CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Automating Financial Report Analysis and Generation using LLMs

Exploring LLM-Based Automation in Financial Analysis and Reporting: Software Engineering Challenges and Design Trade-offs

Master's Thesis in Computer science and engineering

Theo Koraag, Niklas Wagner

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's Thesis 2025

Automating Financial Report Analysis and Generation using LLMs

Exploring LLM-Based Automation in Financial Analysis and Reporting: Software Engineering Challenges and Design Trade-offs

Theo Koraag, Niklas Wagner

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Automating Financial Report Analysis and Generation using LLMs
Theo Koraag, Niklas Wagner

Supervisor: Lucas Gren, Department of Computer Science and Engineering
Examiner: Richard Torkar, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Theo Koraag, Niklas Wagner
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Background: The emergence of Large Language Models (LLMs) has enabled automation of complex natural language processing across a wide range of domains. Still, their application and research on designing them in the financial domain remains limited.

Objective: This study explored how LLMs can be integrated into financial report analysis and commentary generation, with a particular focus on the software engineering challenges encountered during the design and implementation of such solutions.

Method: A Design Science Research methodology was used where an exploratory case study was conducted. Two LLM-based systems were iteratively designed and evaluated: one employing local open-source models in a multi-agent workflow, and the other utilizing GPT-4o. Both solutions were evaluated through expert assessments of a real-world financial reporting use case.

Results: It was observed that LLMs had great potential to automate tasks within financial reporting workflows, yet their integration presents challenges. Through iterative development and expert evaluation, several issues were identified, including prompt design, contextual dependency, and trade-offs between implementation options. Cloud-based models were found to offer greater fluency and ease of use, but raised concerns related to data privacy and reliance on external services. In contrast, local open-source models provide stronger data control and compliance but require substantially more engineering effort to ensure reliability and usability.

Conclusion: LLMs showed strong potential for automating financial reporting, but their integration requires careful attention to architecture, prompt design, and system reliability. Successful implementation depends on addressing domain-specific challenges through tailored validation mechanisms and engineering strategies that strike a balance between accuracy, control, and compliance.

Keywords: Large Language Models, Financial Reporting, NLP, Software Engineering, Workflow Automation, AI Integration, GPT-4o, Local LLMs

Acknowledgements

We would like to express our deepest gratitude to everyone who supported and guided us throughout the course of this thesis, including our family, friends, and fellow students, for their support, motivation, and encouragement during our studies.

First and foremost, we would like to thank our supervisor, Lucas Gren, for the insightful feedback, encouragement, and constructive guidance throughout the entire research process. Your expertise and support were invaluable to the development of this work. We are also grateful to our industrial partner, Getinge AB, for providing us with the opportunity to work on a real-world problem and for their support during our project. Special thanks to the domain experts and business controllers who contributed their time and insights to our evaluation, which played a crucial role in shaping the results of this thesis. We would also like to thank our examiner, Richard Torkar, and our opponents for reviewing our thesis and providing critical and valuable feedback.

Theo Koraag, Niklas Wagner, Gothenburg, June 2025

Contents

List of Figures

List of Tables

1

Introduction

The rapid development of artificial intelligence (AI) has significantly impacted many industries, with natural language processing (NLP) playing a pivotal role in automating text-based tasks. An influential development in NLP is the introduction of the transformer architecture, which paved the way for large language models (LLMs) [72]. These models have demonstrated exceptional performance in understanding, analyzing, and generating text that is human-like. These capabilities enable functionalities that would otherwise be unfeasible or require extensive effort using traditional software engineering approaches [74]. As a result, new opportunities have emerged within domains and industries that seek to automate and improve their workflows with the help of LLM-based tools. Domains such as healthcare and education leverage LLMs for various tasks. In healthcare, they assist with medical documentation, summary of patient records, and diagnostic support [45]. In education, LLMs enhance personalized learning by generating tailored study materials [10].

However, a relatively underexplored area is the application of LLMs in financial report analysis and generation [4, 85]. Financial report analysis is critical for businesses, investors, and regulatory bodies, as it provides insight into a company's financial health. Traditional methods often involve manual efforts that are time-consuming and prone to inconsistencies [3]. Recent advancements in NLP and LLMs offer promising solutions to automate financial document analysis, extracting key insights, summarizing reports, and enhancing decision-making processes [79]. Despite this potential, there is limited knowledge about how these systems can be effectively integrated into real-world financial settings. This research aims to explore the practical design of such systems, while also identifying the challenges that arise in doing so. Issues such as data sensitivity [1], domain-specific terminology [9], and the need for interpretability and accuracy may pose significant obstacles to successful implementation.

1.1 Problem Description

Corporate finance involves a significant amount of manual and repetitive work. Financial analysts spend considerable time analyzing data, preparing financial statements, and performing fact-checking to ensure accuracy. These tasks typically re-

quire the processing of large volumes of information presented in various formats, including Excel, PDFs, and verbal communication, which further adds complexity to the analysis and reporting process. While this manual approach can be effective, it is time-consuming and prone to human error, particularly when consistency and timeliness are critical [3].

AI, and more specifically LLMs, offer promising solutions to these challenges by automating the monotonous and cognitively demanding aspects of data analysis and summarization. By using NLP techniques, LLMs can extract key information, summarize financial reports, and generate coherent narratives based on the underlying data [63]. However, integrating LLMs into such workflows requires addressing several Software Engineering concerns, including data privacy, accuracy, precision, scalability, and maintainability.

In alignment with the direction outlined by Uchitel et al. [71], this thesis investigates systems that incorporate LLMs as a software component. The focus lies not only on the capabilities of LLMs but rather on the broader software engineering processes required to design, implement, and evaluate such systems. The study contributes to the ongoing discussion on how Software Engineering practices are evolving to support AI-enabled systems. Conducted in collaboration with Getinge AB, this thesis is grounded in real-world financial reporting use cases.

1.2 Purpose of the Study

This thesis explores the challenges of integrating Large Language Models (LLMs) into financial workflows through an exploratory case study. Focusing on the automation of financial report analysis, the study aims to identify key technical and organizational barriers and to provide insights that inform future design and development of LLM-based systems in financial contexts.

1.3 Research Questions

To guide this study, the following research questions have been formulated:

RQ1: *What are the primary software engineering challenges in integrating LLMs into financial report analysis workflows?*

RQ2: *How do different implementation approaches compare in terms of software engineering complexity, performance, and maintainability?*

1.4 Scope and Delimitations

This thesis focuses primarily on the software engineering aspects of integrating LLMs into the financial report analysis. It examines the design and development of an architecture for a modular AI-based system aimed at automating the summarization

of specific components within the financial reporting process. The research focuses on workflow optimization, system reliability, and the evaluation of the effectiveness of the developed architecture in real-world financial reporting scenarios.

Several delimitations have been defined for this research. The work does not involve the development or tuning of an LLM but instead concentrates on adapting and integrating existing models. The scope is limited to LLMs and does not extend to other types of NLP models. Moreover, while the study focuses on automating aspects of data analysis, it does not address more complex financial analysis tasks such as stock market prediction or portfolio optimization. The study conducted is qualitative rather than quantitative and does not include large-scale user studies or benchmarking using standardized performance metrics.

1.5 Significance of the Study

The findings of the study will provide a greater understanding of the software engineering challenges involved in implementing LLMs into real-world financial systems. By bridging the gap between AI and the finance world, the research will offer insights into the best practices for designing AI-driven solutions in financial workflows, ensuring efficiency, accuracy, and compliance with industry regulations. Furthermore, the study will contribute to the broader field of Software Engineering by examining key aspects of AI-driven systems such as reliability, security, and maintainability. These findings will help organizations adopt robust engineering practices when implementing LLMs, ultimately improving the trustworthiness and effectiveness of AI applications in finance.

2

Theoretical Background

This chapter provides the theoretical foundation for the thesis by first introducing key approaches to financial analysis, including commonly used methods, financial ratios, and performance indicators. This establishes the domain context necessary for understanding the integration of language technologies. The chapter then explores the evolution of natural language processing (NLP), with a particular focus on the emergence of transformer-based architectures that have driven recent advancements in language modeling. Building on this, an overview of large language models (LLMs) is presented, detailing their training paradigms and capabilities in generating both structured and unstructured text. The chapter concludes with a review of recent developments in LLM system design, highlighting engineering considerations critical for their implementation in financial reporting and analysis workflows.

2.1 Financial Statement Analysis and Reporting

Financial statements offer a snapshot of a company's financial condition, providing insights into its performance, profitability, liquidity, and overall economic position at a given point in time [12]. Analyzing statements, such as income statements, balance sheets, order intake reports, and cash flow statements, is a central activity in assessing a company's financial health and future trajectory. Financial statement analysis supports both external and internal stakeholders in making informed decisions. Investors and creditors use it to evaluate risk and return potential, while executives and management rely on it to monitor performance and steer strategic directions [12].

2.1.1 Financial Statements

- **Income Statements:** Presents a company's revenue, expenses, and profits. Provides detailed information regarding operational efficiency and profitability trends over a specific time period. Includes information such as total revenue, cost of goods sold (COGS), gross profit, operating expenses, operating income (EBIT), and net income [12].

- **Balance Sheets:** Detailing assets, liabilities, and shareholder equity, these statements give a snapshot of the company's financial position at a given time

[12].

- **Cash Flow Statements:** Providing information about a company's cash inflows and outflows, this statement helps assess liquidity and overall cash management. It is divided into three main components. Operating activities include net income, depreciation, and changes in working capital. Investing activities include capital expenditures, asset purchases, and sales. Lastly, financing activities detail debt issuance, dividend payments, and equity transactions [12].

2.1.2 Approach and Challenges to Modern Financial Statement Analysis

Historically, financial statement analysis mainly utilized rule-based systems such as expert systems and spreadsheet analyses. Such methods were reliant on predefined formulas, structured datasets, and human expertise, which made them effective at analyzing standardized reports. Examples of such techniques are:

- **Expert Systems:** Expert systems are computer applications that emulate the decision-making capabilities of human experts within a specific domain. They rely on knowledge explicitly provided by domain experts, typically in the form of rules (e.g., IF-THEN logic). These systems use this structured knowledge base to solve problems, answer questions, and provide recommendations based on the encoded expertise [67].

- **Spreadsheet Analysis:** Spreadsheet analysis utilizes tools like Microsoft Excel and Google Sheets to organize, manipulate, and analyze large amounts of data [64]. It involves building models that perform calculations such as ratio and trend analysis and forecasting to generate insights from financial data.

- **Ratio Analysis:** Ratio analysis involves examining a company's financial health by analyzing the relationship between various financial figures. It provides insight into different financial aspects such as profitability, liquidity, leverage, and efficiency [12].

- **Vertical and Horizontal Analysis:** Vertical and horizontal analysis compare financial data over different periods. Vertical analysis focuses on analyzing the relationship between various financial statements items in a single time period. It helps calculate and assess cost, profitability, and financial stability for each component, such as revenue, operating expenses, etc. Horizontal analysis focuses more on growth trends and patterns by examining financial statements over more extended periods. It helps in identifying increases or decreases in revenue, expenses, and other financial metrics [12].

However, such approaches often struggle with the complexity of modern financial documents, especially when dealing with unstructured data such as management commentaries, footnotes, and verbal communication [36]. In contrast, Machine

Learning (ML) and Natural Language Processing (NLP) solutions can automatically parse and analyze text and numerical data at a larger scale with higher efficiency. Nonetheless, challenges persist, such as inconsistent data formats across filings, the need for domain-specific tuning, and the sheer volume of information.

2.2 Natural Language Processing (NLP)

Natural Language Processing is a subfield within artificial intelligence and computer science that focuses on enabling systems to understand and generate text in a human manner by utilizing various methods such as machine learning, deep learning, and linguistic rules [30]. NLP uses a variety of techniques in order to effectively interpret, generate, and interact with human language. To effectively process textual data, several key steps are performed, such as text preprocessing, feature extraction, text analysis, and model training [66].

- **Text Preprocessing:** Raw text is first preprocessed before being analyzed by transforming it into a simpler format. Longer texts undergo tokenization, where they are broken down into smaller units such as sentences, words, or paragraphs, which reduces complexity. The tokens are then standardized and filtered by lowercasing each character and removing unnecessary filler words such as "the" and "is". Lastly, stemming or lemmatization is performed where words are converted to their root form, for example, eating to eat [30].

- **Feature Extraction:** The processed dataset is then converted into numerical representations that machines can interpret and analyze by using NLP techniques such as Bag of Words (BOW) and Term Frequency-Inverse Document Frequency (TF-IDF). The NLP techniques quantify the importance and occurrences of words in a text by transforming them into vectorized forms, which are easier for machines to process. Vectorization enables mathematical operations, such as calculating word similarities and relationships, and allows machine learning models to capture patterns in the data [41]. In addition to BOW and TF-IDF, other common NLP feature extraction methods like word embeddings, Word2Vec [46], and GloVe [57], which capture semantic meanings and relationships between words.

- **Text Analysis:** Text analysis is the process of interpreting and understanding text data by extracting meaningful information such as the sentence tone, context, and sentiment. Various computational techniques are utilized, such as part-of-speech (POS) tagging, which classifies the grammatical role of each word, named entity recognition (NER), which recognizes and separates names, dates, and locations [30].

- **Model Training:** To ensure high accuracy and precision, a model is trained by identifying patterns within large volumes of data. After preprocessing the data, the model is initialized with random parameters and then fed extensive datasets. It makes predictions based on the inputs, and a loss function

measures the difference between predictions and true values. The model's parameters are then updated and optimized to minimize this loss. This training process is repeated over multiple iterations and epochs with periodic validation to prevent overfitting. Finally, the model is evaluated on a separate test dataset to assess its performance on unseen data [30].

2.2.1 Evolution of NLP, Past and Modern Models and Approaches

Early versions of NLP were built using a rule-based approach with predefined logic structures resembling decision trees. Early examples include Weizenbaum's ELIZA [76], which used manually crafted rules to simulate human-like conversation. The systems were simpler, less scalable, and less effective at processing human language due to the complexity of natural languages. This includes a variety of syntactic structures and the complexity of ambiguity and context. The limitations of a rule-based approach, such as their inability to fully capture the complexities of natural language, led to the development of statistical methods in NLP[19].

Statistical methods in NLP utilize probabilistic models and machine learning techniques to learn and analyze human language. These models interpret natural language by identifying patterns in large volumes of annotated text data. Commonly used probabilistic models in statistical NLP include Hidden Markov Models for part-of-speech (POS) tagging [34], Naïve Bayes classifiers for text classification and spam detection [44], and n-gram models for language modeling and speech recognition [16]. Compared to traditional rule-based systems, statistical NLP models are generally more scalable and adaptable, as they can learn and improve their accuracy through exposure to more data [30].

The development of deep learning based models marked a significant change from earlier rule-based and statistical approaches in Natural Language Processing. Neural architectures enabled models to automatically extract and learn hierarchical representations of language from large-scale datasets, improving the ability to capture syntax, semantics, and contextual dependencies. Early architectures such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks facilitated sequential language modeling by maintaining temporal information over input sequences [26]. More recently, the introduction of the Transformer architecture [72] enabled parallel processing of sequence data and improved modeling of long-range dependencies, paving the way for large-scale pre-trained models. These advances culminated in the development of LLMs such as BERT [18] and GPT [58], which have significantly advanced the state of the art in numerous NLP tasks.

2.3 Large Language Models (LLMs)

LLMs are an advanced version of NLP models designed to understand, analyze, and generate human language with higher accuracy than older models. These models employ deep learning techniques, specifically the Transformer architecture, to ana-

lyze large volumes of data and learn complex linguistic patterns [72]. The models are classified as probabilistic models, as they predict the most likely sequence of words based on the learned data distributions [48].

2.3.1 Transformer Architecture and Key Components

The transformer architecture is a deep learning architecture first introduced by Google researchers in 2017 [72]. The transformer architecture represents a significant step up from older models, such as Recurrent Neural Networks and Long Short-Term Memory (LSTM). Unlike its predecessor, which processes text sequentially, the transformer architecture utilizes a mechanism called self-attention, which allows it to process data in parallel, allowing it to capture long-range dependencies and context more efficiently.

The Self-attention mechanism is a fundamental part of the transformer architecture that allows the model to capture and analyze the relationships between words/tokens regardless of their position in a sentence. Additionally, self-attention enables the models to handle all the tokens simultaneously, improving their efficiency and their ability to understand the relationships between different tokens. Self-attention is calculated using the following mathematically defined formula:

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

Where $Query$(Q), Key(K), and $Value$(V) are vectors obtained from the input embeddings through learned weight matrices. Query represents the token that seeks related contextual information from other tokens in a sequence, while Key represents all the words in a sentence and their linguistic importance, and Value contains the actual information related to each token. QK^T calculates the dot-product similarity between the queries and keys, which determines the importance and relevance of each token in relation to the others. The calculated value is then scaled by $\frac{1}{\sqrt{d_k}}$, which is the key dimension d_k to prevent large values that could cause issues during training due to small gradients after applying the softmax function. The softmax function then normalizes these scores into attention weights, ensuring that they sum to 1. These attention weights are then multiplied by the $Value$ (V) matrix to generate a weighted sum of values [72].

An extension of self-attention called multi-head attention performs multiple self-attention calculations simultaneously, which further increases the model's ability to capture dependencies. Each instance calculates a different relationship between each token, essentially analyzing multiple perspectives of a sentence at once [72].

A standard transformer consists of 2 components, an *encoder* and a *decoder*, with some models using either both or one of the components, depending on the task it's designed for.

- **Encoder:** The encoder is responsible for processing and encoding the input data and converting it into a contextualized output. The encoder consists of

2 layers, the self-attention layer and the feed-forward neural network. The first layer calculates the importance and context of a token in the input data. The feed-forward neural network layer then further transforms the encodings to ensure that they are acceptable for future layers [47]. Models like BERT, which only utilize the encoder stack, are best suited for tasks that analyze the meaning of text, such as text classification, named entity recognition, and question answering [18].

- **Decoder:** The decoder component is responsible for creating sentences by predicting the tokens based on previously generated tokens which allows it to produce coherent and contextually accurate outputs. The decoder has an additional layer compared to the encoder, called the encoder-decoder attention layer, which is responsible for processing the encoder output as input and enables it to incorporate the information into the generated decoder output sequence. Decoder-based models such as GPT are better suited for tasks such as language modeling, text completion, and dialogue systems since they are designed for auto-regressive text generation [58].

- **Encoder and Decoder:** Models like T5 use both components and are typically used for text-to-text transformations such as language translations, text generation, and summarization [47].

Modern LLMs typically undergo two training phases: first, pretraining on large-scale datasets, and then fine-tuning for specific tasks. During the pretraining phase, the model learns a broad understanding of language and world knowledge from vast amounts of text data [13]. Fine-tuning then adapts the pretrained model to narrower domains or tasks, such as sentiment analysis or financial forecasting, by training it on smaller, specialized datasets. This process enables LLMs to apply their general language understanding to specific applications, thereby enhancing their performance and relevance in particular contexts.

2.3.2 Prompting Techniques

As LLMs continue to advance, the way they are prompted plays a critical role in the quality and accuracy of their outputs. Prompt design refers to how tasks are framed and structured for the model, and impacts the relevance, coherence, and correctness of generated responses. Well-constructed prompts typically rely on unambiguous instructions and can define the role of the model or include examples of the expected output format to guide behavior [59]. Prompting strategies typically involve a mix of task instructions and illustrative examples. While some prompts emphasize natural language instructions to convey what the model should do, others include demonstrations that show how to perform the task. Studies suggest that incorporating demonstrations can substantially improve performance, particularly on tasks that require reasoning, structured outputs, or adherence to domain-specific conventions [88].

Anthropic, in their prompt design guidelines, highlight several principles that align

with established findings on effective prompt construction [5]. Firstly, they emphasize the value of providing contextual information, such as the overall purpose of the task, the intended audience, and the task's role within a broader workflow as a means of guiding the model's outputs more effectively. Secondly, they underscore the importance of clearly articulating what the model is expected to do, including formatting requirements or constraints, as well as explicitly stating any undesired behaviors. Thirdly, they recommend sequentially structuring instructions through numbered or bulleted lists to enhance consistency in task execution. Finally, the inclusion of concrete examples of the desired output is suggested as a way to improve response and alignment with user expectations [5].

Two common prompting strategies that have demonstrated effectiveness in structured reasoning and adaptation to new tasks are *Chain-of-Thought prompting* and *Few-Shot learning*.

Chain-of-thought: Is a prompting technique that instructs LLMs to generate and display the intermediate reasoning steps they are taking when solving a problem, improving performance in tasks such as arithmetic, common sense, and symbolic reasoning [75]. By including examples that show how a complex question can be broken down into smaller parts, this method increases interpretability and ensures that the model's final answer is guided by a logical sequence of thought.

Few-shot prompting is a technique used with LLMs where the user provides a small number of examples directly in the prompt to guide the model's output [13]. By including demonstrations of the task within the input, the model can infer the desired format, tone, or logic without requiring additional training. This approach is particularly useful when task-specific data is limited or when fast adaptation to new tasks or domains is needed.

The effectiveness of a prompt can also depend on how it utilizes the models available context window. LLMs, such as GPT-3 and GPT-4, are capable of processing thousands of tokens within a single prompt, enabling the handling of complex multi-step instructions, in-context learning, and long input documents. More recent models have extended this capability. For example, Anthropics Claude 3.7 Sonnet can handle a context window of up to 200 000 tokens [7], and Google DeepMinds Gemini 2.5 supports up to 1 million tokens, making it possible to process entire codebases, books, or multi-document inputs in a single session [23].

However, increased context length does not guarantee better performance. Liu et al. [37] show that language models often struggle to utilize information uniformly across long prompts. Their study finds that model accuracy tends to peak when relevant content appears at the beginning or end of the input, but can drop when key information is located in the middle [37]. This behavior persists even in models designed for extended contexts, highlighting that merely increasing context size is not enough. Therefore, the placement of relevant information can have an effect on the effectiveness of the prompt.

2.3.3 LLMs as Agents

One way of conceptualizing LLMs is to use them as agents where they are assigned autonomous roles within a system to perform reasoning, decision-making, or task execution [56]. In this context, an LLM-as-Agent is treated as an autonomous entity that operates in a partially observable environment, takes actions, observes outcomes, and makes decisions over multiple interaction turns. A common technique to enable this behavior is role prompting, where the model is instructed to assume a specific identity or function, guiding its behavior throughout the interaction [33].

One influential example of using an LLM as an Agent is the ReAct framework proposed by Yao et al. [84], which demonstrates how LLMs can interleave reasoning and acting by generating intermediate thoughts and utilizing external tools as needed to solve tasks. The ReAct framework builds on Chain-of-Thought reasoning and few-shot prompting but introduces a distinct approach that enables language models to act more like interactive agents. While Chain-of-Thought prompts guide models to reason step by step, they are typically limited to static settings without external interaction. ReAct extends this by interleaving reasoning steps with tool-using commands and environment feedback [84]. This loop enables the model to adjust its reasoning in response to new information. Like Chain-of-Thought, ReAct uses few-shot prompting to demonstrate the desired format.

```
Task: Where was the actor in Inception born?

Thought: I need to find out who starred in Inception.
Action: Search["Inception cast"]
Observation: Leonardo DiCaprio is the lead actor.

Thought: Now I need to find out where Leonardo DiCaprio was
   ↪ born.
Action: Search["Leonardo DiCaprio birthplace"]
Observation: He was born in Los Angeles, California.

Thought: I now have the answer.
Action: Finish["Leonardo DiCaprio was born in Los Angeles,
   ↪ California."]
```

Figure 2.1: An example of the ReAct framework in action. The language model alternates between reasoning (Thought), taking environment-interacting steps (Action), and incorporating feedback (Observation) before producing a final answer.

Agent-based use of LLMs introduces opportunities for enhanced modularity, interpretability, and scalability in system design. It enables decomposing complex tasks into specialized components and coordinating them through orchestration patterns, which are described in the next section.

2.3.4 Workflow Patterns and Agent Orchestration

In recent research and applied systems, several agent orchestration patterns have emerged that extend the capabilities of individual LLMs by enabling coordinated, multi-step reasoning workflows.

Promptchaining: is a technique that decomposes complex tasks into smaller, manageable sub-tasks, each executed by an independent run of an LLM. The output of one step serves as the input for the next, creating a sequential workflow [80]. The main benefit is improved accuracy, achieved by simplifying each LLM task, even if it results in higher latency.

Parallelization: is a workflow technique in which multiple LLMs handle sub-tasks independently and in parallel, with their outputs aggregated later [6]. In this approach, sub-tasks can be assigned identically to multiple agents, enabling a voting mechanism to consolidate results. Alternatively, each agent can be assigned a unique sub-task, allowing individual models to focus on specific aspects of the overall task. This method enhances specialized attention to each component of the workflow.

Orchestrator: A workflow technique where a central agent breaks down tasks and delegates them to other agents, who act as workers [6]. An agent then synthesizes the outputs before being returned to the user. This workflow offers greater flexibility than the previously mentioned approaches, especially when tasks are not clearly defined.

Routing: When tasks can be classified into distinct categories, a routing workflow can be employed. In this setup, a classifier agent categorizes the given task and assigns it to the appropriate agent based on its classification [6].

Evaluator - Optimizer: This workflow is useful when the output follows well-defined criteria and iterative refinement leads to better results. In this approach, one agent generates a response, while another agent evaluates it and provides feedback if the response does not meet the specified criteria. This process continues iteratively until the response satisfies the given criteria [6].

While these orchestration techniques offer modularity and specialization, recent research has highlighted that they come with challenges associated with multi-agent LLM systems. Pan et al. [53] introduce the Multi-Agent System Failure Taxonomy (MAST), which categorizes 14 types of failures across three primary dimensions: specification issues, inter-agent misalignment, and task verification failures. These failure modes commonly emerge in orchestrated workflows, where agents must coordinate, pass intermediate outputs, and reason collectively. For instance, step repetition, conversation resets, and misalignment between reasoning and actions are frequent when chaining agents in sequential workflows. Errors in early stages often propagate downstream, and superficial or absent verification steps can cause incorrect outputs to go undetected. Importantly, the study suggests that these failures often stem from shortcomings in system and organizational design, for example, poor role specification or coordination protocols. As such, improvements in base

model performance alone are unlikely to resolve the full range of observed issues.

2.3.5 Reasoning Models

Recent advancements in the development of LLMs have led to models incorporating many of the techniques mentioned above within their own structure. OpenAI's o1 marked a turning point in the development of large language models, introducing explicit optimization for multi-step reasoning through reinforcement learning on chain-of-thought trajectories [29]. This shift laid the groundwork for more advanced reasoning models, such as OpenAI's o3 and Anthropic's Claude 3.7 Sonnet, which incorporate many of the prompting and orchestration techniques discussed in earlier sections into a unified, end-to-end policy. These models are fine-tuned on data that combines natural language reasoning with structured tool use, allowing them to autonomously decide when to invoke tools and how to integrate external observations into their internal thought processes [52, 8].

For instance, o3 is trained on data that includes interactions with external tools such as web search, Python code execution, image analysis, and file handling. This allows the model to autonomously decide when to invoke a tool and how to incorporate the resulting information into its ongoing reasoning process. Similarly, Claude 3.7 implements an "extended thinking" mode, in which the model is explicitly incentivized to generate coherent, multi-step reasoning sequences before providing an answer [8].

Through these mechanisms, models like o3 and Claude 3.7 effectively internalize techniques such as Chain-of-Thought prompting, task decomposition, ReAct-style reasoning-action loops, and role prompting. As a result, simpler natural language prompts can initiate complex, multi-step workflows involving planning, tool use, and decision-making without relying on external orchestration frameworks.

2.4 Challenges with Integrating LLMs

The development and implementation of LLMs into real-world applications present engineering challenges. High computational demands, scalability issues, data privacy concerns, and ethical considerations all contribute to the complexity of LLM-based systems.

2.4.1 High Computational Requirements

LLMs require large amounts of computational resources for both training and operation. Advanced models, such as GPT-4o, are estimated to contain over a trillion parameters [62], which requires powerful GPUs or TPUs to process them efficiently. Training these models often takes weeks or months and involves thousands of computing units working together. Even after training, running these models remains resource-intensive due to the complexity of their architecture. [50]. GPT-4.5, for example, is a very large and resource-intensive model compared to GPT-4o and

therefore more expensive to use [51].

In contrast, smaller models are easier and computationally cheaper to run, making them more suitable for applications where compute power is a constraint. This typically comes at the cost of reduced performance and capabilities. Many state-of-the-art models are released in multiple size variants (e.g., 3B, 7B, 13B), allowing developers to select a configuration that balances efficiency and task performance, for example, with LLaMA [70]. Techniques such as quantization and pruning are employed to reduce memory usage and computational requirements during inference. Quantization involves reducing the precision of model weights, often from 32-bit floating-point to 8-bit or lower, thereby enabling faster inference and smaller memory footprints, with minimal performance degradation [22]. Pruning removes parameters deemed less important for the model's predictive accuracy [39]. These optimization methods make it feasible to deploy language models on edge devices or consumer-grade hardware.

2.4.2 Data Security and Privacy

A major concern with integrating LLMs into financial workflows is potential data security and privacy issues. From a software engineering perspective, LLM integration introduces several challenges that must be considered during system design, deployment, and maintenance.

A potential issue is the risk of LLMs unintentionally learning from the sensitive dataset. Models trained on sensitive data can, under certain conditions, regenerate and output the sensitive data through prompt-based attacks[15]. Furthermore, attacks such as model inversion or membership inference attacks pose additional risks to data confidentiality. These attacks target the model's behavior and extract private training data. This creates significant software engineering challenges as secure systems must be designed with safety measures such as input sanitation, data minimization, and clear audit trails to reduce the risk of exposing private information. Robust security protocols should be adopted during the deployment phase, such as including differential privacy techniques, encryption during data transmission, and secure API gateways [21].

2.4.3 Hallucinations and Contextual Understanding

While LLMs offer promising capabilities for automating complex reasoning and analytical tasks, their use introduces challenges that can affect the factual accuracy and reliability of generated outputs. The accuracy of financial insights is important, as errors or misleading interpretations can have consequences for decision making, knowledge, and understanding of the company. While LLMs are adept at processing large volumes of textual and numerical data, they are also prone to hallucinations, generating plausible but incorrect information [11]. This issue arises because LLMs rely on probabilistic text generation rather than direct fact retrieval, making them vulnerable when outputs need to be highly accurate.

Although general-purpose LLMs perform well across a wide range of tasks, they often struggle with contextual understanding in specialized domains [24] found that GPT-3.5 frequently identified concepts with limited relevance due to its generalized training, whereas a domain-specific model was better able to interpret nuanced context and provide more precise classifications. This suggests that general LLMs may lack the depth of contextual sensitivity required for accurate domain-specific analysis [24]. Prior studies have reported variability in LLMs' ability to extract meaningful financial insights when local and global contextual nuances are present [2]. These findings indicate that additional mechanisms may be necessary to validate and refine AI-generated financial analyses within complex financial workflows.

3

Related Work

This section reviews literature relevant to the use of LLMs in financial and software engineering contexts. It is divided into two main parts. The first part focuses on the application of LLMs in the financial domain, including domain-specific models, prompt engineering techniques, and emerging system architectures. The second part covers software engineering perspectives on building and maintaining LLM-based systems, including design frameworks, development practices, and operational challenges. Together, these topics provide the foundation for identifying the research gap addressed in this thesis.

3.1 Large Language Models in Finance

The capabilities of LLMs have enabled their application in various financial analysis tasks, including text summarization, information extraction, sentiment analysis, and financial forecasting [35]. The versatility of LLMs for different tasks has led to the development of specialized models and architectures to address domain-specific challenges. Ongoing research into techniques like prompt engineering and multi-agent frameworks aims to enhance their performance in financial systems further.

3.1.1 LLMs for Financial Tasks

Several LLMs have been developed for financial tasks. One of the first examples is FinBert [27], which is built on Google's BERT architecture [18]. These are a set of models that are built using only financial data or combined with general domain data. FinBert is designed for structured NLP tasks such as sentiment analysis, information extraction, and classification. Another notable example is BloombergGPT [78], pre-trained on a massive set of financial data combined with general knowledge, which has demonstrated the potential for text generation, question and answering, text summarization, and financial forecasting within the domain.

Additionally, research has explored models through instruction tuning for financial tasks. Instruction tuning involves training models on financial datasets with instruction sets for specific tasks [87]. For example, Zhang et al. [86] demonstrated that instruction tuning can significantly enhance LLMs for financial sentiment analysis. Their model, Instruct-FinGPT, based on LLaMa, outperformed FinBERT and

GPT-3.5 across multiple datasets in terms of accuracy and F1 score. Advancements included improved numerical sensitivity, enabling better sentiment interpretation of financial indicators, and enhanced contextual understanding, which facilitated accurate sentiment prediction in nuanced scenarios. Other notable examples using this approach include FinGBT [82] and PIXIU [81].

3.1.2 Enhancing LLMs with Prompt Techniques

Zhang et al. [86] demonstrate that instruction tuning combined with Retrieval-Augmented Generation (RAG) significantly enhances LLM performance in financial sentiment analysis. RAG provides the model with external contextual data, thereby addressing some of the LLM's limitations in understanding complex financial language. This approach achieves notable improvements, with accuracy and gains in the F1 score of 15–48%.

The rapid development of the capabilities of general-purpose models give them the potential to perform effectively within the financial domain. This potential is further amplified when combined with prompt engineering techniques such as Chain of Thought (CoT), which have proven effective in guiding LLMs toward more accurate and transparent reasoning processes. The effectiveness of this was demonstrated in [32]. Their study compared human analysts with GPT-4 [50], a general-purpose model, to predict future earnings using two approaches: simple prompting and CoT prompting. Without CoT, GPT-4 achieved an accuracy of 52.33%, marginally better than human analysts at 52.71%. However, with CoT prompting, the accuracy of GPT-4 increased significantly to 61.45%, outperforming human analysts. Notably, GPT-4 achieved this without access to the narrative context typically available to human analysts, highlighting the potential of CoT to improve LLM reasoning capabilities.

Research has also focused on evaluating LLM summarization capabilities in the financial domain. Yang et al. [83] conducted a comprehensive benchmark comparing GLM-4, Mistral-NeMo, and LLaMA 3.1 (70B variants) on financial report summarization tasks. Their evaluation combined traditional metrics (ROUGE, BERTScore) with an LLM-based qualitative review, assessing dimensions such as accuracy, informativeness, and coherence. The results showed that while models like LLaMA 3.1 achieved higher accuracy and coherence scores, they incurred significantly longer processing times. The study also found that model size alone does not guarantee better performance, highlighting the importance of prompt design, domain fit, and system constraints.

3.1.3 Architectural Innovations for Financial LLM Systems

To enhance the capabilities of systems utilizing LLMs in the financial context, research is also exploring the development of architectures, such as multi-agent frameworks. Mixture of experts [43] is one approach, where individual agents function as specialized experts, handling their distinct tasks to enhance reasoning, generalization, and task-specific performance [14]. This design allows for increased model

capacities without the need for increased computational demand. This is achievable by only activating the relevant expert during inference.

Yu and Tiwari [85] propose FinTeamExperts, a framework using a Mixture of Experts (MoE) approach with large language models (LLMs) customized for financial analysis. It employs three specialized models that simulate distinct roles: Macro Analysts, Micro Analysts, and Quantitative Analysts, each trained on domain-specific data. Through a two-phase training process, pre-training on role-specific corpora, and instruction-tuning with routing gates, therefore integrating diverse expertise for financial tasks. Upon evaluation on public datasets, the framework outperformed comparable and larger models.

3.1.4 Integration of LLMs in the financial domain

Tavasoli et al. [68] present a framework that covers some core challenges financial institutions face when deploying LLMs. Their takeaway is that successful and responsible adoption of LLMs in finance requires navigating several technical, regulatory, and strategic factors. Through a six-part framework, the authors outline sequential decisions institutions must make: 1 determining whether an LLM is necessary over simpler methods, 2 establishing robust data governance under regulations such as GDPR, 3 implementing targeted risk management practices including explainability, human-in-the-loop oversight, and adversarial testing, 4 embedding ethical oversight to ensure fairness and transparency, 5 assessing the return on investment from both tangible and intangible perspectives, and 6 choosing an appropriate implementation strategy [68].

The authors further discuss the tradeoffs that different implementation strategies imply. For example, open-source models offer greater transparency and flexibility but require significant internal resources and expertise. In contrast, proprietary models provide stronger performance out of the box at the cost of limited control and potential compliance risks. Similarly, in-house deployment supports strict data control but demands high technical capacity, whereas vendor-managed solutions ease operational burden but raise concerns over data exposure and vendor lock-in. In terms of model adaptation, the authors argue that prompt engineering is an agile and easy method but lacks reliability, full fine-tuning offers better accuracy at the cost of a high computational burden, and RAG improves factual accuracy through external references, however, with increased infrastructure complexity. cite tavasoli2025responsible.

3.2 Software Engineering for LLM-based systems

AI and machine learning have become increasingly integrated in software engineering, both as tools to improve development workflows and as components within software systems [61, 42]. AI-driven techniques are, for example, used in automated

testing, code generation, bug detection, software maintenance, improving software quality and developer productivity. However, as AI components become more deeply embedded in software products, they introduce new challenges related to system reliability, explainability, and risk management.

3.2.1 Software Engineering for AI

To better understand the role of AI in software engineering, efforts have been made to categorize AI applications. One such attempt is the AI-SEAL taxonomy by Feldt et al. [20], which classifies AI use based on point of application (development process, product, or runtime), type of AI, and level of automation. Their work highlights that AI integration at the runtime level introduces higher risks, requiring robust governance mechanisms to ensure system reliability and compliance.

Within the broader AI landscape, LLMs have emerged as a distinct category, primarily used to support software development tasks such as code generation, documentation, and debugging. However, research on LLMs as components within software systems remains limited [74]. The authors further underscore that there is a need for new software engineering practices. Specifically, it calls for treating prompts as software assets and developing robust testing and monitoring frameworks for systems that rely on LLM-generated outputs. Although studies on LLM-integrated applications from a software engineering perspective remain limited, the field is rapidly emerging.

3.2.2 Software Engineering for LLM-based applications

Chen et al. [17] propose a feature requirement framework for developing applications based on LLMs in industry to address challenges with implementing LLMs in an industry setting. Feature requirements need to be addressed because of the limitations of current LLMs, coming from hallucination, the black box nature, and weak specialized knowledge.

Studies concerning the architecture of LLM-based systems are a growing area of research. Lu et al [38] propose a taxonomy of foundation model-based systems that outlines key architectural considerations across three categories. First, the pre-training and adaptation dimension distinguishes approaches such as in-context learning, fine-tuning, and distillation, each carrying trade-offs in control, accuracy, and infrastructure requirements. Second, the system architecture design dimension introduces the idea of foundation models as architectural connectors that serve roles such as communication, coordination, and conversion across system components. This framing positions LLMs as middleware capable of orchestrating complex, multi-step workflows. Third, Responsible AI by design dimension presents mechanisms for managing risk and ensuring compliance, including verifiers and prompt guardrails. An insight from their work is that in-context prompting alone is often insufficient for tasks requiring high output fidelity. Architectural patterns such as prompt chaining, task decomposition, or verifier modules are necessary, where interpretability, traceability, and formatting consistency are essential. The taxonomy also draws attention

to deployment trade-offs between external and sovereign models, a distinction that is especially relevant in data-sensitive and regulated environments [38].

Expanding on the architectural considerations, Mahr et al. [40] propose a reference architecture for deploying LLM applications in constrained environments, with a focus on addressing data security, limited connectivity, and infrastructure constraints. The architecture defines five stages: data preparation, model strategy selection, model evaluation, deployment, and prompt engineering. It outlines deployment options suitable for environments with strict data protection requirements, including on-premises and edge-based solutions. The component of the model strategy includes alternatives such as parameter-efficient fine-tuning (PEFT), full fine-tuning, and retrieval-augmented generation (RAG), each with different trade-offs in resource usage and control. The paper also incorporates LLMOps practices to support ongoing evaluation and adaptation, which are relevant for maintaining reliability and compliance in regulated domains.

Wang et al. [73] highlight key architectural and operational aspects of LLM-based autonomous agents. They propose a modular framework where LLMs are enhanced with memory, planning, and action modules, enabling them to handle complex workflows, recall context, and generate structured outputs. The study emphasizes that agents can acquire expertise through fine-tuning or prompt-based adaptation. In addition, they stress the importance of rigorous evaluation, combining objective benchmarks with subjective expert assessments to ensure reliability.

Research has also explored practical frameworks for developing LLM-based applications. One notable example is LangChain [69], an open-source library designed to facilitate rapid development of LLM-based applications. LangChain provides modular abstractions for interacting with various data sources and applications, enabling developers to build flexible and scalable LLM-driven solutions. Other examples of frameworks are Hugging Face's SmolAI Agents [28] designed for lightweight autonomous agents when computational resources are a constraint and AutoGen [77] facilitates multi-agent collaboration. These frameworks contribute to the growing ecosystem of LLM development and agent coordination

3.2.3 Engineering challenges

Regarding the challenges that developers face in real-world settings, a study that conducted 26 interviews with software engineers [54] identified four key issues. First, interacting with LLMs demands extensive prompt engineering, output modification, workflow development, and unpredictability management. Second, testing and validation remain challenging due to the absence of standardized metrics and the necessity for custom testing solutions. Third, ensuring safety, privacy, and compliance introduces concerns related to user consent and regulatory adherence in AI-driven actions. Finally, developers struggle with learning and tool integration due to the lack of centralized resources and the continuously evolving nature of workflows.

3.3 Summary and gaps

Existing research on LLMs in the financial domain has primarily focused on enhancing model performance for specific tasks such as sentiment analysis, summarization, and forecasting. Techniques like instruction tuning, domain-specific pretraining, prompt engineering, and architectures have been explored to enhance capabilities. However, these efforts are often evaluated in isolation and do not extend to a real integration of LLMs into financial workflows.

In parallel, software engineering research has begun to explore architectural frameworks, development practices, and operational challenges for LLM-based applications. While this work provides valuable insights, much of it remains general-purpose, is mainly theoretical, and lacks application-specific depth in domains like finance.

As a result, a gap exists at the intersection of LLMs' capabilities and system-level implementation. Specifically, there is limited research on how LLMs can be designed into financial environments. This thesis addresses this gap by investigating the practical challenges of developing and deploying LLM-driven artifacts in financial reporting workflows to inform future system design and implementation practices.

4

Method

In relation to the ABC framework [65], the thesis is conducted as a field study, which means the research is taking place in a real-world setting without actively controlling or altering the environment. This approach facilitates a nuanced understanding of the challenges and opportunities associated with integrating LLMs into financial workflows. However, it may come at the expense of precision and generalization compared to studies conducted in a more controlled environment. Within the context of field studies, this thesis could be categorized as an exploratory field study following the definition by Höst et al. [60]. The goal is to generate new insights, identify key challenges, and propose hypotheses for future exploration. To achieve this, the study applies Design Science Research (DSR) [25], which blends practical problem-solving with knowledge generation through iterative design and evaluation.

4.1 Methodology

The study follows the Design Science Research (DSR) framework proposed by Pffeffers et al. [55]. Design Science Research is particularly suitable for research that aims to develop, implement, and evaluate artifacts, making it a suitable approach to explore how LLMs can be utilized within financial analysis and reporting. Unlike traditional empirical research, which seeks to describe and explain phenomena, DSR focuses on solving problems by designing and evaluating solutions in real-world environments [55]. Given that this thesis aims to develop and evaluate an AI-driven solution to identify its challenges, DSR provides a structured process to guide the design, implementation, and validation of the proposed system. The framework proposed by Peffers et al. [55] consists of the following six phases:

1. **Problem Identification & Motivation:** This stage defines the challenges in the analysis of financial reports and explores the motivations for automation. It also identifies the reported difficulties involved in integrating LLMs into financial workflows and research to overcome them.

2. **Objectives of a Solution:** This stage examines relevant use cases within financial control to identify the challenges associated with integrating LLMs into existing workflows. The aim is to clarify what is feasible within the given context and to establish what would support financial workflows.

3. **Design & Development:** Two LLM-based solutions are developed to address the identified challenges. This process includes selecting suitable models, architectures, and tools, and implementing the solution in the context of financial reporting and analysis. The implementation also serves as a means to explore practical challenges that may arise when integrating such a system into real-world financial workflows.

4. **Demonstration:** The developed solutions are tested in a representative setting using realistic financial data. The output is used to compare the performance of the system with traditional methods. This step will iterate with Step 3, as insights gained will inform and guide further development.

5. **Evaluation:** The system is evaluated based on its accuracy, efficiency, interpretability, and usability within financial workflows. Feedback is gathered from domain experts to generate insights into the challenges of integrating LLMs in practice, complementing the technical assessment with experiential and contextual perspectives.

6. **Communication:** The findings and implications of the developed system are presented to academic and industry audiences. This includes a discussion of the results, limitations, and suggestions for the future direction in AI-driven financial reporting and analysis.

An overview of the research methodology is illustrated in 4.1, which maps each methodological step to the design science research process while highlighting the practical insights gained during the phase.

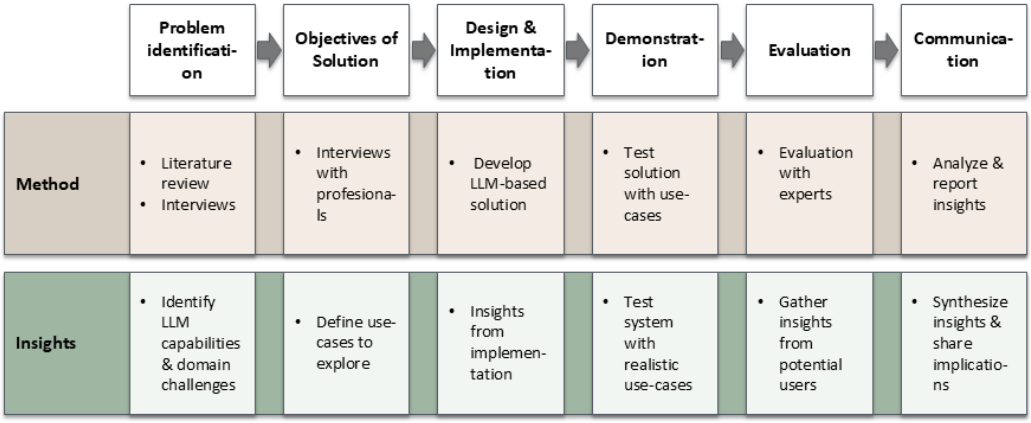

Figure 4.1: Alignment of DSR steps with the specific methods applied and the insights each step is intended to generate.

4.2 Research Setting

This study is conducted in collaboration with the finance department at Getinge AB, a Swedish medtech company specializing in healthcare and life sciences solutions. Getinge provides a wide range of products and services and is publicly listed on the Nasdaq Stockholm stock exchange. The company operates globally, with a presence in over 40 countries, focusing on areas such as intensive care, operating rooms, sterile reprocessing, and infection control.

The finance department at Getinge operates within a sensitive financial environment, handling confidential data that demands a high level of accuracy and reliability in reporting and analysis. While an internal LLM-based document retriever is in use, its functionality is limited to retrieving content from specific predefined documents, with no broader integration or reasoning capabilities.

4.3 Literature review

The first part of the study involved conducting a literature review, following the guidelines proposed by Keele [31] for software engineering purposes. The review was conducted in three phases: planning, conducting, and synthesizing the results. The review aimed to gain an understanding of the current research on LLMs within the financial domain and software engineering for LLM-based systems.

A planning phase was conducted to refine the research scope and methodology. This involved reviewing existing surveys and literature to identify research directions, exploring industry reports on the application of LLMs in finance, and conducting broad keyword searches in academic databases. These efforts helped structure the literature review, which served to understand the current state of research and practices regarding the integration of LLMs into financial applications, as well as understanding the current state of software engineering research related to AI-driven applications. The insights gained during this phase informed the study design and supported the formulation and refinement of the thesis's main research questions.

To guide the review, five themes were defined. These included current use cases of LLMs in financial applications, reported limitations and opportunities, and existing engineering frameworks and development methodologies. Additionally, the review considered common software engineering challenges encountered when integrating AI-driven systems into industry settings. These themes helped structure the literature synthesis and provided a foundation for understanding the broader context of LLM integration in financial workflows.

The search was conducted in IEEE Xplore, ACM Digital Library, ScienceDirect, and Google Scholar. Third, the findings were synthesized to identify key patterns and trends within the literature. To the extent possible, peer-reviewed articles were prioritized. However, given the rapid advances in research related to the field, many recent studies have not yet undergone peer review. Additionally, a substantial

portion of relevant research originates from industry. To ensure the most up-to-date information, non-peer-reviewed papers and online articles were also considered.

4.4 Overview of Research Activities

The research activities in this thesis are structured into a series of exploratory and design-oriented phases, which are referred to as cycles in this study. Following the methodological framing and literature review, Cycle 0 explores the problem space through engagement with domain experts, aiming to identify relevant use cases and uncover potential challenges associated with designing LLM-based solutions into financial workflows. Based on these insights, two design cycles were conducted to implement a solution. Finally, an evaluation was conducted together with experts in the domain to asses and compare the solutions.

4.4.1 Cycle 0: Domain Exploration

Cycle 0 served as the exploratory phase of the study, aimed at gaining a contextual understanding of the department and its operations and identifying potential opportunities for LLM-based automation. This was achieved through two focus group sessions followed by a task analysis. The focus groups were conducted with business controllers to uncover high-effort cognitive tasks and understand current practices in generating narrative financial commentary.

The insights gained from this cycle informed the formulation of system objectives and design considerations for the LLM-based artifacts developed in Cycles 1 and 2.

An overview of the conducted activities and participating stakeholders is presented in Table 4.1 below. A more detailed account of this process is provided in the subsequent chapter.

Activity	**Participants**	**Duration**	**Location**
Focus Group	3× Business Controllers	1 hour	On-site
Focus Group	3× Business Controllers	1 hour	On-site
Workflow Walkthrough	2× Business Controllers	1 hour	On-site

Table 4.1: Overview of exploratory study activities

4.4.2 Cycle 1 and 2 Design and Development

Following the initial exploration, two design and experimentation cycles were carried out to investigate the applicability of two different LLM implementation strategies for automating financial analysis tasks.

Cycle 1: focuses on developing and testing a locally hosted solution using lightweight open-source models. The procedure involved implementing a modular, multi-agent

workflow where agents processed structured financial data and generated summarizing commentary.

Cycle 2: explores the use of a commercially available LLM (OpenAI's GPT-4o) accessed via API. The procedure mirrored that of Cycle 1 in terms of workflow and summarization tasks, but prioritized the evaluation of output quality, interpretability, and ease of integration.

Together, these cycles provided insights into different approaches to LLM design in real-world financial reporting contexts.

4.4.3 Evaluation

To evaluate the outputs from both artifacts, AI-generated financial summaries were compiled and shared with domain experts working in financial reporting. The initial plan was to conduct a live, structured evaluation session in which the summaries would be assessed collaboratively. However, this format proved impractical, since we wanted to assess multiple aspects such as the content's clarity, accuracy, language used, and potential integration into existing workflows.

The evaluation format was therefore adapted, participants were given access to the summaries and a set of guiding questions, which they could review independently at their own pace over the course of a week. A follow-up session was then held to discuss their reflections, during which follow-up questions and requests for clarification could be addressed.

Activity	**Participants**	**Duration**	**Format**
Initial review	2× Business Controllers	1 hour	On-site
Independent review of summaries	2× Business Controllers	N/A	N/A
Follow-up discussion session	2× Business Controllers	1 hour	Online

Table 4.2: Overview of evaluation activities

To guide the evaluation, a set of open-ended questions is grouped under six key themes. These themes focused on areas such as language clarity, content accuracy, perceived usefulness, integration into participants' workflows, and trust in AI-generated content. The themes and associated guiding questions can be found in the Appendix A.7.

5

Design Iterations and Evaluation

The following section presents the design, implementation, and evaluation of a solution designed to automate financial commentary using LLMs. It begins by outlining the exploratory work conducted to identify use cases for LLM-based automation within the financial reporting process. This is followed by a description of the selected use cases and observations made.

Subsequent sections detail the development of two prototypical solutions, first a local LLM-based workflow and a cloud-based alternative using GPT-4o. This is followed by observations made from experimentation.

The chapter concludes with an evaluation of both prototypes, highlighting findings across critical dimensions such as, accuracy, usability, and trustworthiness.

5.1 Cycle 0: Domain Exploration

To identify the relevant areas for LLM-based automation within the financial department of Getinge AB, two exploratory meetings were conducted with representatives of the business control team. The initial session introduced the thesis scope and aimed to establish a foundational understanding of the department's operations.

The discussion then focused on identifying processes characterized by high cognitive workload and frequent manual interpretation. These were deemed particularly suitable for LLM integration. Through these dialogues, the task of preparing financial statements was identified as a promising entry point. Other areas of interest included synthesizing insights from unstructured information sources and performing financial forecasting.

To deepen the understanding of reporting workflows, a follow-up session was held to map the steps involved in generating narrative commentary for financial statements. Based on these discussions, three specific use cases were defined as suitable for initial implementation and evaluation.

These use cases represent scenarios that were both considered contextually relevant and technically feasible within the scope of the study. They serve as concrete points of departure for exploring the practical challenges of designing LLM solutions in

the financial reporting context. The selected tasks reflect typical areas of manual analysis: identifying deviations in financial metrics, summarizing key drivers behind those deviations, and producing structured qualitative commentary.

5.1.1 Objectives of the Solution

To clarify what an LLM-based solution should achieve, the reporting process was analyzed in collaboration with two business controllers. The aim was to identify the underlying data sources, determine the relevant information, and clarify the cognitive steps involved in the process. Emphasis was placed on capturing the thought process typically employed by business controllers when formulating these summaries, as this reasoning is what the LLM-based solution is intended to emulate and support.

The process begins by detecting significant changes in key metrics, such as order intake or net sales, for specific product segments. These changes are then traced to underlying drivers by examining performance across product lines and geographic regions. Finally, business controllers assess the relative importance of these drivers and distill the findings into concise narrative summaries for inclusion in financial reports.

The goal of the LLM-based system is to emulate and support this workflow by automating the identification, synthesis, and articulation of key insights. This involves mimicking not just the data processing, but also the judgment applied in prioritizing and articulating findings.

5.1.2 Identified Use Cases for LLM-Based Automation

The following is a description of the workflow that the LLM is intended to facilitate, outlining its objectives, key insights, process, and evaluation criteria.

UC1: Generating Summarizing Comments for Income Statements

Objective: Automatically generate qualitative comments on the income statement presented alongside the financial statement for a given period.

- **Key Insights:** Identify increases or decreases in profits, OPEX, and business area-specific trends.
- **Process:** The data is analyzed and compiled into a CSV file, highlighting deltas of changes in Revenue and Costs. Based on this data, the LLM model should infer key highlights and drivers of changes during the period.
- **Evaluation:** Compare AI-generated insights against human analysis.
- **Example:** Cost related to [category] is the main driver for increases in OPEX.

UC2: Summarizing Comments in Changes in Order Intake

Objective: Generate a summary of increase and decrease in order-intake divided into business-area, product-area and find the drivers within the product-lines.

- **Key Insights:** Highlight notable drivers for the changes in order-intake between two periods in a qualitative summary. For example, the current and previous months.
- **Data Source:** Structured order intake data containing numerical figures for two periods, along with categorical information such as business areas, product lines, and regions, extracted either from a database or provided in a CSV format.
- **Evaluation:** Compare AI-generated summaries to human-generated insights.
- **Example Output:** [Product Line] in [Region] as main growth driver. [Product Line] up in all regions. [Product Line] increasing, mainly in [Region]. Decrease from [Product Line] party products in [Region].[Product Line] decreasing in [Region].

UC3: Summarizing Comments in Changes in Net Sales

Objective: Generate a summary of increases and decreases in net sales, divided into business areas and product areas, and identify the key drivers within the product lines.

- **Key Insights:** Highlight notable drivers for the changes in net sales between two periods in a qualitative summary, for example, between the current and previous month.
- **Data Source:** Structured net sales data containing numerical figures for two periods, along with categorical information such as business areas, product lines, and regions, extracted either from a database or provided in a CSV format.
- **Evaluation:** Compare AI-generated summaries to human-generated insights.
- **Example Output:** [Product Line] in [Region] as main growth driver. [Product Line] up in all regions. [Product Line] increasing, mainly in [Region]. Decrease from [Product Line] party products in [Region]. [Product Line] decreasing in [Region].

5.1.3 Challenges with integrating LLMs in the financial context

A practical challenge observed early during the exploration of applying LLMs to financial analysts workflows was that the data used in their work comes in varied formats, requiring substantial data preprocessing. Financial information used in analysis and reports is collected from multiple sources and formats, including Excel

spreadsheets, PowerPoint presentations, ERP systems, and oral communication during meetings. These inconsistencies in data structure and terminology require some form of preprocessing to ensure that the input to the LLM is usable and appropriate for the task given to the model.

It became evident during this phase that any practical application of LLMs would require tailored preprocessing logic. Although the use cases later tested in the study relied on structured data and custom scripts to extract and format this data into prompt-ready text, this approach was closely tied to the structure and content of that particular case. The early exploration suggested that similar preprocessing efforts would likely be necessary for other types of reports or financial analyses, each requiring its own logic to handle differences in layout, terminology, and contextual focus.

An obstacle when designing LLMs into financial analysis workflows is the sensitive nature of the data involved. Rules that the department needs to adhere to prohibit sharing non-public financial information to third parties, as in the case of LLMs hosted by a third party. As a result, experiments involving these models had to rely on synthetic or anonymized data to ensure compliance. Importantly, this limitation is not unique to the research setting, it reflects the operational reality for employees as well, who are restricted from sharing sensitive information externally due to regulatory requirements and internal governance policies.

These constraints were the reason Use Case 1 was not pursued in the study, as the data used for the task comes in varied data formats, and making a realistic transformation on the data would be infeasible without making the data non-realistic. Instead, use cases 2 and 3 were continued, as they were more practical to implement under the given constraints.

5.2 Cycle 1: Local setup

In response to the challenges identified in the initial exploration, particularly on restrictions on sharing sensitive data externally, the first cycle focused on testing a workflow using local LLMs. This would support data privacy by keeping all processing on-site and is potentially lightweight enough to operate on standard office hardware, a consideration given the departments lack of dedicated infrastructure. It also presents a cost-effective alternative to commercial solutions. Furthermore, if the tasks assigned to the system can be handled effectively by a simpler mode, there is little justification for adopting a more complex solution, especially one that is more computationally demanding, costly, and introduces additional challenges concerning deployment.

Use Cases 2 and 3, which focus on summarizing changes in order intake and net sales, respectively, across business and product areas, were selected for continued exploration. Both relied on structured datasets that could be anonymized or synthetically replicated, making them suitable for local experimentation while respecting

data sensitivity constraints.

5.2.1 Data pre-processing

Before either solution could generate summaries, it was essential to extract and structure the relevant data in a format that the language models could interpret effectively. The raw data was initially provided as CSV spreadsheets containing thousands of entries. To prepare this data for model input, we used Python and the Pandas library to carry out pre-processing steps, including calculating the differences in net sales or order intake between two periods, grouped by product line and region. To ensure the input was both manageable and meaningful, the dataset was then condensed into a focused and structured summary of key trends.

While Excel files were used in this prototype for convenience and accessibility, a real-world implementation would likely rely on direct integration with a database or business intelligence system. Table 5.1 shows an example of the data fed into the model for interference.

Table 5.1: Modified Example Data as input to model.

Product Line	Region	Total Difference	Contribution (%)
CCVE	EMEA - EMEA	-5,619,037.0	-5.48
CCVE	APAC - Asia/Pacific	-17,886,827.0	-16.49
CCVE	AMER - Americas	89,963,264.0	65.35
CCSE	EMEA - EMEA	32,582,792.0	34.23
CCSE	APAC - Asia/Pacific	18,493,137.0	17.90
CCSE	AMER - Americas	7,142,562.0	6.85
CCOT	APAC - Asia/Pacific	654,822.0	0.49
CCOT	EMEA - EMEA	-2,874,312.0	-3.15
CCOT	AMER - Americas	-12,589,374.0	-11.78
CCHH	EMEA - EMEA	3,405,672.0	2.78
CCHH	APAC - Asia/Pacific	2,598,442.0	2.55
CCHH	AMER - Americas	-1,078,936.0	-1.02
CCHD	EMEA - EMEA	-2,874,098.0	-2.71
CCHD	APAC - Asia/Pacific	-6,945,325.0	-6.51
CCHD	AMER - Americas	-785,214.0	-0.73
CCAA	APAC - Asia/Pacific	-1,678,432.0	-1.35
CCAA	EMEA - EMEA	11,230,874.0	9.99
CCAA	AMER - Americas	5,093,287.0	5.15

It lists the product lines and their corresponding regions, along with the aggregated difference in order intake or net sales between two periods ("Total Difference"). The column 'Contribution (%)' indicates the relative impact of each product line and region on the overall change, guiding the model on the importance of individual drivers.

5.2.2 Local LLM experimentation

To facilitate experimentation with models deployed on local machines, we used Ollama [49], a tool that provides a setup for running and interacting with various open-source LLMs. Initially, two smaller models, LLaMA3.2 and Qwen2.5, were tested, each with approximately 3 billion parameters, to explore their suitability. As experimentation progressed, we moved on to models with around 8 billion parameters. However, inference at that scale became significantly slower due to hardware limitations, as the models were running solely on CPU without GPU acceleration. A summary of the models tested in the experimentation can be found in Table 5.2

During early testing, single-prompt inputs were explored, in which instructions and data were provided to the model within a single input. This approach was found to be ineffective, leading to hallucinations, poor structural coherence, and factual inaccuracies. As a result, a multi-agent workflow was employed. In this design, distinct roles were assigned as specific agents, each tasked with handling a specific sub-task of the overall task. LLaMa3.1 was used in the later setups, since after the initial testing, it proved to be the most stable for the given task.

Model	Size	Reasoning	Quantized	Context Window	Type
Qwen 2.5	3B	No	4-bit	32k	Instruct
LLaMA 3.2	3B	No	4-bit	128k	Instruct
Qwen 2.5	7B	No	4-bit	32k	Instruct
LLaMA 3.1	8B	No	4-bit	128k	Instruct
DeepSeek-R1	8B	Yes	4-bit	128k	Distilled

Table 5.2: Overview of models tested in early stages.

5.2.3 Implementation of Multi-Agent Workflow

To implement the four-agent design, a Python-based pipeline was developed. Each agent was implemented as a modular function responsible for a specific task. The workflow followed a linear sequence, prompt-chained, where the output of each agent was passed as input to the next.

The core components of the implementation included:

- **Environment:** Python 3.11, with communication to the LLM handled through API calls to Ollama.
- **Prompt Templates:** Prompts were stored as templates and dynamically populated with relevant data.
- **Execution Logic:** A controller script coordinated the agents, invoking them sequentially and handling intermediate data transformation.
- **Logging and Validation:** Validation feedback was recorded for each execution to support iterative refinement.

The workflow follows a defined sequence:

1. **Data Interpretation:** The *Financial Data Analyst* (Agent 1) processes structured input data and converts it into natural language summaries, while also categorizing observed changes. This step serves to translate tabular financial data into a format that is more accessible for subsequent language-based reasoning, thereby improving the input quality for the downstream agents.
2. **Trend Analysis:** The *Business Analyst* (Agent 2) identifies significant trends, verifies consistency across product lines, and determines key influencing factors. By receiving pre-processed textual summaries from Agent 1, this agent is better equipped to perform analytical tasks that benefit from semantically processed input rather than raw data.
3. **Report Generation:** The *Report Writer* (Agent 3) compiles the analysis into a structured executive summary. The motivation behind this role is to ensure that the final output is clearly phrased, professionally formatted, and

aligned with expectations for reporting style and tone, thus making it suitable for direct use or minimal revision in financial reports.

4. **Validation:** The *Financial Validator* (Agent 4) assesses whether the final summary is consistent with the input data and flags for factual inaccuracies. The rationale for including this role is to introduce a verification step that enhances the reliability of the output. While the current setup only performs validation passively, future iterations could enable feedback loops, where errors detected by the validator trigger corrections upstream. However, further development in that direction was constrained by computational limitations.

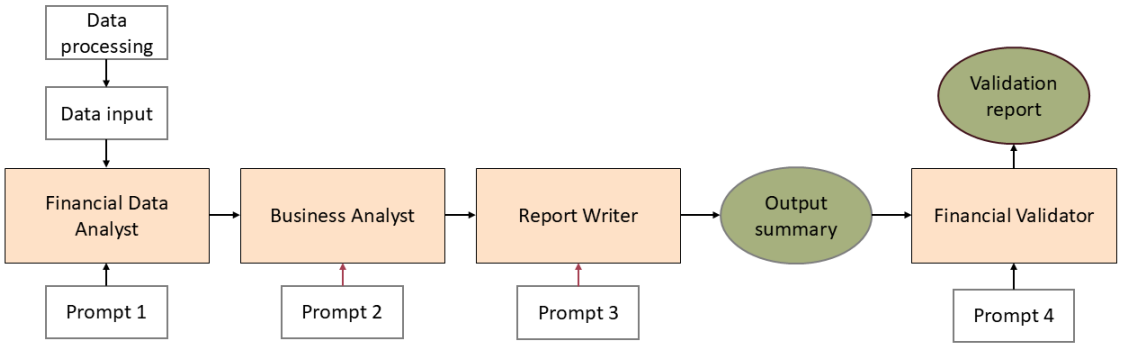

Figure 5.1: Figure illustrates the agent structure of the locally hosted LLM. Each box represents an agent with its own separate task, which it then sends over to the next agent.

5.2.4 Prompts

Each agent was guided using a structured prompt composed of five parts: role, context, task, rules, and an expected output format. This format adheres to best practices in prompt design by providing the model with clear instructions and relevant contextual information, including the task's purpose, what it should and should not do, and what constitutes a successful output. The prompt for Agent one is presented below, and the rest can be found in Appendix A.1.

5.2.5 Observations from Cycle 1

The experimentation revealed several technical and usability challenges.

The initial testing was conducted using smaller models with 3 billion parameters. These models could run on a standard computer without significant resource demands. However, they had to be ruled out for continued experimentation, as they consistently struggled with following instructions and exhibited a high degree of hallucination.

For instance, when using LLaMA 3.2 and prompting the model to summarize trends from a table of financial data, it frequently responded with a Python script intended

Step 1: Row-Level Summarization

Role: You are a financial data analyst summarizing trends for the order intake trends for different product lines and regions

---

Context: The dataset contains order intake changes across various product lines and regions between two periods. Each row represents the change, including its magnitude and the relative impact on the overall change.

Table: `{table_summary}`

Task:

- Process each row individually and convert the structured data into a natural language summary.
- Clearly indicate whether the change is considered a major or minor impact.
- Structure the output concisely while ensuring that key details are retained.

Rules:

- Use concise, structured sentences.
- Do **not** include raw numbers, just directional trends.
- Do **not** generate code.

Example Output Format: `[Product Line] in [Region] experienced a [major/minor] [increase/decrease].`

Table 5.3: Step 1: Row-Level Summarization

to calculate summary statistics, rather than providing a natural language summary. This behavior persisted even when explicitly instructed not to produce code and to instead give the response in natural language. The model often defaulted to describing how a script could be written, rather than directly providing the requested insights.

When using a chain of three language models, hallucination and not adhering to instructions became sometimes even more apparent. If one model in the chain deviates from the given instruction or hallucinates, that error is propagated and often amplified by downstream agents similar to the dynamics of a "whisper game." For instance, when an upstream agent is instructed solely to summarize observed trends in the data, it may also infer potential causes or suggest follow-up actions based on the trends. These unintended inferences are then treated as factual input by downstream agents, which incorporate and build upon them in their respective responses. Figure 5.2 illustrates a hallucinated output generated during such a chained interaction:

```
Summary
Each product line has distinct features that cater to the unique needs of
its respective region:
1.  CC Other (Central and Eastern Europe):
  - Features:  Customizable UI/UX design.
  - Benefits:  Improved usability through tailored interface elements.

2.  CC Advanced Monitoring Disposables (Central and Southern Hemisphere
Development):
  - Features:  Advanced analytics tools.
  - Benefits:  Enhanced data analysis for better decision-making.

These features ensure that users in each region get a more personalized,
effective product experience.  This approach helps maximize user
satisfaction and efficiency within their designated regions.
```

Figure 5.2: Example of hallucinated LLM output

In this case, the output demonstrates a hallucinated response that resulted from a breakdown in the chained interaction. While the original task was to produce a concise summary of order intake trends, one of the upstream agents appears to have gone off track, generating fabricated content about product features and regional benefits that were neither part of the input nor requested. This misinformation was then passed along, with downstream agents treating it as valid input and building upon it. This resulted in an output that was entirely fabricated and disconnected from the input data.

Hallucination-related issues persisted when using larger models (8 billion parameters), although they appeared less frequently during experimentation. While the generated outputs often aligned with the input data and the expert created summaries, one example can be seen in table 5.4, one consistent challenges was the overall stability of the system. In several cases, the system failed to identify some of the more obvious trends and deviated much from the expected format.

A challenge encountered was designing prompts that consistently guided the models to perform the intended tasks. Achieving the desired behavior, particularly when asking the system to identify relevant insights across multiple dimensions, proved complex. Similarly, crafting prompts that generalize well across variations in data distribution presented additional difficulties.

The main constraint for the experimentation with running models locally was the computing demand required for inference when the necessary computational power was not available. Although the models used in the experiments were relatively small compared to many other LLMs, they still imposed hardware requirements that exceeded the computational capabilities typically available on standard laptops. As a result, further experimentation with this setup was impeded, and alternative solutions were subsequently investigated.

Model-Generated Summary (Local LLM)	**Expert-Written Summary**
Ventilation was a main growth driver, with major increases in US and APAC. Service saw an upward trend across all regions, with minor increases in each region. Anesthesia also experienced an upward trend, mainly driven by a major increase in EMEA. In contrast, Other and Advanced Monitoring Disposables saw downward trends, with minor decreases in all regions.	Ventilation in US as main growth driver. Service up in all regions. Anesthesia increasing, mainly in EMEA. Decrease from Other 3rd party products in Canada. Monitoring Disposables decreasing in China.

Table 5.4: Example of model-generated summary compared to the corresponding one written by an expert.

5.3 Cycle 2: Cloud-based setup with GPT-4o

To experiment with a commercial solution developed using OpenAI's GPT-4o model. This implementation aimed to explore the opportunities and challenges of using a large-scale model for generating natural language summaries of financial data.

The environment for this implementation was based on Python 3.11, utilizing OpenAI's official Python SDK (version 1.0 or later) to facilitate interaction with the model. For the prompt design, we followed the same principles as in Cycle 1, to define the model's role and structure the task. These prompts were dynamically populated with context-specific data to ensure relevance and clarity. The same type of data pre-processing steps as used in Cycle 1 were employed in this iteration. To enhance readability and make the output suitable for direct inclusion in financial reports, the model's responses were post-processed to split the summary into individual sentences and compiled into a CSV sheet.

The GPT-4o-based workflow is structured as follows:

1. **System Role Definition:** The model is explicitly instructed to adopt the role of a financial analyst. This role prompts domain-specific behavior and constrains the model to maintain a professional and analytical tone aligned with financial reporting practices. The prompt emphasizes clarity, brevity, and factual accuracy, instructing the model to avoid speculation or numerical outputs.

2. **User Instruction and Input:** The user prompt follows a structured format inspired by instruction tuning techniques. It includes detailed task guidelines, style requirements, and formatting constraints, such as a maximum of four sentences and no numbers. It also draws inspiration from few-shot prompting by providing curated example outputs to guide the model toward the desired

style and structure. The data block is injected into the prompt in a clearly marked input section. During experimentation, prompts without example outputs were explored.

3. **Summary Generation:** The complete prompt is submitted to the GPT-4o API, which generates a concise, four-sentence qualitative summary. The model is guided to prioritize trends with the largest impact, identify consistent regional behaviors, and distinguish between positive and negative drivers while staying within strict output constraints.

4. **Post-processing and Output Export:** To improve readability, the generated summary is parsed so that each sentence is displayed on a separate line. The output is then saved to both a text file and a CSV file for further use in reporting or evaluation.

Prompt to GPT-4o

Context:
You are a financial analyst writing a financial report. Your task is to analyze and summarize changes in order intake data.

Instructions:
You must write a concise summary in at most four sentences to complement the numerical data in a financial report.
Focus on clear, direct language and only state what is evident in the data—no speculation or recommendations.
Do not include numbers. Focus on trends, main drivers, and regions as specified.

The data shows net sales changes segmented by product line and region, with relative contributions to the overall change in a business area.

- Identify whether the overall trend is an increase or decrease.
- Detect the main positive and main negative drivers.
- Detect if there are product lines where the direction of change is consistent across all regions.
- Write a concise final qualitative summary about the detected trends.
- Start by mentioning the trend that has the biggest impact overall.

Rules for the final summary:

- Maximum of four sentences.
- Do not speculate, do not include numbers.
- Do not include the overall trend in the summary since that is already understood.
- Use concise, factual phrasing as in the examples below.

Expected output examples:

- `All product lines up. [Product Line] in [Region] as main growth driver. [Product Line] increasing in [Region]. Increase from [Product Line] in [Region]. [Product Line] up in all regions.`
- `[Product Line] and [Product Line] decreasing in [Region]. [Product Line] in [Region] as main detractor, partly offset by [Product Line] in [Region].`
- `[Product Line] in [Region] as main detractor. Increase from [Product Line] in [Region].`
- `[Product Line] up in all regions. [Product Line] down in [Region]. [Product Line] up in [Region].`

Input:
`Data: {data_block}`

Table 5.5: Prompt for Generating Order Intake Summary

5.3.1 Observations from Cycle 2

Setting up this requires minimal setup and configuration during experimentation. In our tested use cases, summarizing financial trends, the model appears to capture most underlying trends without requiring the same level of task decomposition. An example of the generated and expert-written summary can be seen in Table 5.6. However, it sometimes misses important insights.

Model-Generated Summary	**Expert-Written Summary**
Beta Bags and Consumables in EMEA as main growth driver. Major increases in Service in the US and Fluid Pathway in the US. Sterilization in EMEA as main detractor, with significant decreases also in China and the Americas. Bio Reactors up in EMEA and the US, but down in China and APAC.	BetaBags increasing in all regions, primarily in EMEA. Service up in all regions. Increase from Bio Reactors in US and EMEA. Fluid Pathway up in all regions, mainly in US. Decline from Sterilization in all regions. Isolation down in all regions, primarily EMEA.

Table 5.6: Example summary from GPT-4o compared with the expert written one.

When provided with example summaries to guide format and tone, GPT-4o often mirrored these examples too rigidly, resulting in outputs that followed the structure but also occasionally ignored important variations in the input data. In contrast, when no format example was provided, the model captured the underlying trends more accurately but produced summaries that did not adhere to the desired format. To get the model to produce the desired output, clear and representative examples were necessary.

Deployment of a high-capacity model such as GPT-4o would be a challenge due to its significant computational requirements. On-premise deployment was not feasible within the constraints of the available hardware, necessitating the use of third-party hosting solutions. This setup introduced considerations related to data privacy, regulatory compliance, and dependence on external providers, which emerged as challenges when integrating LLMs in real-world financial workflows.

5.4 Evaluation

The evaluation compared the quality of two AI-based summarization approaches, one using GPT-4o and the other a local LLM solution to assess their effectiveness in supporting financial reporting tasks. Each model generated 20 summaries based on the same underlying financial data, which were then compared to corresponding human-written summaries used in practice. Feedback was gathered from two professional users with experience in financial reporting, who evaluated each model's output across five key dimensions: usefulness, accuracy, clarity, trust, and integration into existing reporting workflows.

5.4.1 Clarity and Language

According to the participants, both the GPT-4o and the local LLM models produced outputs that were professionally formulated and suitable for use in financial contexts. Participant 1 and Participant 2 both described GPT-4o's language as concise and aligned with the tone typically expected in financial reporting, particularly for presentation purposes. The phrasing was characterized as consistent with the style used by financial domain experts.

The local LLM was also described as maintaining a professional tone but generating more descriptive outputs. Participant 1 stated that "the local LLM offers a more descriptive summary that is easier to understand, but it occasionally includes unnecessary details." Based on this feedback, the local model's output was considered potentially suitable for internal use due to its format and level of detail.

Both participants observed that, in some cases, the presentation of information reduced clarity. GPT-4o was reported to occasionally include both "main drivers" and "main detractors" within the same summary section. In situations where a single, dominant factor was expected, this structure was perceived as potentially ambiguous. According to the participants, this occasionally made it difficult to determine whether the overall trend was positive or negative. They also noted that positive and negative developments were sometimes presented without a clearly defined structure. The following excerpt was cited by one participant as an example of this issue:

> *"Disposables ECLS in the US and EMEA as main growth drivers. Hardware increasing in all regions. Disposables Surgical Perfusion in EMEA as main detractor. Service up in the US and EMEA."*

Additionally, both models were noted to have a tendency to repeat themselves, mentioning the same product line multiple times within the same product area. Participant 1 commented: *"Both the local LLM and GPT-4o model sometimes include the same product line multiple times in a product area that I would not comment on myself, as the focus should be on the most significant changes."*

5.4.2 Accuracy and Reliability

Both participants identified aspects related to the accuracy and reliability of the summaries. Feedback indicated that the summaries sometimes omitted key information or included unnecessary details. Participant 1 summarized this observation as follows: "The summaries are helpful as a foundation, but they do not always explain the trends accurately. GPT-4o sometimes omits key drivers or mixes positive and negative changes in a way that makes interpretation difficult. Additionally, it had a tendency to misprioritize the most impactful changes."

Feedback related to the local LLM model suggested that while it captured broader trends, its descriptive style occasionally led to the inclusion of less relevant details. This was noted as potentially limiting the clarity of the key drivers, though it was

not necessarily characterized as inaccurate.

Regarding overall trust and reliability, both participants stated that manual review would still be necessary before using the summaries in formal reporting. According to participant 2, “The summaries still need manual input for the comments of the reporting,” and the outputs were described as more appropriate for internal use.

Both participants also noted a lack of transparency in how content was selected and structured by the models. Participant 2 remarked that the rationale behind which elements were emphasized was not always clear, making it difficult to assess the reasoning behind specific summary components. Additionally, both experts highlighted the lack of numerical values in the generated outputs as a limitation, stating that the inclusion of such figures would help contextualize changes and improve evaluative precision.

Participant 1 also reported instances of unclear or incorrect regional attributions, noting that such errors could result in misinterpretation when used in a financial reporting context.

While the feedback indicated certain limitations in accuracy and reliability, both participants noted that the model-generated summaries could serve as a helpful starting point for drafting financial commentary, particularly if supplemented with manual adjustments.

5.4.3 Summary of the Evaluation Findings

The evaluation indicated challenges across five themes. GPT-4o was preferred for reporting purposes due to its concise and professional language, while the Local LLM solution shows some potential to be suitable for internal analysis. Summaries from both models would require manual refinement, often including redundant or irrelevant content. Clarity was generally intense, particularly in GPT-4o, but both models occasionally mixed trends or listed multiple "main drivers". Accuracy issues were observed, including the omission of key changes and incorrect regional attributions. Structural inconsistencies and the absence of numerical values affected prioritization. While both models facilitated faster draft creation, their use in formal reporting remained limited due to concerns about trust and accuracy.

Theme	Identified Challenges
Usability	Summaries require manual refinement due to redundancy and inclusion of less relevant content.
Language	Sounds professional, but occasional inconsistencies in phrasing and emphasis affect interpret ability.
Accuracy	Some omissions of important information and inconsistencies in the trends.
Trust and Reliability	Limited confidence in summaries due to occasional missed trends.
Workflow Integration	Models support early-stage drafting but are not yet dependable for finalized reporting without human validation.

Table 5.7: Summary of Key Challenges by Evaluation Theme

5.5 Refinement of Cycle 2: Prompt-Chained Approach

Following the evaluation of the solution developed in Cycle 2, several limitations were identified, and it was concluded that refinements would be necessary if the solution were to produce output suitable for a final report. These included the model's occasional failure to adhere to formatting constraints, difficulties in handling multiple concurrent instructions, and challenges in logical consistency within the summary.

To address these issues, a prompt-chaining strategy was introduced as a refinement of the Cycle 2 solution. Rather than using a single prompt to both interpret and summarize the financial data, the task was split into two distinct steps.

- **Step 1 – Analyst Prompt:** The model first acted as a financial data analyst, interpreting the data and outputting a structured breakdown of product-line level changes. The analyst's output included details on key regions, consistency across regions, and each product line's impact on the overall trend.
- **Step 2 – Report Writer Prompt:** The analysis was then passed as input to a second prompt, where the model adopted the role of a report writer. This step focused purely on writing the summary in the specified format. Based on the feedback from the evaluation, some instructions were added and made stricter to make outputs clearer and closer to the desired format used in reporting. For example, a product line in a single summary or a summary can not include both a driver that is a main detractor and a main growth driver at the same time.

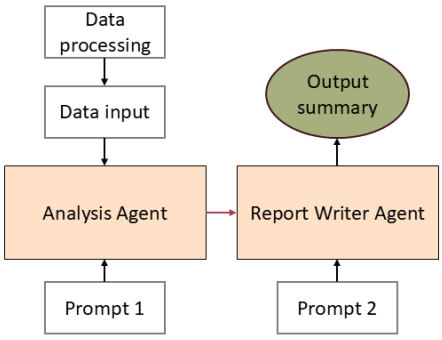

Figure 5.3: Two-agent system with analysis and summary generation.

This separation of concerns was intended to reduce the cognitive load placed on the model in a single prompt and to improve adherence to formatting rules and logical constraints. The prompts used for this can be found in the Appendix A.6.

5.5.1 Observations

The prompt-chained approach led to improvements in how the model interpreted financial data and structured its analysis. The analytical output produced in the first stage generally corresponded well with the underlying financial trends, indicating that the model was able to extract and organize relevant information at the product-line and regional level with a high degree of alignment to the data.

However, several limitations still persisted in the second stage, where the model generated the final report-ready summaries. Although improved, one limitation was the continued difficulty in prompting the model to consistently follow highly specific instructions, particularly those resembling conditional logic (e.g., "if X, then Y"). This issue became especially apparent when multiple constraints needed to be satisfied simultaneously. For example, summaries occasionally mentioned a main growth driver even when the overall direction of change was negative, despite the prompt explicitly instructing the model to avoid such statements under these conditions. Similarly, some summaries included repeated references to the same product line, contrary to the instruction that each product line must be mentioned only once.

In addition, the commentary generated in the second stage did not always align with the preceding structured analysis. Although the analysis step correctly identified key product-line and regional trends, these elements were sometimes omitted, rephrased inaccurately, or contradicted in the final summary.

These findings suggest that while dividing the summarization task into sequential roles reduces some of the cognitive load placed on the model, the challenge of translating structured intermediate output into precise, constraint-adherent final summaries remains. This highlights the difficulty for LLMs in reliably applying logic and formatting rules in text generation tasks within financial reporting, when multiple nuanced instructions must be satisfied concurrently.

6

Discussion

6.1 Challenges When Designing LLM Solutions in Financial Workflows

This section addresses RQ1 by identifying and discussing the primary software engineering challenges encountered when designing and integrating LLMs into financial report analysis workflows. Based on our iterative development and evaluation process, three key challenges emerged: 1. task-specificity in prompt and system design, 2. variability and complexity in data preparation, and 3. constraints arising from data sensitivity.

6.1.1 Task Specificity in Financial LLM Applications

One challenge emerging from the study is the high degree of specificity in financial reporting use cases and the impact it has on the design of LLM-based systems. For example, in the scenarios examined in this study, the summaries intended for inclusion in reports should follow a particular format. These summaries must be concise, precise, and unambiguous to align with domain expectations and support accurate interpretation.

Even with clear and explicitly worded prompts, the solutions struggle to follow the set rules for the output. How to get an LLM to produce a summary in a specific format posed a persistent challenge, particularly in cases where certain phrasing should be avoided if specific conditions are met. While the language used generally aligns with the tone and wording typical of financial reporting, the summaries frequently became ambiguous to readers with domain knowledge.

The introduction of a prompt-chaining strategy, both in the case when using local models and in the last iteration with GPT-4o, improved the overall structure and alignment with the underlying data for the analytical part of the task. However, errors still appeared in the final summarization stage, where the output sometimes failed to reflect key points identified in the structured analysis. This suggests that it is difficult for LLMs to handle many specific instructions simultaneously. While structured prompting strategies, such as CoT, show improvements in model reasoning for financial prediction tasks [32], these techniques do not fully apply to sum-

marization tasks that impose strict constraints on phrasing, tone, and structure. In these contexts, the challenge lies not only in guiding the model toward relevant content, but in designing prompts that consistently enforce detailed output, where small deviations can undermine clarity or correctness of the outputs.

This reflects a broader system-level challenge that has also been identified in previous research. Mahr et al. [40] frame prompt engineering as only one component within a larger architectural approach, emphasizing the importance of complementary mechanisms such as evaluation loops, feedback cycles, and verification steps to ensure reliable behavior in constrained and precision-critical applications. Our findings support this view, particularly in the context of end-to-end automated financial reporting workflows, where strict adherence to formatting and logical rules is required. While further decomposing the task and introducing validation mechanisms may help, designing prompts that enforce this level of control remains challenging. In more flexible use cases, such as internal analysis or drafting tasks, these issues are less critical. In such contexts, the model's output can still contribute value to the workflow despite occasional inconsistencies.

6.1.2 Data Preparation and Variability

Beyond the challenges of output formatting and control, another obstacle lies in the data formats used within the workflows. A challenge when designing LLM-based systems for financial reporting is the complexity of preparing and processing data that is used during inference. Financial reporting often requires accessing data from multiple systems and in varied formats. Depending on the use case, relevant information can be spread across spreadsheets, presentation decks, ERP systems, databases, or transmitted verbally during meetings. This diversity in data sources and formats presents an obstacle to developing generalizable and robust LLM workflows.

In this study, the input data used in the analysis were provided in a structured format and were preprocessed into a standardized format suitable for model inference. A custom script was developed to retrieve the relevant information and perform the necessary calculations. While effective for the specific scenario explored in this study, the solution was tightly coupled to a fixed input structure and lacked generalization across other similar tasks. This highlights one challenge when trying to generalize this approach to other similar tasks.

One potential approach to increasing system flexibility is the use of a tool that uses agents. LLMs can be integrated with external tools for data extraction and transformation. Rather than relying on fixed prompts, such agents can dynamically generate and execute scripts to retrieve relevant information from structured sources such as Excel files or databases. Frameworks such as ReAct show how chain-of-thought reasoning can be interleaved with tool calls to realize this behavior [84]. Orchestration patterns proposed by Anthropic demonstrate how tasks can be decomposed and delegated to specialized worker agents, yielding a more generalizable pre-processing pipeline that adapts to diverse data sources and formats [6]. In addition, many of

the most advanced models, for example, OpenAI's o3 series, embed function calling and code execution capabilities directly in the model, further reducing dependence on static prompts [52]. Together, these advances could enable a system to adjust its pre-processing strategy based on the task context dynamically.

In cases where inputs come in varied formats, models with multimodal capabilities may help address some challenges. For example, the model used in this study, GPT-4o is designed to interpret textual, verbal, and visual information in a single context window [29]. Other multimodal models, such as Claude 3.7 Sonnet [8] and Gemini 2.5 [23], could also process other formats like scanned documents or audio transcripts without explicit manual conversion. Consequently, these models could potentially handle verbal communication from meetings or presentation decks, thereby further increasing the system's flexibility.

6.1.3 Constraints from Data Sensitivity

A further consideration in the design of LLM-based systems for financial reporting is the sensitivity of the underlying data. Financial reports often contain confidential and proprietary information, which imposes specific requirements on data handling and model deployment. While such concerns can be addressed through vendor agreements or private cloud configurations, these arrangements can limit flexibility in choosing models or services. This implies that organizations are often restricted to using the models made available by a specific provider under the agreed terms. This dependency might prevent the adoption of more suitable or more capable alternatives. This is also a theme in the literature, where Tavasoli et al. [68] discuss the trade-offs between vendor-managed and internal deployments, highlighting tensions between data control, compliance, and operational efficiency, as well as the risk of vendor lock-in. Similarly, Lu et al. [38] emphasize how deployment decisions, particularly between external and sovereign models, shape the architectural options in sensitive and regulated domains. As a result, data sensitivity not only affects infrastructure decisions but also has a direct impact on the design space for LLM-enabled solutions in regulated domains such as finance.

6.1.4 Summary of RQ1

This study identified three key themes for designing LLM-based systems for financial reporting: strict task-specific output requirements, variability in input data formats, and data sensitivity constraints. These challenges might constrain the development of generalizable solutions, as prompt design is often tightly coupled to specific formatting and logic, input data must be carefully structured, and deployment is constrained by privacy concerns. While tool-using agents and multimodal models offer added flexibility, these factors also present meaningful engineering challenges that require thoughtful design and integration. This underscores the value of carefully assessing where and how LLM-based solutions can be most effectively implemented.

These insights align with prior research highlighting the importance of develop-

ing both architectural and organizational strategies that address such constraints. Tavasoli et al. [68] emphasize the need to evaluate whether LLMs are the most suitable solution compared to simpler alternatives, while also considering the return on investment from both operational and strategic perspectives. Their framework encourages weighing the benefits of automation and accuracy against the costs of infrastructure, compliance, and oversight. Similarly, Mahr et al. [40] argue that reliable LLM implementation depends not only on model capabilities but also on the broader system and institutional context in which the technology is deployed.

6.2 Analysis of Implementation Approaches

Following the discussion of the main engineering challenges identified in regarding RQ1, attention is now turned to RQ2, which concerns the comparison of different implementation approaches. Two solutions were developed and evaluated, a locally deployed one and one using GPT-4o. Expert feedback was used to assess output quality, reliability, and feasibility for practical use. The aim is to analyze the trade-offs observed and their implications for LLM integration in financial reporting contexts.

In contrast to evaluation-focused studies such as Yang et al. [83], which assessed several large open-source LLMs on financial news and trading report summarization using benchmark datasets and automatic quality metrics, the present work adopts a more holistic, system-oriented perspective grounded in a real-world use case. Rather than comparing multiple models in isolation, this study explores the different trade-offs involved in how two specific implementation options behave when embedded into domain-specific workflows for generating structured commentary in financial reporting.

6.2.1 Implementation with Local LLM Models

The main motivation for exploring local, open-source LLMs was their potential to meet key requirements within the financial domain. Running models locally offers full control over data, ensures privacy, and reduces dependency on external services. Smaller models also have lower computational demands, making them suitable in environments where dedicated infrastructure is limited. For simpler tasks, this appeared to be a promising direction. This aligns with Tavasoli et al. [68], who emphasize the importance of deployment strategies that consider data governance, infrastructure constraints, and regulatory requirements in financial settings.

During experimentation, several limitations with small-scale (3B parameter) models became apparent. These models frequently struggled with instruction following, hallucinated content, and occasionally returned code instead of natural language, even when explicitly instructed otherwise. Such problems suggest that using such models for specific tasks within financial reporting would imply that they come with many issues related to system stability. Therefore, experimentation continued with LLaMA 3.1 (8B parameter)

To address these limitations and better structure the reporting workflow, a multi-agent setup was implemented. This design improved output coherence, reduced hallucinations, and provided a more modular and manageable interaction flow. Yet, it introduced new challenges, early errors in the pipeline were sometimes propagated downstream, compounding deviations and making it more difficult to ensure reliable results. These challenges reflect broader limitations when designing multi-agent LLM systems. According to Pan et al. [53], such failures are often rooted in systemic design flaws, such as unclear role specification, misaligned coordination, or insufficient verification, rather than model capacity alone.

While the system was able to produce summaries that often aligned with underlying data, backed by the evaluation that it could support internal drafting tasks, two key issues persisted. First, it was difficult to prompt the model consistently to use the tone and phrasing expected in formal financial reporting. Evaluation results indicated that although the local solution could potentially support internal use cases, the generated language lacked the stylistic consistency needed for an end-to-end automation of the workflow. Second, the overall reliability of the system remained limited, as some summaries failed to capture key trends from the input data. These observations point to the need for further refinement before such setups can meet the standards of professional reporting.

To explore mechanisms for improving reliability, a basic self-evaluation strategy was tested, in which the model was prompted to assess its own summaries. This approach was intended as a step toward a feedback-driven pipeline using evaluator/optimizer patterns [6], in which generated outputs are iteratively reviewed and revised for accuracy and conformance to formatting standards. Although not fully implemented, this strategy represents a direction for improving stability in future iterations.

Further development was limited by insufficient access to dedicated computational resources. Although this restriction was specific to the research setting, it mirrors the conditions common in finance departments, where employees typically use standard laptops without access to high-performance infrastructure. In such environments, similar performance limitations can be expected when LLMs are deployed locally. This reflects a broader trade-off highlighted by Tavasoli et al. [68], who note that while open-source and internal deployment strategies offer strong advantages in terms of privacy, transparency, and control, they also demand substantial internal technical capacity. Nonetheless, where infrastructure is available, the local deployment of LLMs remains a feasible and potentially valuable option for internal use cases. The findings suggest that while these models may not be suitable for producing complete, report-ready outputs, they hold potential for supporting internal documentation, analysis assistance, or drafting tasks where stylistic precision is less critical.

6.2.2 Implementation with GPT-4o

The use of GPT-4o in Cycle 2 introduced a high-capacity model into the summarization workflow, requiring minimal setup for experimentation and offering fast

inference through third-party cloud services. The model demonstrated the ability to identify trends without extensive task decomposition, suggesting that powerful general-purpose LLMs can capture domain-relevant patterns even with relatively abstract task definitions. However, these capabilities came with notable trade-offs related to output quality, instruction adherence, and deployment feasibility.

Based on expert feedback, GPT-4o retained business-oriented linguistic patterns and generated summaries that were similar to those written by professionals. However, the model suffered from some structural issues in its outputs, as it had a tendency to occasionally include multiple "main drivers" in a single summary or accidentally mix negative and positive trends, resulting in incorrect or incoherent summaries. One insight from this ties into the role of prompt design and how to instruct the model. When example summaries were provided to guide structure and tone, the model tended to imitate the examples too rigidly, at times reproducing the phrasing given in the examples from the prompt, even when it conflicted with the actual input data. In contrast, when examples were omitted from the prompt, the model responded more accurately to the data, however, it did not produce outputs in the desired format for reports.

To address this, a prompt chaining strategy was attempted. This division, where one step focused on analysis and the next on generating the summary, led to improvements in how the model extracted and interpreted data. However, the second task of creating suitable summary documents for reporting still remained a challenge. Despite task decomposition and stricter instructions, the final summaries were often misaligned with the previous analysis, especially when prompts included conditional logic. These shortcomings support the findings of Lu et al. [38]. In line with their observations, this case also illustrates that in-context prompting alone is often insufficient for tasks requiring high formatting fidelity and suggests that further mechanisms are needed.

While the prompt design guidelines from Anthropic [5] supported by literature, [59] were applied in this study to help align LLM outputs with desired formats and communication styles, the results revealed a limitation. The use of format-oriented examples appeared to constrain GPT-4o's flexibility, leading it sometimes to prioritize stylistic consistency over data fidelity. This behavior aligns with Anthropic's caution that example outputs can overly constrain the model's behavior if not paired with clear instructions about when to adapt. In this study, the examples helped enforce structure but were not always paired with clear guidance on when to adapt. As a result, the model often prioritized format over content, reflecting the broader trade-off between consistency and contextual flexibility in prompt design.

Finally, the practical deployment of GPT-4o introduces an additional consideration. On-site deployment was not feasible in the case of this study, which required reliance on third-party APIs. While this enabled fast inference and simplified setup, it also raised concerns regarding data privacy and vendor dependency, issues also emphasized in prior work as central to deployment decisions in regulated domains [68, 38, 40].

6.2.3 Comparisons of the Approaches

Having analyzed each implementation individually, this section now compares their performance and reflects on their implications for system design, maintainability, and practical deployment in financial reporting contexts. As discussed in Section 5.4, expert feedback revealed important differences between the local LLM and GPT-4o in terms of output quality, clarity, and structure. From a software engineering perspective, these differences have direct implications for performance, implementation complexity, and maintainability, core aspects examined in RQ2, which explores how different implementation approaches compare in software engineering terms.

The locally deployed LLM tended to produce verbose, descriptive outputs. While this style offered potential value for internal documentation or exploratory analysis, it often lacked the conciseness required for formal reporting. In relation to RQ2, this variability in output significantly affects software complexity, filtering irrelevant content, enforcing consistent structure, and correcting phrasing require additional post-processing logic. These requirements increase the size and intricacy of the codebase, introduce more failure points, and reduce long-term maintainability. Therefore, while the local model offers benefits such as data privacy and deployment flexibility, its practical use imposes a heavier software engineering burden to achieve consistent and reliable behavior.

In contrast, GPT-4o produced summaries that more consistently matched the tone and structure expected in financial reports. This reduced the need for extensive formatting or content filtering, which in the context of RQ2 lowers the initial development effort and simplifies system maintenance. However, the model occasionally introduced logical inconsistencies, such as merging contradictory trends or failing to follow conditional phrasing rules. These issues, though less frequent compared to the local solution, still require engineering interventions such as verification modules or conditional correction logic. Thus, GPT-4o reduces routine engineering overhead but still necessitates additional safeguards to ensure correctness and trustworthiness.

A shared challenge across both approaches, relevant to RQ2, is the lack of interpretability. Neither model provides traceable reasoning for its outputs, making it difficult to debug inconsistencies or validate results. This limits the ability to build transparent, testable, and auditable systems. From an engineering perspective, this black-box behavior complicates error handling, testing, and quality assurance, all of which are critical for production-ready systems.

In summary, the findings suggest that while both models are capable of producing draft-quality summaries, they currently fall just short of delivering report-ready material suitable for external financial reports. This limitation highlights the need for further refinement before reliable automation can be achieved. The implementation approaches differ in how their outputs impact software engineering concerns highlighted in RQ2. The local model demands greater engineering investment to control and refine its behavior, whereas GPT-4o offers a more usable foundation with residual logic and reliability issues. These trade-offs underscore the need to

evaluate LLM solutions not only based on output quality but also on their broader implications for code maintainability, complexity, and reliability in financial software systems.

6.3 Implications for Future Work

Although LLM capabilities have been extensively explored in recent research, the software engineering perspective on how to build, integrate, and maintain such systems remains underdeveloped. Existing literature offers limited guidance on engineering practices and tool support for developing reliable, maintainable, and auditable LLM-based applications in complex, domain-specific environments.

This thesis has investigated the challenges of integrating LLMs into financial reporting workflows, as well as two integration approaches. Through this, several key software engineering challenges have emerged, including engineering for task-specific tasks, coordination within multi-agent systems, and trade-offs between deployment options. These findings highlight the need for more structured methods, evaluation practices, and architectural support for implementing LLMs in real-world contexts like finance

Although approaches like Chain of Thought (CoT) prompting have shown effectiveness in financial reasoning tasks, improving prediction accuracy [32], and organizations like Anthropic [5] have outlined best practices for prompt design, the findings suggest that these techniques may be insufficient for tasks with highly specific output requirements. In structured financial reporting, prompts must not only guide the model toward the correct content but also enforce precise phrasing, tone, and formatting, often under conditional rules. This level of specificity is a challenge.

Future work could explore modular, rule-aware prompt templates, methods for integrating conditional logic, and strategies for maintaining consistent domain-specific language across varied input. These efforts could include reusable prompting patterns, versioning strategy, and validation steps to improve system reliability and maintainability. As Weber et al. [74] argue, prompts should be treated as software assets. Building on this perspective, future research could investigate tooling for prompt versioning, documentation, and performance tracking, alongside test suites and benchmarking practices that enable more structured and reproducible development of prompts.

The exploration of multi-agent setups showed that architectural modularity improves task decomposition and clarifies task boundaries, but also introduces fragility when early-stage components produce faulty outputs. Errors tend to propagate across chained agents, particularly when verification steps are weak or absent. Future work could therefore focus on formalizing coordination protocols, enhancing fault isolation, and effectively designing evaluation loops that enable agents to provide feedback and correction to one another. These priorities align with the architectural taxonomy proposed by Lu et al. [38], which identifies prompt chaining, verifier

modules, and coordination layers as critical for ensuring robustness in LLM workflows. Building on these insights, future studies could empirically examine how such components impact maintainability, testability, and long-term reliability, especially in domain-specific environments like financial reporting, where interpretability and output fidelity are essential.

The comparison between local and commercial cloud-based models highlighted trade-offs between control, cost, and ease of integration. Local models provide full control over data and deployment environments, which is crucial for meeting strict privacy or compliance requirements. However, they demand significant engineering effort and infrastructure. In contrast, cloud-based models like GPT-4o offer ease of use and strong out-of-the-box performance but raise concerns about data sensitivity, compliance risks, and vendor lock-in. Future work could explore hybrid deployment strategies that combine local processing for sensitive logic or easier tasks with external models for more complex tasks. While existing frameworks, such as those proposed by Lu et al. [38] and Tavasoli et al. [68], offer guidance on architectural design and implementation considerations, it would be beneficial to explore how to design, manage, and evaluate hybrid LLM deployments in practice.

6.4 Limitations and Threats to Validity

The research follows the Design Science Research process, progressing through iterative cycles of problem identification, artifact design, evaluation, and reflection. Several potential threats to validity and reliability remain.

A limitation affecting internal validity is the limited number of domain experts involved in the evaluation. This small sample size was primarily due to limited access to individuals with the necessary expertise. As a result, only qualitative expert assessments were feasible, which may introduce subjectivity and reduce the consistency of evaluation outcomes. While the insights provided were valuable for assessing the solutions relevance and usability, the limited pool of evaluators constrains the robustness of the conclusions drawn.

Regarding construct validity, the evaluation relies on qualitative expert assessments rather than quantitative metrics, due to the structured and domain-specific nature of the generated outputs. Common metrics such as BLEU or ROUGE are not well-suited to capture correctness, regulatory compliance, or business relevance in financial reporting. By using expert judgment aligned with domain expectations, the research aimed to more accurately assess the artifacts real-world utility. However, this approach introduces a degree of subjectivity and limits comparability with other systems.

A practical limitation affecting external validity is the use of transformed and old financial data, introduced due to confidentiality constraints. Although the data was modeled to reflect real-world characteristics, actual organizational variability and data noise may influence performance in practice. Furthermore, the artifact was

designed and evaluated within the context of a specific financial reporting workflow at a single company. This narrow context limits the extent to which the findings can be generalized to other industries, organizational structures, or use cases.

Finally, reliability may be impacted by factors such as model non-determinism and infrastructure differences, as the system involves prompt engineering and multi-agent orchestration of LLMs.

7

Conclusion

This thesis examines the integration of LLMs into financial report analysis workflows, with a primary focus on the software engineering challenges that arise when designing these solutions. Through a Design Science approach, both local and cloud-based LLM systems were designed and evaluated for automating summarization tasks in a financial context. The findings respond to the two central research questions:

RQ1: *What are the primary software engineering challenges in integrating LLMs into financial report analysis workflows?*

RQ2: *How do different implementation approaches compare in terms of software engineering complexity, performance, and maintainability?*

While the study has shown that LLMs hold potential in automating financial reporting tasks, it also finds that they present some engineering challenges. Key issues include managing high task specificity, enforcing strict formatting rules, processing heterogeneous and sensitive data inputs, and ensuring traceability and explainability of outputs. These challenges are especially pronounced in precision-critical domains like financial reporting, where inconsistencies or ambiguity can compromise the usefulness and reliability of the output. These findings further highlight the importance of carefully evaluating LLMs by balancing the benefits of automation with the engineering effort and implementation costs introduced in the context.

Regarding RQ2, the comparison of implementation approaches revealed some trade-offs. The locally deployed solution offered stronger data privacy and greater architectural control, but this comes at the cost of increased engineering complexity and a need for greater technical expertise within an organization. Developing such systems requires more extensive post-processing logic, task decomposition, and robust validation mechanisms. Even with these efforts, the solutions are unlikely to meet formal financial reporting standards in their current state without substantial customization and oversight. However, they still hold potential for internal tasks where rigid formatting is less of a concern. In contrast, GPT-4o demonstrated better linguistic quality and required less manual effort to support reporting tasks, reducing the need for extensive task decomposition. The model was able to capture domain-relevant language and identify trends with less prompt engineering, showing potential for

generating summaries similar to those written by professionals. However, at times, it struggles with consistent instruction adherence and maintaining logical coherence when multiple constraints are involved. While prompt chaining improved intermediate results, the final report often lacked alignment with the intended structure and content. Furthermore, the reliance on external infrastructure raises concerns around data sensitivity and vendor dependency.

Overall, while full end-to-end automation of formal financial reporting remains a challenge, promising progress has been demonstrated in using LLMs to support and partially automate such workflows. With further refinements in task adherence, validation mechanisms, and system coordination, full automation would likely be feasible.

Bibliography

[1] Emmanuel A. Abbe, Amir E. Khandani, and Andrew W. Lo. Privacy-preserving methods for sharing financial risk exposures. *The American Economic Review*, 102(3):65–70, 2012.

[2] R. Ahmed, S. A. Rauf, and S. Latif. Leveraging large language models and prompt settings for context-aware financial sentiment analysis. In *Proceedings of the 2024 5th International Conference on Advancements in Computational Sciences (ICACS)*, pages 1–9. IEEE, February 2024.

[3] Serap Akcan Yetgin and Hilal Altas. Analyzing the corporate business intelligence impact: A case study in the financial sector. *Applied Sciences*, 15:1012, 01 2025.

[4] Humaid Al Naqbi, Zied Bahroun, and Vian Ahmed. Enhancing work productivity through generative artificial intelligence: A comprehensive literature review. *Sustainability*, 16(3), 2024.

[5] Anthropic. Be clear, direct, and detailed [internet], 2024. San Francisco, CA: Anthropic; 2024 [cited 2025 May 8]. Available from: https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/be-clear-and-direct.

[6] Anthropic. Building effective agents [internet], 2024. San Francisco, CA: Anthropic; 2024 [cited 2025 Feb 19]. Available from: https://www.anthropic.com/research/building-effective-agents.

[7] Anthropic. Claude 3.7 sonnet [internet], 2025. San Francisco, CA: Anthropic; 2025 [cited 2025 May 7]. Available from: https://www.anthropic.com/claude/sonnet.

[8] Anthropic. Claude 3.7 sonnet system card [internet]. Technical report, Anthropic PBC, 2025. San Francisco, CA: Anthropic PBC; 2025 Feb [cited 2025 May 7]. Available from: https://www.anthropic.com/news/claude-3-7-sonnet.

[9] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models, 2019.

[10] John DuChateau Baierl. Applications of large language models in education: Literature review and case study. Master's thesis, University of California, Los Angeles, 2023. Available from: https://escholarship.org/uc/item/6kf0r28s.

[11] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623. Association for Computing Machinery, March 2021.

[12] J. Berk, P. DeMarzo, and J. Harford. *Fundamentals of Corporate Finance*. Pearson Higher Education AU, 6th edition, 2023.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang. A survey on a mixture of experts in large language models. *Authorea Preprints*, 2024.

[15] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021.

[16] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

[17] Wang Chen, Liu Yan-yi, Guo Tie-zheng, Li Da-peng, He Tao, Li Zhi, Yang Qing-wen, Wang Hui-han, and Wen Ying-you. Systems engineering issues for industry applications of large language model. *Applied Soft Computing*, 151:111165, 2024.

[18] J. Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*, arXiv:1810.04805, 2018.

[19] Jacob Eisenstein. *Introduction to Natural Language Processing*. MIT Press, 2019.

[20] R. Feldt, F. G. de Oliveira Neto, and R. Torkar. Ways of applying artificial intelligence in software engineering. In *Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, pages 35–41, May 2018.

[21] G. Feretzakis and V. S. Verykios. Trustworthy ai: Securing sensitive data in

large language models. *arXiv preprint arXiv:2409.18222*, September 2024.

[22] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[23] Google Cloud. Gemini 2.5 pro [internet], 2025. Mountain View, CA: Google Cloud; 2025 [cited 2025 May 7]. Available from: https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro.

[24] Arash Hajikhani and Carolyn Cole. A critical review of large language models: Sensitivity, bias, and the path toward specialized ai. *Quantitative Science Studies*, 5(3):736–756, August 2024.

[25] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, pages 75–105, Mar 2004.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] A. H. Huang, H. Wang, and Y. Yang. Finbert: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 40(2):806–841, May 2023.

[28] Hugging Face. Smolagents documentation [internet], 2024. Place of publication unknown: Hugging Face; 2024 [cited 2024 Feb 27]. Available from: https://huggingface.co/docs/smolagents/en/index.

[29] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[30] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. Manuscript, Stanford University, 3rd edition, 2025. Online manuscript released January 12, 2025.

[31] S. Keele. Guidelines for performing systematic literature reviews in software engineering. Technical report, EBSE Technical Report, 2007. Version 2.3.

[32] A. Kim, M. Muhn, and V. Nikolaev. Financial statement analysis with large language models, 2024. arXiv preprint.

[33] Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv:2308.07702*, 2023.

[34] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model.

Computer Speech Language, 6(3):225–242, 1992.

[35] Jean Lee, Nicholas Stevens, and Soyeon Caren Han. Large language models in finance (finllms). *Neural Computing and Applications*, pages 1–15, 2025.

[36] Craig Lewis and Steven Young. Fad or future? automated analysis of financial text and its implications for corporate reporting. *Accounting and Business Research*, 49, 2019.

[37] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[38] Q. Lu, L. Zhu, X. Xu, Y. Liu, Z. Xing, and J. Whittle. A taxonomy of foundation model based systems through the lens of software architecture. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 1–6, Apr 2024.

[39] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

[40] F. Mahr, G. Angeli, T. Sindel, K. Schmidt, and J. Franke. A reference architecture for deploying large language model applications in industrial environments. In *2024 IEEE 30th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 19–23. IEEE, Oct 2024.

[41] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[42] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner. Software engineering for ai-based systems: A survey. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(2):1–59, 2022.

[43] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, 2014.

[44] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI, 1998.

[45] Xiangbin Meng, Xiangyu Yan, Kuo Zhang, Da Liu, Xiaojuan Cui, Yaodong Yang, Muhan Zhang, Chunxia Cao, Jingjia Wang, Xuliang Wang, Jun Gao, Yuan-Geng-Shuo Wang, Jia-ming Ji, Zifeng Qiu, Muzi Li, Cheng Qian, Tianze Guo, Shuangquan Ma, Zeying Wang, Zexuan Guo, Youlan Lei, Chunli Shao, Wenyao Wang, Haojun Fan, and Yi-Da Tang. The application of large language

models in medicine: A scoping review. *iScience*, 27(5):109713, 2024.

[46] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[47] Jacob Murel and Joshua Noble. Encoder-decoder model. IBM [Internet], 2024. [cited 2025-03-18].

[48] NVIDIA. What are large language models (llms)? [internet], 2024. Santa Clara, CA: NVIDIA; 2024 [cited 2025 Apr 15]. Available from: https://www.nvidia.com/en-us/glossary/large-language-models/.

[49] Ollama. Ollama: Run llms locally [internet], 2024. [cited 2025 Mar 14]. Available from: https://github.com/jmorganca/ollama.

[50] OpenAI. Gpt-4 technical report. *arXiv*, abs/2303.08774, 2023. DOI: 10.48550/arXiv.2303.08774.

[51] OpenAI. Introducing gpt-4.5 [internet], 2025. San Francisco: OpenAI; 2025 [cited 2025 May 5]. Available from: https://openai.com/index/introducing-gpt-4-5/.

[52] OpenAI. Openai o3 and o4-mini system card [internet]. Technical report, OpenAI, 2025. San Francisco, CA: OpenAI; 2025 Apr [cited 2025 May 7]. Available from: https://openai.com/index/o3-o4-mini-system-card/.

[53] Melissa Z Pan, Mert Cemri, Lakshya A Agrawal, Shuyi Yang, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Kannan Ramchandran, Dan Klein, et al. Why do multiagent systems fail? In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025.

[54] C. Parnin, G. Soares, R. Pandita, S. Gulwani, J. Rich, and A. Z. Henley. Building your own product copilot: Challenges, opportunities, and needs. *arXiv e-prints*, arXiv:2312, Dec 2023.

[55] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[56] Baolin Peng, Chien-Sheng Wu, Andrei Chronopoulos, Xiangru Tang, Hady Elsahar, Jiun-Yu Kao, Lars Liden, Michel Galley, and Jianfeng Gao. AgentBench: Evaluating foundation models as agents. *arXiv preprint arXiv:2308.11458*, 2023.

[57] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[58] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[59] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, New York, NY, USA, 2021. Association for Computing Machinery.

[60] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, Chichester, UK, 1 edition, 2012.

[61] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed. Machine learning for software engineering: A systematic mapping. *arXiv preprint*, arXiv:2005.13299, May 2020.

[62] Sakib Shahriar, Brady D Lund, Nishith Reddy Mannuru, Muhammad Arbab Arshad, Kadhim Hayawi, Ravi Varma Kumar Bevara, Aashrith Mannuru, and Laiba Batool. Putting gpt-4o to the sword: A comprehensive evaluation of language, vision, speech, and multimodal proficiency. *Applied Sciences*, 14(17):7782, 2024.

[63] Sharma. Using llms in financial statement analysis: A deep dive [internet], 2025. [place unknown]: Medium; 2025 [updated 2025 Feb; cited 2025 Apr 15]. Available from: https://medium.com/arya-ai-tech-blog/using-llms-in-financial-statement-analysis-a-deep-dive-22aba0d66eed.

[64] Stanford University. Cs102: Spreadsheets [internet], 2020. Stanford (CA): Stanford University, Department of Computer Science; 2020 [cited 2025 May 15]. Available from: https://web.stanford.edu/class/cs102/lectureslides/SpreadsheetsSlides.pdf.

[65] K. J. Stol and B. Fitzgerald. The abc of software engineering research. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 27(3):1–51, 2018.

[66] Stryker, Cole and Holdsworth, Jim. What is natural language processing?, 2024. Armonk, NY: IBM; [2024] [cited 2025 Mar 11]. Available from: https://www.ibm.com/think/topics/natural-language-processing.

[67] Chee Fai Tan, L.S. Wahidin, Siti Nurhaida Khalil, Noreffendy Tamaldin, Jun Hu, and Matthias Rauterberg. The application of expert system: A review of research and applications. 11:2448–2453, 01 2016.

[68] Ahmadreza Tavasoli, Maedeh Sharbaf, and Seyed Mohamad Madani. Responsible innovation: A strategic framework for financial llm integration. *arXiv preprint arXiv:2504.02165*, 2025.

[69] O. Topsakal and T. C. Akinci. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056, Jul 2023.

[70] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, and A. Rodriguez. Llama: Open and efficient foundation language models. *arXiv preprint*, arXiv:2302.13971, feb 2023. Feb 27.

[71] S. Uchitel, M. Chechik, M. Di Penta, B. Adams, N. Aguirre, G. Bavota, D. Bianculli, K. Blincoe, A. Cavalcanti, Y. Dittrich, and F. Ferrucci. Scoping software engineering for ai: The tse perspective. *Institute of Electrical and Electronics Engineers*, 2024.

[72] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[73] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, and W. X. Zhao. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, Dec 2024.

[74] Irene Weber. Large language models as software components: A taxonomy for llm-integrated applications, 2024.

[75] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[76] Joseph Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[77] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, and A. H. Awadallah. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint*, arXiv:2308.08155, Aug 2023. Accessed: 2024-02-27.

[78] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, G. Mann, et al. Bloomberggpt: A large language model for finance. *arXiv preprint*, arXiv:2303.17564, 2023.

[79] Shijie Wu, Ozan Irsoy, Steven Lu, and Vadim Dabravolski. Bloomberggpt: A large language model for finance, 2023. arXiv preprint.

[80] T. Wu, M. Terry, and C. J. Cai. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–22, Apr 2022.

[81] Q. Xie, W. Han, X. Zhang, Y. Lai, M. Peng, A. Lopez-Lira, and J. Huang. Pixiu: A large language model, instruction data and evaluation benchmark for finance. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 33469–33484, Dec 2023.

[82] H. Yang, X. Y. Liu, and C. D. Wang. Fingpt: Open-source financial large language models. *arXiv preprint*, arXiv:2306.06031, Jun 2023.

[83] X. Yang, S. Zang, Y. Ren, D. Peng, and Z. Wen. Evaluating large language models on financial report summarization: An empirical study. *arXiv preprint*, arXiv:2411.06852, 2024.

[84] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. arXiv preprint arXiv:2210.03629.

[85] Y. Yu and P. Tiwari. Finteamexperts: Role specialized moes for financial analysis, 2024. arXiv preprint.

[86] B. Zhang, H. Yang, and X. Y. Liu. Instruct-fingpt: Financial sentiment analysis by instruction tuning of general-purpose large language models, 2023. arXiv preprint.

[87] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, and G. Wang. Instruction tuning for large language models: A survey. *arXiv preprint*, arXiv:2308.10792, 2023.

[88] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A

Appendix

A.1 Prompts for local solution

Step 1: Row-Level Summarization

Role: You are a financial data analyst summarizing trends for the order intake trends for different product lines and regions

---

Context: The dataset contains order intake changes across various product lines and regions between two periods. Each row represents the change, including its magnitude and the relative impact on the overall change.

Table: `{table_summary}`

Task:

- Process each row individually and convert the structured data into a natural language summary.
- Clearly indicate whether the change is considered a major or minor impact.
- Structure the output concisely while ensuring that key details are retained.

Rules:

- Use concise, structured sentences.
- Do **not** include raw numbers, just directional trends.
- Do **not** generate code.

Example Output Format: `[Product Line] in [Region] experienced a [major/minor] [increase/decrease].`

Table A.1: Step 1: Row-Level Summarization

Step 2: Identifying Key Trends and Drivers

Role: You are a business analyst specializing in data-driven insights for net-sales trends

---

Context: The summarized data presents order intake changes across two periods. It categorizes changes per product line, in which regions they occurred, as well as their magnitude (minor/major increase or decrease).

Summary: `{generated_summary}`

Task:
1. Consider each product line individually.
2. Identify if the direction of change is consistent across regions.
3. Determine major drivers `[Product Line] + [Region]` for increases and decreases.
4. Summarize findings in natural language. Do not generate code)

Rules:
- Use clear, structured language.
- Do not include raw numbers, only directional trends.
- Do not generate code.

Example Output Format: `[Product Line] has consistently increased across all regions. [Product Line] saw the largest decline in [Region].`

Table A.2: Step 2: Identifying Key Trends and Drivers

Step 3: Final Executive Summary

Role: You are a report writer summarizing trends from order intake for a qualitative summary.

Context: The following analysis summarizes major and minor order intake changes across product lines and regions. The objective is to generate a concise executive summary for reporting purposes.

Analysis: `{generated_analysis}`,

Task:
Create a summary of the data
1.Write a **concise summary** of the key trends.
2.Highlight significant changes and ignore minor fluctuations.

Rules
1. Maximum 4 sentences
2. Make it descriptive and concise.
3. Do not generate code

Example Output Format: `[Product Line] in [Region] as the main growth driver. [Product Line] up in all regions. [Product Line] increasing, mainly in [Region]. Decrease from [Product Line] in [Region].`

Table A.3: Step 3: Final Executive Summary

Step 4: Sanity Check

Role: You are a financial validator that factchecks the financial reporting

---

Context: Below is a generated summary of order intake trends.
Your task is to validate if the summary describes the raw data correctly.

Summary: `{generated_summary}`, **Raw data**`{raw_data}`

Task:
1. Verify that trends match the raw data.
2. Flag any incorrect or misleading information.

Rules: Do not rewrite; just provide a validation report.

Example Output Format:

```
- Validation Passed:  No inconsistencies.
- Validation Warning:  [Product Line] in [Region] is reported as
an increase, but data shows a decrease.
- Validation Warning:  [Product Line] reported decrease across
all regions, but data show increases in [Region].
```

Table A.4: Step 4: Sanity Check

A.2 Prompts for Refinement with GPT-4o

Prompt to GPT-4o

Context:
You are a financial data analyst. Your task is to review segmented financial data and identify key patterns.

Instructions:
You must output your findings in a clear, structured, and concise format.

- Determine accumulated direction of change across all product lines (Positive/Negative). The accumulated change is provided.
- List the product lines in descending order of overall impact.
- For each product line, include:
  - Key Regions: Mention regions with notable increases or decreases, and numeric values if available.
  - Consistency: Indicate whether the direction of change is consistent across all regions.
  - Impact: Specify whether the product line is a main contributor, secondary impact, or has offsetting effects.

Rules:

- Be factual and direct.
- Avoid speculation.
- Use scientific notation for numbers if applicable.
- Use short, professional phrasing.

Output Format:

```
- Accumulated Direction for the Segment: [Positive/Negative]
- [Product Line Name]:
  - Key Regions: [Region] with [increase/decrease] of [value],
  followed by [Region]...
  - Consistency: [Consistent across regions / Inconsistent
  / Mixed].
  - Impact: [Main contributor / Offset by decline
  / Secondary impact].
```

Input:

```
Data:  {data_block}
```

Table A.5: Prompt for Analyzing Net Sales Patterns

Prompt to GPT-4o

Context:
You are a financial report writer writing a financial report. Your task is to analyze and summarize changes in financial data.

Instructions:
You must write a concise summary in at most four sentences to complement the numerical data in a financial report.
Focus on clear, direct language and only state what is evident in the data—no speculation or recommendations.
Do not include numbers. Focus on trends, main drivers, and regions as specified.

The data shows net sales changes segmented by product line and region, with relative contributions to the overall change in a business area.

- Identify whether the overall trend is an increase or decrease.
- Detect the main positive and main negative drivers.
- Detect if there are product lines where the direction of change is consistent across all regions.
- Write a concise final qualitative summary about the detected trends.
- Start by mentioning the trend that has the biggest impact overall.

Rules for the final summary:

- Maximum of four sentences.
- STRICT RULES: Do not mention a main growth driver if the accumulated direction of change is negative.
- STRICT RULES: Do not mention a main detractor if the accumulated direction of change is positive.
- Mention each product line no more than once.
- Do not speculate, do not include numbers.
- Do not include the overall trend in the summary since that is already understood.
- Use concise, factual phrasing as in the examples below.

Expected output examples:

- `All product lines up. [Product Line] in [Region] as main growth driver. [Product Line] increasing in [Region]. Increase from [Product Line] in [Region]. [Product Line] up in all regions.`
- `[Product Line] and [Product Line] decreasing in [Region]. [Product Line] in [Region] as main detractor, partly offset by [Product Line] in [Region].`
- `[Product Line] in [Region] as main detractor. Increase from [Product Line] in [Region].`
- `[Product Line] up in all regions. [Product Line] down in [Region]. [Product Line] up in [Region].`
- `[Product Line] increasing in all regions. Partly offset by [Product Line] in all regions as well as [Product Line] in [Region].`

Below is an analysis of financial data. Your task is to convert this into a summary, following the style and constraints defined. `{analysis_text}`

Table A.6: Prompt for Generating Net Sales Summary

A.3 Evaluation

Evaluation Themes and Guiding Questions
Comprehension and Usefulness
Which summary feels most helpful or closest to what you would actually want to send or present?
Which model and summary type do you prefer, and why?
Do any of the summaries miss something important, or include something unnecessary?
Do any of the summaries provide insights or phrasing you found useful?
Clarity and Language
How would you rate the clarity and professionalism of the language used in each summary?
Do they sound like something a business controller or financial analyst would write?
Accuracy
Do you feel the summaries explain the trends accurately?
Is there anything that they bring up that is redundant?
Workflow Integration
If one of these models could automatically generate drafts of your commentary, would that help you? How?
What part of your job could this realistically support?
What part would still require manual input?
What would make you not trust a summary generated by AI?
Practical Concerns
Would you feel comfortable using this after seeing this?
What features or improvements would make you more likely to actually use such a tool?
Final Reflections
What's your biggest concern and biggest opportunity with using AI in your financial reporting?
Do you have any other thoughts or reflections?

Table A.7: Evaluation themes and guiding questions used in the artifact assessment