasize and NLQ for consistency of the results.

5.1.5 Usability Evaluation

The purpose of this study is to evaluate the usability of the `DataViz` system through task completion and feedback exchange with study participants. This study aims to evaluate the user experience and the possible integration of LLM into data analytics practices within enterprise workflows. This study will fundamentally focus on usability features of the system and display of available features in the system. The users will be informed that `DataViz` is not yet a tool ready for production and is aimed solely at accessing integration of LLM-driven solutions and data analytics. The user study will be followed by a reiteration of the design of the system. All materials used are available on GitHub in the *usability_study* folder for experiment reproducibility[2].

[1] `https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz/blob/main/dataset_preparation.ipynb`

[2] `https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz/tree/main/usability_study`

Dataset Size	Label
10KB	*XXS*
100KB	*XS*
1MB	*S*
10MB	*M*
100MB	*L*
1GB	*XL*
10GB	*XXL*

Table 5.5: Dataset Sizes and Labels.

The system is evaluated through hands-on experience of business enterprise employees from the Unilever Foods Innovation Center in Wageningen. It assumes participation of 10 workers, five of them showing expertise in data science and analytics, while the remaining half have no further background in these fields. We synthesized that 50% of the participants were in the age range of 25-34 years, 40% in the age range of 35-44 years and 10% in the age range of 45-54 years. At the beginning of the study, we verify the participant's expertise in data science and analytics and familiarity with using LLM-based interfaces.

Dependent variables: user feedback (verbal and non-verbal), user prompt (Task C),
Independent variables: dataset, user prompts and tasks (bar from prompt in Task C), experiment infrastructure, natural language interface, questionnaire evaluation questions

The purpose of the user study is to evaluate the different functionalities of the system by the participants and to obtain feedback from the professionals of the enterprise. We schedule 10 hour-long online sessions with the employees to conduct the study and interact with the researcher machine to solve the following predefined tasks, which will be displayed to the participant in the form of presentation slides. The study will begin with a 5-minute introduction to the participants, who will be informed about the motivation for the research. For tasks A and B, the participant will be asked to navigate the system by strictly following the steps and insert pre-defined prompts. For task C, the user study participant will have the opportunity to explore the dataset in more depth themselves and insert a free-text prompt. This was planned to make sure the participants were actively engaged with the system and to explore possible erroneous areas. If needed, there will be a sheet with curated queries to choose from or use as reference for the ease of use of the participant. After completion of Task A, the user will be asked to answer the first part of

the questionnaire, after Task B, the second part, after Task C, the third and final part of the form. The fourth part will relate to the overall impression of the application and integration of LLM-driven analytics. After each task completion, any additional feedback voiced-out will be written down. The questionnaire after each task will include the parameters shown in Table 5.6 used in a form of a Likert scale evaluation. All tasks are independent of each other, and the application should be refreshed between every task.

Parameter	Description
Ease of Use	The degree to which the user expects the tech system to be free of effort.
Quality	Information, technical, and overall service success.
Functionality	Reflects a correct technical functioning.
Interactivity	Extent to which users can manipulate technology and/or control device; emphasizes the role of interaction between the user and the system.
Enjoyment	Core affect, typically arising from connection or sensory pleasure, interchanged with happiness.
Satisfaction	Occurs when customers find the products or services meet or exceed their positive expectations.

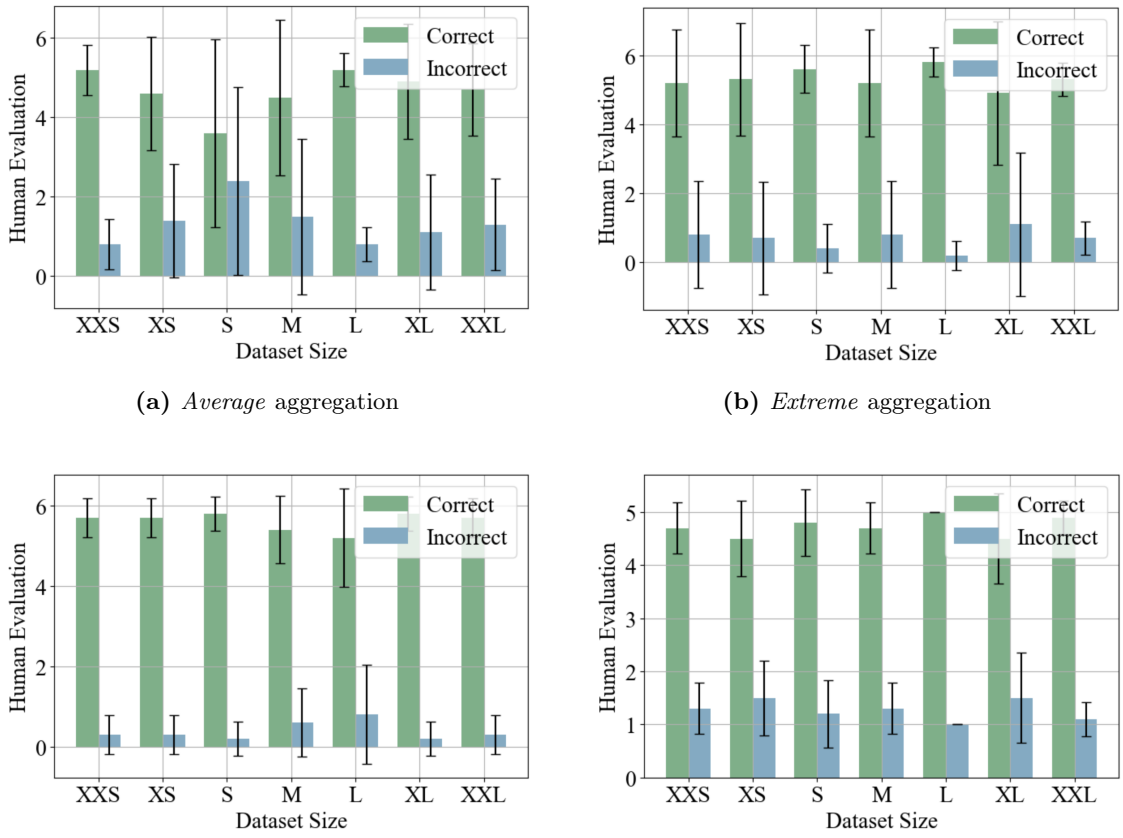
Table 5.6: Core Parameters in Chatbot Efficiency Evaluation, reproduced from [4].

5.2 Evaluation Results of Accuracy

The application of the human evaluation metric produced the results shown in Figure 5.2. We gather scores across different aggregations (*average*, *extreme*, *count*, *sum*) and sorted by increasing dataset sizes (from *XXS* to *XXL*). Each bar chart shows the mean number of correct and incorrect system outputs per prompt, and generally the correct label is significantly more frequent than the incorrect one. For aggregations *Extreme* and *Count* the correct and incorrect results are relatively similar and stable, with a minimal number of incorrect scores. The incorrect scores for the *Average* aggregation are impacted by the incomplete data visualizations generated by the LLM, as it often failed due to the lack of mention or visual highlight of the average values for sales and turnover volumes. The similar reason impacted the error rates in the *extreme* aggregation on a smaller scale. The count of incorrect scores was the highest for the last aggregation, *sum*, as in none of the runs the model did not sort the turnover levels from *very low* to *very high* in the displayed line chart, which affects the readability of the data. This could be influenced by lack of specific clarification in the prompt or/and the fact that the model is never provided with

the whole dataset. Aggregation *count* have the most stable scores according to the size or its error bars, while *average*, have the largest ones. The accuracy improvements of the DataViz system based on these findings stress the importance of prompt clarity when guiding LLMs to generate interpretable data insights. Furthermore, a few-shot learning or conversational mode could improve the intent and detail alignment with the user according to their specific visualization needs.

The results of the VER metric showed that compilation errors were rare, hence the data shown in Table 5.7. The highest compilation error rate was for the aggregation *extreme*, most of the compilation errors were caused by incorrect reading of the file or by the incorporation of column names different from those in the provided dataset sample. This metric reveals that the compilation errors are not significant in the DataViz system, which can assure the structured chain-of-thought improves accuracy of the visualization script.



(a) *Average* aggregation

(b) *Extreme* aggregation

(c) *Count* aggregation

(d) *Sum* aggregation

Figure 5.2: Human Evaluation results for different aggregation strategies.

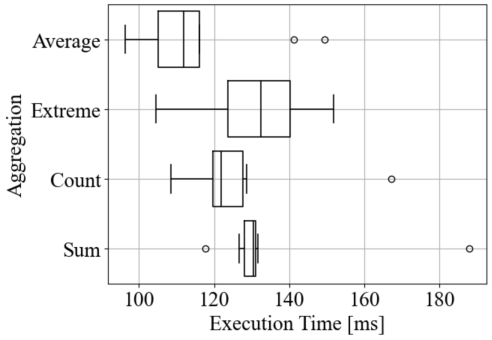
Aggregation	VER [%]
Average	2.7
Extreme	5.4
Count	2.7
Sum	4.1

Table 5.7: Results from visualization error rate (VER) metric, based on [14].

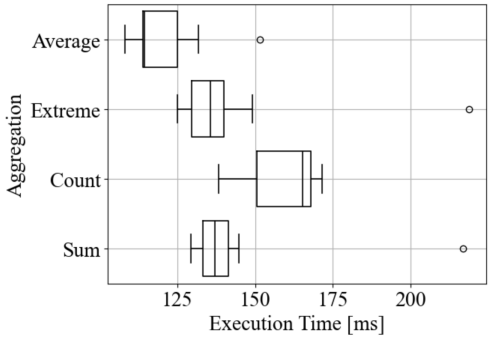
5.3 Evaluation Results of Performance

The performance results are shown in Figure 5.3, which represents the time (ms) for each aggregation type—*average*, *extreme*, *count*, and *sum* across seven dataset sizes, ranging from *XXS* to *XXL*. The latencies, time needed for data visualization generation per dataset size, and aggregation are presented in a form of *modified boxplots* where outliers lie above $Q_3 + 1.5 \times \text{IQR}$ or below $Q_1 - 1.5 \times \text{IQR}$ as individual hollow data points, where Q_1 and Q_3 represent the first and third quartiles, respectively. The interquartile range (IQR) is defined as $Q_3 - Q_1$ [61, 65, 73]. The figure shows the gradual increase in script execution time per dataset size. For the smallest datasizes (*XXS*, *XS*, *S*) the time is very low, where the variance is small and there are a few outliers. For larger dataset sizes (*M*-*XXL*) the median and variance increase significantly, particularly for the *count* and *average* aggregations. This can be influenced by the more comprehensive calculations and operations that need to be performed. This creates execution delays apparent in the *modified boxplots*. The most stable script runtime is for the aggregation *sum*. The wider IQRs and the presence of outliers in the *L*, *XL*, and *XXL* plots further highlight the variability in script performance on a larger scale. This can be influenced by the processing of large-scale files. In general, the performance results reveal that large-scale execution with the use of `DataViz` system is achievable as the execution of the data visualization script for the 10 GB dataset size takes around 1-1.25 minutes.

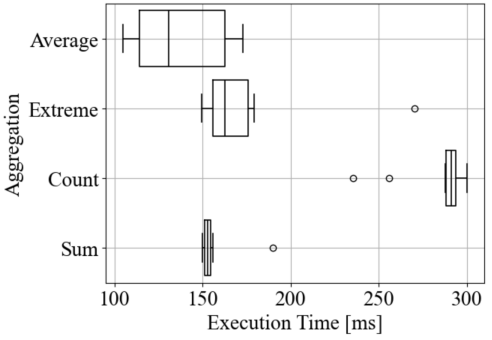
During each run of the experiment, the count of input and output tokens was recorded and presented using Table 5.8 for further evaluation. The token count was rounded up to the nearest whole number. The amount of input tokens did not fluctuate, it remained the same for every call, which signifies a low-latency structured prompt, which minimizes successfully the token usage. In contrast, the output tokens vary within every call, yet the visualizations were nearly identical, which means that the LLM's output flexibility was not compromised.



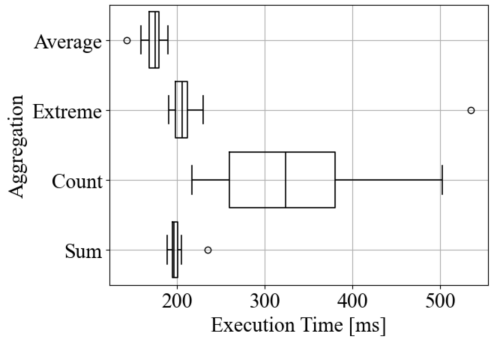
(a) *XXS*



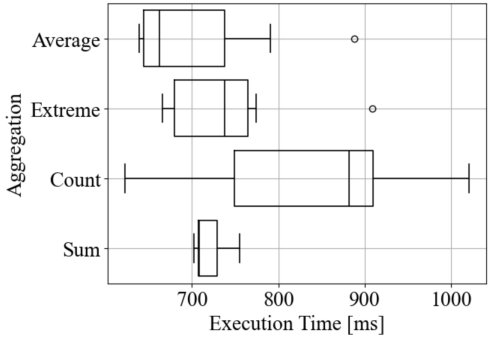
(b) *XS*



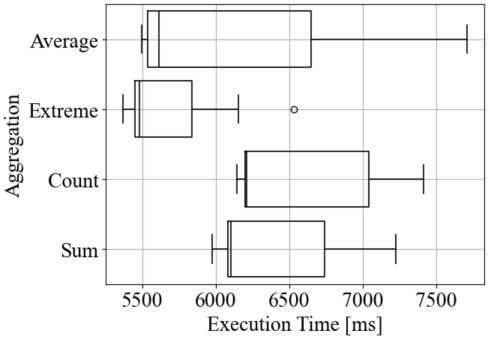
(c) *S*



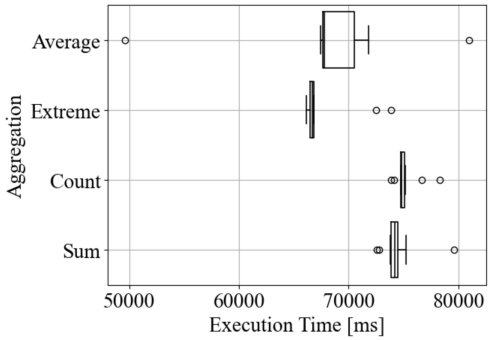
(d) *M*



(e) *L*



(f) *XL*



(g) *XXL*

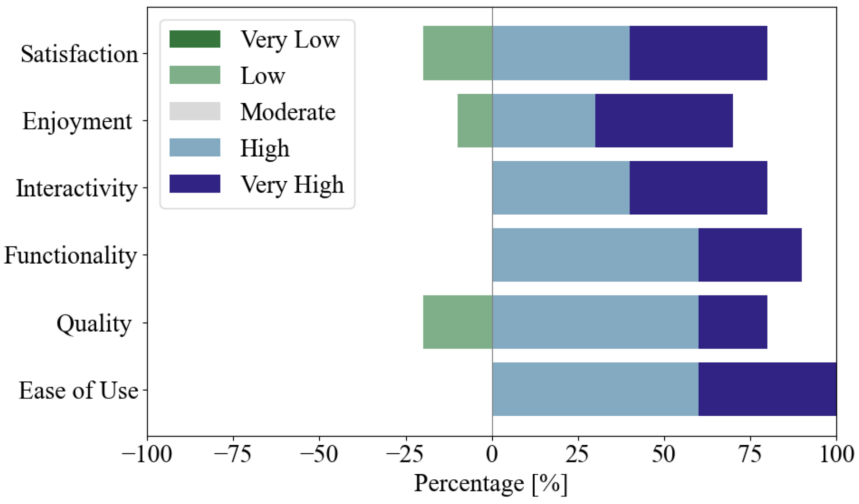
Figure 5.3: Script Runtime Performance Across Dataset Sizes (*XXS* to *XXL*).

Aggregation	Input Tokens	Ouput Tokens
Average	260	166
Extreme	257	200
Count	257	163
Sum	266	170

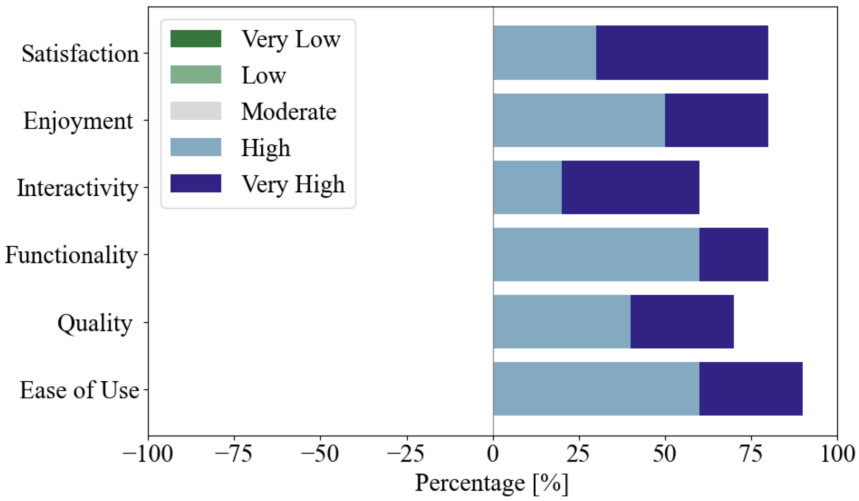
Table 5.8: Token Usage per Aggregation.

5.4 Evaluation Results of Usability

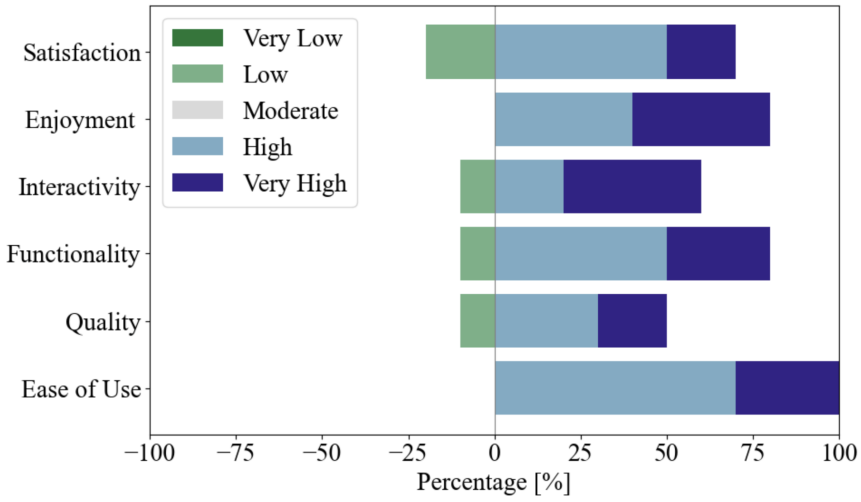
The results of the usability study based on the 5-scale Likert scale are visualized through the *centered stacked bar chart* in Figure 5.4 [53, 67]. Such data display allows us to section the results into positive and negative percentages to notice the overall response patterns in a clear manner. The center line represents the split for the *moderate* rating. In general, all tasks were met with significantly more positive response. However, fluctuations for tasks A and C are visible, specifically for the user satisfaction and quality of the data visualizations. It is worth mentioning that the steps of Task A and Task B were more strict than those of Task C, allowing the participants to interact with the application with their self-curated NLQ. This may be the reason for the more diverse ratings in task C. The voiced positive feedback mentioned a.o. the convenience of timely data visualization generation, ease of use, model diversity, and overall design of the application. The constructive feedback mentioned the data visualization section, which previously was below the data visualization display, making it difficult for participants to notice, lack of plot interactivity, and DataLake connection for easier file upload [12]. The overall positive response was significant among the employees who asked when the system would be in production. The final questionnaire section revealed that 80% of the user study participants were *very likely* and 20% were *somewhat likely* to use such an LLM-driven analytics tool in the future.



(a) Task A



(b) Task B



(c) Task C

Figure 5.4: Usability Study Results for Tasks A-C.

5.5 Summary of the DataViz Evaluation

In this chapter, we addressed **RQ4** and evaluated the LLM-driven system for large-scale business data visualization. We examined the capabilities of the system in key dimensions: accuracy, performance, and usability, to ensure a thorough analysis motivated by research (**RC3**). The summary of key findings is listed below. The evaluation limitations are discussed in Section 6.2.

F5.1 DataViz ensures that LLM-driven fully automated visual analytics systems can be accurate, efficient for large-scale datasets and useful in supporting enterprise data.

F5.2 The accuracy was the highest for *count* and *sum* aggregations, with minimal execution errors. The variability in more complicated aggregations (*average*, *extreme*) results highlights the need for more descriptive prompts or integration of the conversational mode.

F5.3 The lack of complete data provided to an LLM affected the visual aspects of the types of generated graphs, specifically a line chart with an ordered x-axis.

F5.4 The performance of script execution was evaluated in all dataset sizes. For large-scale files of 10 GB, visualization generation remained below 1.25 minutes, with low and stable token usage. The average visualization script execution latency across 10 KB-10 GB files was 11.29 seconds.

F5.5 The user feedback was mostly positive, as participants praised ease of use and timely visualization and provided constructive feedback on interactivity with improvements to data visualizations.

F5.6 Multiple evaluation limitations were identified and consequent future work was proposed.

6

Limitations and Improvements

To improve readability and maintain a clear document structure, this chapter outlines the limitations of the system and proposes future improvements where relevant. The dimensions of the assessment are categorized by the design (Section 6.1), implementation (Section 6.2) and evaluation (Section 6.3); The chapter is concluded with a discussion of the threats to validity in Section 6.4.

6.1 Design of `DataViz`

We identified multiple limitations of the proposed design of `DataViz`, many of which could be addressed in future work.

L3.1 Design accuracy can be compromised due to non-deterministic nature of the LLMs. This can lead to variability in plot generation even when identical prompts are resubmitted several times repeatedly [35].

L3.2 Context-aware visualization can be affected by the currently ordered prompt building, as it assumes a structured dataset. This could be solved by focusing on the research of RAG, vector embeddings or dataless prompts [11, 49, 66, 86]. It cold be proposed that it could be evaluated which lines of the dataset are most representative or / and provide dataset schema to the LLM. These improvements could be done with the deployment of previously mentioned methods or by an LLM.

L3.3 Design of data upload could be a limitation in a production setting. For a more seamless enterprise integration connection to for instance DataLake for file access would be more efficient according to interviews with stakeholders [12].

L3.4 Programming languages can be a limitation in the current version of `DataViz` as it only supports plots generated using Python. This could be extended to, for instance, SQL scripts.

L3.5 NLI Design is limited by being only available to English-speaking stakeholders. Due to the simplified UI it would be essential to extend its functionality, aesthetics, and comprehension of multiple languages.

L3.6 Singular visualization generation is a design limitation as it does not yet take into account the generation of multiple images and dashboards.

L3.7 Zero-shot prompting limits the design by not allowing a conversational interaction due to the initial stages of the research. This could be easily implemented and would require the investigation of the evaluation metrics of such a system.

L3.8 Effective data visualization could be supported by the inclusion of more API calls, which would first evaluate the complexity of the user input question, so that the system could evaluate which LLM to use to create the specific visualization. This could improve the accuracy of the created data insights and minimize token usage based on the task complexity.

L3.9 Evaluation metrics aimed more at code evaluation, for example Pass@K.

6.2 Implementation of `DataViz`

We identified multiple limitations of the proposed implementation of `DataViz`, many of which could be addressed in future work.

L4.1 Data visualization accuracy can be a limitation due to incorrect understanding of the file structure or / and content of the OpenAI LLM. This is because the system reads the first three rows, including the headers statically, without assessing if these rows are representative. This calls for a future improvement of semantically choosing which rows to use as a sample, which would lower the execution failures when a dataset lacks particular hears or has empty rows and ambiguities.

L4.2 API cost is worth taking into account, as the use of API tokens can become fairly expensive when implemented in production. Future implementations could consider the deployment of open source models, such as Llama, and their accuracy evaluation [40].

L4.3 API time duration can also be a limitation of the performance of the DataViz system, as the varying OpenAI models have different price limits. In our design and implementation, we try to use as few tokens as possible to not extend the time it takes for the API to respond or cause timeouts. However, when using online models and servers, API calls may vary in latency due to server occupation, queueing, and network bandwidth.

L4.4 Large-scale data processing is limited in the fact that the LLM is not disclosed with the full dataset, which could affect its understanding of patterns between data and data ordering conventions. This calls for further research on efficient ways to provide context, such as RAG and vector embeddings or ways of extracting the schema of the dataset (headers and related data formats) [86].

L4.5 Secure script execution should be improved to decrease the security risk, as GitHub can contain malicious programs that alter or change their environments [57]. This is essential to protect both the confidentiality of the data and the privacy of the user. *Chen et al.* propose the use of Sandbox environment tailored to OpenAI training infrastructure and cloud services limitations [8].

L4.6 Singular visualization generation is a beginning point for research to create LLM-driven dashboards. Extending the use of the application could explore the implementation of Plotly for dashboard generation or visualization generation only from the frontend for improved interactivity and performance [54].

L4.7 Dataset limitations can be caused by varying the formats of the CSV file separators. The system should be implemented in a way that detects CSV file separators and adjusts effectively.

6.3 Evaluation of DataViz

We identified multiple limitations of the proposed evaluation of DataViz, many of which could be addressed in future work.

L5.1 Human evaluation is limited due to the possibility of human bias occurrence. For future work, it would be useful to assess the accuracy of the visualizations generated by data analytics experts.

L5.2 Data type variability is limited because the evaluation focused only on categorical, ordinal, and quantitative data types. Future assessments should include more sophisticated data types to verify the system's effectiveness across more complex datasets.

L5.3 Applied metrics are suitable but could be improved by more sophisticated methods of code and accuracy evaluations. Future work could include comparison against a data visualization baseline, for instance the research done by *Zheng et al.* [87].

L5.4 Performance evaluation could be improved by comparing the performance of various OpenAI models, especially those with lower cost rates.

L5.5 System usability within an enterprise deployment could be improved by a more seamlessly integrated data upload feature, preferably through the DataLake API connection [12].

L5.6 Dataset limitations can be caused by varying the formats of the CSV file separators. The system should be evaluated on more diverse datasets, preferably from different companies, to ensure reproducibility.

L5.7 Ambiguous and erroneous prompt evaluation is a limitation, as we currently only test it during Task B of the usability study, where the participant inserts a prompt using acronyms. This should be tested more thoroughly for further evaluation of the OpenAI API integration [35, 43]

6.4 Threats to Validity

We identified multiple threats to the validity of the thesis and listed them below.

T.1 Design validity can be obstructed by the system's reliance on structured CSV data as input. The system is designed in a way that allows large-scale file processing but does not account for scalability.

T.2 Implementation validity is possible due to lack of handling of CSV file ambiguities, such as varying data, missing data or headers, which can lead to potential failure or visualization inaccuracies. The usage of OpenAI API server can affect the performance of the application due to variability of latencies to successfully return a data visualization script.

T.3 Evaluation validity can be compromised by the lack of variability of the case study dataset. This can affect the generalization of the evaluation results. Furthermore, the accuracy study, specifically the human evaluation, introduces a potential bias in the findings. This is possible despite efforts to standardize the evaluation. The evaluation was

conducted in a controlled setting based on curated prompts using PE, therefore the evaluation might not fully represent a real-world application of the system.

7

Conclusion

The development of NLIs has paved the way for advancements in the application of LLM-driven solutions in varying fields and business environments. Making data visualization more accessible to a wider range of users by allowing them to express their queries and analysis intentions in natural language was one of the main goals of `DataViz`. However, the process of translating NLQs into large-scale visualizations is a challenging and rapidly developing study, which we addressed in this thesis. In this chapter we answer the research questions (Section 7.1) and summarize the work done and propose future improvements (Section 7.2) of the large-scale data visualization system using LLMs.

7.1 Answering Research Questions

With the research questions in mind we structured the work into four main chapters, each of which contributes to answering the main research question, **MRQ:** *How to design, implement, and evaluate a LLMs-based business data visualization system?*.

7.1.1 Chpater 2: Literature Review

To address **RQ1** *What are the state-of-the-art LLMs applications in data analytics and visualization?* we conduct a time-constrained systematic literature review of LLM-based systems for data analysis, specifically data visualization. The results are summarized and analyzed in Chapter 2. We synthesize the current literature gap in lightweight LLM-driven data visualization systems for large-scale data insight generation and subsequent evaluation approaches. We explore research on GPT models, as well as PE methods for script generation, concluding the Chapter with a concise terminology list and a summary (**RC1**).

7.1.2 Chpater 3: Design of DataViz

To address **RQ2** *How to design a business data visualization system driven by LLMs?* based on the literature review, we design the DataViz system for large-scale data visualization driven by LLMs. We perform a comprehensive requirements analysis and follow it by a proposed design overview, which comprises the NLI and the data visualization pipeline. We ensure system usability and interactivity using various error handling approaches. We perform PE using the novel Structured Chain-of-Thought method and analyze its components in Section 3.3.3. We establish metrics useful for DataViz evaluation and conclude the Chapter with limitations, future improvements, and summary of the design (**RC2**).

7.1.3 Chpater 4: Implementation of DataViz

To address **RQ3** *How to implement a business data visualization system driven by LLMs?* we implement the DataViz system for large-scale data visualization driven by LLMs. We provide in detailed motivation behind implementation choices, as well as its reproducibility. Each repository on GitHub provides instructions on running the system and the experiments. We conclude the Chapter with limitations, future improvements, and summary of the implementation (**RC3**).

7.1.4 Chpater 5: Evaluation of DataViz

To address **RQ4** *How to evaluate a business data visualization system driven by LLMs?* we evaluate the DataViz system among varying categories: accuracy, performance, and usability. We apply the metrics described in Chapter 3 and provide detailed experimental set-up recommendations. We provide the results of the experiment as visual insights to ensure clarity of the findings. We conclude the Chapter with limitations, future improvements, and summary of the evaluation (**RC3**).

7.2 Summary and Future Work

Due to the rapid development of LLM-driven analytics, the future implications of this thesis are vast. We present a concise list of possible directions for future work.

F6.1 We expect the DataViz system to expand the functionality it offers and provide deeper data analytics insights to stakeholders. This could involve a textual analysis driven by LLMs of the data and its visualizations.

F6.2 We identify a list of possible design improvements that could positively influence the accuracy of the `DataViz` visualizations. We consider the integration conversational mode as valuable to ensure enhanced UX of the NLI for non-technical users. In addition, enhancing the system to create interactive plots and dashboards is of great importance. Expanding the system according to specific business workflows is essential for its success, therefore updating the data upload module by connections to company files stored on, for instance,various platforms could improve its usability. Finally, providing the LLM with a dateset schema could offer more sophisticated context-aware insights.

F6.3 We notice a clear opportunity for a more comprehensive evaluation of `DataViz`, and the application of novel metrics and systems for the semantic evaluation of the data visualization script and the intention behind the NLQ and the dataset. A comparative study against state-of-the-art baselines could offer more insight into the system's capabilities and areas for improvements.

The implementation of `DataViz` and all the experiment data produced as part of this work are publicly available on GitHub[12] (**RC4**).

[1] `https://github.com/mmyalen/DataViz`
[2] `https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz`

Bibliography

[1] Adobe. Png vs. svg: What are the differences?, 2024. Accessed: 2025-06-07. 39

[2] T. Auger and E. Saroyan. Overview of the openai apis. In *Generative AI for Web Development: Building Web Applications Powered by OpenAI APIs and Next. js*, pages 87–116. Springer, 2024. 29, 38

[3] S. Bezjak, A. Clyburne-Sherin, P. Conzett, P. L. Fernandes, E. Görögh, K. Helbig, B. Kramer, I. Labastida, K. Niemeyer, F. Psomopoulos, et al. The open science training handbook. 2018. 6, 7

[4] S. Bialkova. *The Rise of AI User Applications*. Springer, 2024. 1, 47

[5] A. Booth, M. Martyn-St James, M. Clowes, and A. Sutton. Systematic approaches to a successful literature review. 2021. 11

[6] E. Castro. *HTML for the world wide web*. Peachpit Press, 2003. 36

[7] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie. A survey on evaluation of large language models. 15(3), 2024. 9

[8] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 10, 13, 30, 31, 57

[9] M. Chui, E. Hazan, R. Roberts, A. Singla, and K. Smaje. The economic potential of generative ai. 2023. 1, 2, 3

[10] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. 13

BIBLIOGRAPHY

[11] D. Coelho, H. Barot, N. Rathod, and K. Mueller. Can llms generate visualizations with dataless prompts? *arXiv preprint arXiv:2406.17805*, 2024. 55

[12] Databricks. Introduction to data lakes, 2023. Accessed: 2025-06-07. 51, 55, 58

[13] V. Dibia. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*, 2023. 1, 2, 13, 14, 31

[14] V. Dibia. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*, 2023. 31, 44, 49

[15] A. Dünser, R. Grasset, H. Seichter, and M. Billinghurst. Applying hci principles to ar systems design. 2007. 25

[16] S. Ekin. Prompt engineering for chatgpt: a quick guide to techniques, tips, and best practices. *Authorea Preprints*, 2023. 38

[17] D. O. Embarak and O. Embarak. The importance of data visualization in business intelligence. *Data analysis and visualization using python: analyze data to create visualizations for BI systems*, pages 85–124, 2018. 2

[18] Figma. Figma: The collaborative interface design tool, April 2025. Accessed on April 13, 2025. 5, 25

[19] L. Giray. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633, 2023. 15

[20] K. Healy. *Data visualization: a practical introduction*. Princeton University Press, 2024. 44

[21] J. J. Hilda, C. Srimathi, and B. Bonthu. A review on the development of big data analytics and effective data visualization techniques in the context of massive and multidimensional data. *Indian Journal of Science and Technology*, 9(27):1–13, 2016. 1

[22] IBM. Ibm analytics, 2025. Accessed: 2025-05-09. 2

[23] A. Iosup, L. Versluis, A. Trivedi, E. Van Eyk, L. Toader, V. Van Beek, G. Frascaria, A. Musaafir, and S. Talluri. The atlarge vision on the design of distributed systems and ecosystems. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1765–1776. IEEE, 2019. 19

[24] A. Joshi, S. Kale, S. Chandel, and D. K. Pal. Likert scale: Explored and explained. *British journal of applied science & technology*, 7(4):396, 2015. 32

[25] S. Kujala. *User studies: A practical approach to user involvement for gathering user needs and requirements.* Helsinki University of Technology, 2002. 10

[26] K. Kyoreva. State of the art javascript application development with vue. js. In *Proceedings of International Conference on Application of Information and Communication Technology and Statistics in Economy and Education (ICAICTSEE)*, pages 567–572. International Conference on Application of Information and Communication ..., 2017. 36

[27] Y. Levy and T. J. Ellis. A systems approach to conduct an effective literature review in support of information systems research. *Informing Science*, 9, 2006. 11

[28] J. Li, G. Li, Y. Li, and Z. Jin. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–23, 2025. 10, 16, 28, 30

[29] S. Li, X. Chen, Y. Song, Y. Song, and C. Zhang. Prompt4vis: Prompting large language models with example mining and schema filtering for tabular data visualization. *arXiv preprint arXiv:2402.07909*, 2024. 1, 14, 29

[30] C. Liu, Y. Han, R. Jiang, and X. Yuan. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20, 2021. 43, 44

[31] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *ArXiv*, abs/2304.08485, 2023. 10

[32] Y. Luo, N. Tang, G. Li, C. Chai, W. Li, and X. Qin. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1235–1247, 2021. 12, 13

[33] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226, 2021. 1

[34] C. Macrae. *Vue. js: up and running: building accessible and performant web apps.* " O'Reilly Media, Inc.", 2018. 36

[35] P. Maddigan and T. Susnjak. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *Ieee Access*, 11:45181–45193, 2023. 1, 2, 10, 13, 14, 15, 55, 58

[36] V. Maphosa. The rise of artificial intelligence and emerging ethical and social concerns. *AI, Computer Science and Robotics Technology*, 2024. 1

[37] G. Marvin, N. Hellen, D. Jjingo, and J. Nakatumba-Nabende. Prompt engineering in large language models. In *International conference on data intelligence and cognitive informatics*, pages 387–402. Springer, 2023. 10

[38] Matplotlib Development Team. *Matplotlib: Agg Backend API*, 2024. Accessed: 2025-06-07. 38

[39] S. McCrystal. How unilever's digital transformation is driving operational excellence. *Unilever News*, February 2025. 1

[40] Meta Platforms, Inc. Llama by meta, 2025. Accessed: 2025-06-07. 56

[41] Microsoft. Write copilot prompts for creating report pages in power bi, 2024. Accessed: 2025-05-09. 2

[42] A. M. Mithun, Z. A. Bakar, and W. M. Yafooz. The impact of web contents color contrast on human psychology in the lens of hci. *Int. J. Inf. Technol. Comput. Sci.*, 11(10):27–33, 2019. 25

[43] A. Narechania, A. Srinivasan, and J. Stasko. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2020. 2, 58

[44] A. Narechania, A. Srinivasan, and J. Stasko. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2020. 2, 13

[45] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35, 2021. 9, 10

[46] W. S. L. Nasution and P. Nusa. Ui/ux design web-based learning application using design thinking method. *ARRUS Journal of Engineering and Technology*, 1(1):18–27, 2021. 5, 26

[47] OpenAI. Best practices for api key safety. `https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety`, 2024. Accessed: 2025-06-16. 37

[48] OpenAI. Latency optimization guide, 2024. Accessed: 2025-06-27. 31, 32

[49] OpenAI. Openai platform documentation. `https://platform.openai.com/docs/overview`, 2024. Accessed: 2025-05-21. 16, 29, 38, 45, 55

[50] P. Paidy and K. Chaganti. Securing ai-driven apis: Authentication and abuse prevention. *International Journal of Emerging Research in Engineering and Technology*, 5(1):27–37, 2024. 29

[51] Pallets Projects. *Flask: web development, one drop at a time*, 2024. Version 2.3.3. 36

[52] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002. 30

[53] F. Petrillo, A. S. Spritzer, C. M. D. S. Freitas, and M. S. Pimenta. Interactive analysis of likert scale data using a multichart visualization tool. *IHC+ CLIHC*, 67:358–368, 2011. 51

[54] Plotly Technologies Inc. Dashboards - plotly, 2025. Accessed: 2025-06-07. 38, 57

[55] L. P. Ramos. Python's exec(): Execute dynamically generated code, 2022. Accessed: 2025-06-07. 38

[56] P. Ranganathan and N. J. Gogtay. An introduction to statistics–data types, distributions and summarizing data. *Indian journal of critical care medicine: peer-reviewed, official publication of Indian Society of Critical Care Medicine*, 23(Suppl 2):S169, 2019. 42

[57] M. O. F. Rokon, R. Islam, A. Darki, E. E. Papalexakis, and M. Faloutsos. {SourceFinder}: Finding malware {Source-Code} from publicly available repositories in {GitHub}. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 149–163, 2020. 57

[58] S. Sah, R. Mitra, A. Narechania, A. Endert, J. Stasko, and W. Dou. Generating analytic specifications for data visualization from natural language queries using large language models. *arXiv preprint arXiv:2408.13391*, 2024. 2, 3, 13

[59] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024. 15, 16

[60] B. Schmarzo. *Big Data: Understanding how data powers big business*. John Wiley & Sons, 2013. 1

[61] L. K. M. Schulz. ShareBench: Performance characterization of distributed resource-sharing mechanisms. 49

[62] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th annual symposium on user interface software and technology*, pages 365–377, 2016. 38

[63] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 29(6):3121–3144, 2023. 10

[64] R. Siatri. The evolution of user studies. 1999. 10

[65] Y. Song. *Constructing Natural Language Interfaces for Data Querying and Visualization*. PhD thesis, Hong Kong University of Science and Technology (Hong Kong), 2024. 49

[66] Y. Song, X. Zhao, R. C.-W. Wong, and D. Jiang. Rgvisnet: A hybrid retrieval-generation neural framework towards automatic data visualization generation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1646–1655, 2022. 10, 55

[67] L. South, D. Saffo, O. Vitek, C. Dunne, and M. A. Borkin. Effective use of likert scales in visualization evaluations: A systematic review. In *Computer Graphics Forum*, volume 41, pages 43–55. Wiley Online Library, 2022. 51

[68] F. Southey. How unilever is using ai and big data to transform its food portfolio. *FoodNavigator*, July 2023. 1

[69] K. Srinath. Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, 4(12):354–357, 2017. 24, 36

[70] Tableau. Tableau: Business intelligence and analytics software, 2025. Accessed: 2025-05-09. 2

[71] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. Chartgpt: Leveraging llms to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 10, 13, 14, 15

[72] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 13

[73] M. F. Triola, W. M. Goodman, R. Law, and G. Labute. *Elementary statistics*. Pearson/Addison-Wesley Reading, MA, 2004. 49

[74] Unilever. Datalab, November 2023. Accessed on April 13, 2025. 3, 4, 29, 35

[75] Unilever. Hive homepage, June 2024. Accessed on April 12, 2025. 1, 2

[76] Z. Wan, Y. Song, S. Li, C. J. Zhang, and R. C.-W. Wong. Datavist5: A pre-trained language model for jointly understanding text and data visualization. *arXiv preprint arXiv:2408.07401*, 2024. 30

[77] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232, 2022. 1, 2

[78] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022. 9

[79] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 15, 16

[80] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023. 15

[81] C. Wu, S. Yin, W. Qi, X. Wang, Z. Tang, and N. Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023. 10

[82] Y. Wu, Y. Wan, H. Zhang, Y. Sui, W. Wei, W. Zhao, G. Xu, and H. Jin. Automated data visualization from natural language via large language models: An exploratory study. *Proceedings of the ACM on Management of Data*, 2(3):1–28, 2024. 10

[83] Y. Xiao and M. Watson. Guidance on conducting a systematic literature review. *Journal of planning education and research*, 39(1):93–112, 2019. 11, 12

[84] E. You and the Vue.js Core Team. *Vue.js – The Progressive JavaScript Framework*, 2024. Version 2.6.14 or 3.x depending on your usage. 36

[85] H. Zhang, H. Wang, J. Li, and H. Gao. A generic data analytics system for manufacturing production. *Big Data Mining and Analytics*, 1(2):160–171, 2018. 9

[86] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024. 55, 57

[87] J. G. Zheng. Data visualization in business intelligence. In *Global business intelligence*, pages 67–81. Routledge, 2017. 2, 58

[88] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. 15, 16

[89] P. Zikopoulos and C. Eaton. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011. 9

Appendix A

Reproducibility

A.1 Artifact check-list (meta-information)

- **Program:** `DataViz` **program for large-scale data visualization.**
- **Model: OpenAI models (GPT-4o, GPT-4o-mini, O1-mini)**
- **Dataset:** **`https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz`.**
- **Hardware: Apple M2 (8-core CPU, 10-core GPU), 16 GB Unified Memory, 494.38 GB SSD.**
- **Metrics: Human Evaluation, Likert Scale, VER**
- **Output: Data visualization based on NLQ and CSV dataset.**
- **How much disk space required (approximately)?: 332K**
- **How much time is needed to prepare workflow (approximately)?: 5 minutes.**
- **How much time is needed to complete experiments (approximately)?: 1-3 days.**
- **Publicly available?: Yes**
- **Code licenses (if publicly available)?: Apache License 2.0**

A.2 Description

A.2.1 How to access

The following repositories contain all the code and works:

Implementation: `https://github.com/mmyalen/DataViz`

Experiments: `https://github.com/mmyalen/experiments-bsc-thesis-2025-DataViz`

A.2.2 Software dependencies

Pipfile: `https://github.com/mmyalen/DataViz/blob/main/Pipfile`
Pipfile.lock: `https://github.com/mmyalen/DataViz/blob/main/Pipfile.lock`

A.3 Installation

1. Install Python 3.11 and Pipenv
2. Run `pipenv install`
3. Create a `.env` file with:

   ```
   OPENAI_API_KEY='...'
   ```

4. Run the app with `pipenv run python app.py`