

UHASSELT



Maastricht University

KNOWLEDGE IN ACTION

## Faculteit Wetenschappen School voor Informatietechnologie

master in de informatica

### Masterthesis

*Evaluating the efficiency of large language models in data integration tasks*

**Kiran Singh**

Scriptie ingediend tot het behalen van de graad van master in de informatica

### PROMOTOR :

Prof. dr. Frank NEVEN

### BEGELEIDER :

dr. Brecht VANDEVOORT

De transnationale Universiteit Limburg is een uniek samenwerkingsverband van twee universiteiten in twee landen: de Universiteit Hasselt en Maastricht University.



UHASSELT

KNOWLEDGE IN ACTION

[www.uhasselt.be](http://www.uhasselt.be)

Universiteit Hasselt  
Campus Hasselt:  
Martelarenlaan 42 | 3500 Hasselt  
Campus Diepenbeek:  
Agoralaan Gebouw D | 3590 Diepenbeek

2023  
2024



**Maastricht University**

# **Faculteit Wetenschappen**

## ***School voor Informatietechnologie***

master in de informatica

### ***Masterthesis***

***Evaluating the efficiency of large language models in data integration tasks***

**Kiran Singh**

Scriptie ingediend tot het behalen van de graad van master in de informatica

**PROMOTOR :**

Prof. dr. Frank NEVEN

**BEGELEIDER :**

dr. Brecht VANDEVOORT



# Acknowledgment

The master's thesis marks the finish of an incredible five-year journey at UHasselt. This thesis has set me in the right direction for my career in the data space. I have documented the entire journey through my final master's year thesis here. The successful completion of this work would not have been possible without the support of the following persons.

I would like to thank my promoter, prof. dr. Frank Neven, for his fantastic support throughout my master's thesis. With bi-weekly meetings, he always ensured that I was on the right track. His insights and advice were indispensable. By creating interim presentations, taking notes throughout the period, and frequently consulting with each other, we always managed to find a solution.

I would also like to thank my supervisor, dr. Brecht Vandevooort, for guiding me through this master's thesis. I could always turn to him with urgent questions, whether they were about implementation or analysis. He was always ready to help me with any difficulties.

Finally, I want to thank my parents, brother, and friends for all their support. I could share all my findings, setbacks, and insights with them, and they helped me further investigate and incorporate these into this thesis.

Kiran Pal Singh  
Hasselt, June 19, 2024

# Dutch summary

Het primaire doel van dit proefschrift is het evalueren van de effectiviteit en bruikbaarheid van large language models<sup>1</sup> (LLMs) in data-integratietaken. Data-integratie omvat het extraheren van data uit verschillende bronnen en het verwerken ervan voor gebruik in een integratieworkflow. Dit proefschrift richt zich op de data-extractiecomponent van data-integratietaken, waarbij LLMs worden ingezet om dit proces te verbeteren. LLMs zijn een type kunstmatige intelligentie waarbij neurale netwerken<sup>2</sup> worden getraind op enorme hoeveelheden data om antwoorden op vragen te genereren.

Om dit onderzoek te sturen, zijn er verschillende onderzoeksvragen geformuleerd. De eerste onderzoeksvraag probeert te bepalen of LLMs consistent tekst kunnen extraheren uit grote invoerdocumenten. De tweede vraag, gerelateerd aan de eerste, richt zich op het beoordelen van de nauwkeurigheid van LLMs bij het genereren van key-value pairs tijdens vraag-en-antwoordtaken. De “key” is de vraag en de “value” is het gegenereerde antwoord. Nauwkeurigheid wordt gemeten door het gegenereerde antwoord te vergelijken met de correcte respons.

LLMs kunnen worden gecategoriseerd in twee typen: open-source modellen, die openbaar beschikbaar zijn en lokaal gedownload en getest kunnen worden, en propriëtaire modellen, die alleen toegankelijk zijn via APIs. De derde onderzoeksvraag onderzoekt de prestaties en bruikbaarheid van zowel open-source als propriëtaire modellen, en of ze uitwisselbaar gebruikt kunnen worden.

Tenslotte, aangezien LLMs incorrecte antwoorden kunnen genereren, evalueert de vierde onderzoeksvraag of menselijke tussenkomst de bruikbaarheid van LLMs in data-integratietaken kan verbeteren.

## Opzet van experimenten

Om de onderzoeksvragen te beantwoorden, zijn twee casestudies geanalyseerd waarin zowel open-source als propriëtaire LLMs werden getest om hun prestaties en beperkingen te evalueren. Voor propriëtaire LLMs werden de GPT-3 en GPT-4 modellen van OpenAI<sup>3</sup> getest, terwijl voor open-source LLMs modellen van Meta AI<sup>4</sup> (Llama2 modellen) en Mistral AI<sup>5</sup> (Mistral en Mixtral modellen) werden onderzocht. Een end-to-end testframework genaamd de LLM-pipeline (zie section 3.3) werd ontwikkeld om deze experimenten te faciliteren. Deze pipeline is verantwoordelijk voor het testen en valideren van de resultaten voor zowel open-source als propriëtaire modellen.

De LLM-pipeline bestaat uit vijf onderdelen: 1) Data Extraction, 2) Validation Set Creation, 3) LLM Inferencing, 4) Post-Validation, and 5) Visualization and Analysis. The pipeline begint met ruwe invoergegevens, die ofwel PDF- of HTML-bestanden kunnen zijn, afhankelijk van de casestudie. De Data Extractor module extraheert en converteert deze invoergegevens naar

---

<sup>1</sup><https://www.ibm.com/topics/large-language-models>

<sup>2</sup><https://www.ibm.com/topics/neural-networks>

<sup>3</sup><https://platform.openai.com/docs/models>

<sup>4</sup><https://ai.meta.com/meta-ai/>

<sup>5</sup><https://mistral.ai/>

tekst die de LLM kan verwerken. Vanwege het beperkte contextvenster van LLMs, converteert de Data Extractor module ook de invoerdataset naar een relevant-only dataset, wat soms handmatige tussenkomst vereist op basis van de eisen van de casestudie.

Voordat de data naar de LLM wordt gestuurd, omvat een tussenstap het creëren van een validatieset. In deze stap wordt een goldenset opgebouwd uit de originele invoerdataset, die aangeeft welke key-value pairs aanwezig zijn in de PDFs. Als de invoer dataset een metadatabestand bevat, kan dit worden omgezet naar de goldenset; anders wordt de goldenset handmatig geconstrueerd.

De kernfunctionaliteit van de LLM-pipeline is de LLM Prompter, die de output van de Data Extractor module en de goldenset combineert om de LLM aan te sturen. De goldenset wordt gebruikt om de prompts bij te werken om vragen te beantwoorden gerelateerd aan de geselecteerde casestudie. Deze prompttemplate is iteratief ontworpen om de LLM te beperken van hallucineren en incorrecte antwoorden te geven. Hallucinaties treden op wanneer de LLM output genereert die niet aanwezig is in de invoercontext.

Voor propriëtaire modellen wordt de OpenAI API gebruikt om met de GPT modellen te communiceren. Om open-source modellen te testen, wordt de infrastructuur van het Vlaams Supercomputer Centrum (VSC) gebruikt. De volledige LLM-pipeline wordt naar het VSC gekloond voor te testen, omdat open-source modellen meer GPU-geheugen nodig hebben dan een standaard computer kan leveren. Het VSC stelt zijn infrastructuur beschikbaar voor onderzoeksdoeleinden en dit proefschrift benut die mogelijkheid.

De output van de LLM Prompter module wordt gebruikt in de Validation module, samen met de validatieset, voor het nabewerken van de ruwe tekstuele LLM-output naar JSON-formaat en het berekenen van de nauwkeurigheid van de LLM met behulp van de goldenset uit de validatieset. Het validatieproces telt correcte en incorrecte antwoorden en genereert een JSON-bestand dat kan worden gevisualiseerd met de Human-In-The-Loop (HITL) tool of verder geanalyseerd met de ontworpen python notebooks.

### **Human-In-The-Loop (HITL)**

Om de onderzoeksvraag met betrekking tot menselijke tussenkomst bij de experimenten te beantwoorden, hebben we een Human-In-The-Loop interface ontworpen. Deze interface helpt ontwikkelaars en eindgebruikers de output van het Language Learning Model (LLM) beter te begrijpen bij data-integratietaken. Het toont visueel de prestaties van de LLM bij data-extractie, waardoor gebruikers deze eenvoudiger kunnen evalueren.

De tool is gebouwd met Streamlit<sup>6</sup> en gebruikt de HTML PDF-renderer om de PDFs op de interface weer te geven. Het is een standalone tool en voert niet automatisch tests uit op VSC en maakt ook geen gebruik van de OpenAI API. Gebruikers moeten handmatig de LLM-pipeline uitvoeren en het resulterende JSON-bestand downloaden. Dit JSON-bestand kan dan samen met het gewenste PDF-document van de casestudy in de tool worden geüpload voor verdere analyse.

De interface heeft twee primaire pagina's. De eerste pagina stelt gebruikers in staat de output van de LLM te vergelijken met de goldenset om de nauwkeurigheid te beoordelen. De tweede pagina is bedoeld voor scenario's waarin de goldenset niet beschikbaar is; in dit geval creëert de tool een geaggregeerde set om gebruikers te helpen bij het opbouwen van de goldenset. Beide pagina's bieden een visuele weergave van de output om de bruikbaarheid van de invoerdataset te evalueren.

Gebruikers kunnen fouten en patronen in de LLM-output visualiseren en analyseren via een intuïtieve interface, wat een gebruiksvriendelijke verkenning van de prestaties van de LLM mogelijk maakt.

---

<sup>6</sup><https://streamlit.io/>

## Casestudies

Er werden twee casestudies uitgevoerd om de prestaties en beperkingen van Large Language Models (LLMs) te analyseren. De eerste casestudy betrof energieleveringscontracten als de invoerdataset. Deze dataset werd handmatig samengesteld door 8 energieleveringscontract PDF-documenten te downloaden van mijnenergie.be<sup>7</sup>, waarbij we 8 verschillende key-value pairs testten. De tweede casestudy richtte zich op de financiële sector, specifiek op SEC-indieningsdocumenten van Coca-Cola.<sup>8</sup> We downloadden 31 documenten die inzichten gaven in de financiële prestaties van het bedrijf, en testten 12 verschillende key-value pairs. In beide casestudies werden de LLMs getest op twee hoofdgebieden van de PDFs: vragen met antwoorden in gewone tekst en vragen met antwoorden in tabellen.

Een duidelijk patroon dat werd waargenomen was dat de geteste LLMs moeite hadden met het extraheren van data uit tabellen, waarbij ze de laagste nauwkeurigheid vertoonden op deze key-value pairs. Met behulp van de ‘‘HITL’’ visualisatietool merkten we dat de LLMs vaak de data verkeerd interpreteerden en antwoorden uit aangrenzende kolommen haalden. We specificerden opzettelijk niet welke kolom de juiste antwoorden bevatte om de leercapaciteit van de LLMs te testen. Topmodellen (GPT4-o en Mixtral 7x8B) leerden dit patroon succesvol, terwijl kleinere modellen (LLAMA 7B, 13B en Mistral 7B) dat niet deden. De gedetailleerde nauwkeurigheden zijn beschikbaar in Table 1, Table 2 en Table 3. De resultaten tonen aan dat propri etaire modellen de hoogste nauwkeurigheid en consistentie hebben onder alle geteste modellen, wat de eerste twee onderzoeksvragen beantwoordt.

M7B	L7B	L13B	L70B	GPT3.5 MD	GPT3.5
34	41	45	48	61	63

**Table 1:** Score of LLMs tested (Part 1), score ranging from 0 to 100, 100 meaning that the model answers every question correctly.

M8x7B MD	GPT4	M8x7B	GPT4-o
69	77	79	87

**Table 2:** Score of LLMs tested (Part 2), score ranging from 0 to 100, 100 meaning that the model answers every question correctly.

Mistral 7B	Mixtral 7x8B	GPT4-o
69,34	93,36	99,30

**Table 3:** Score of all the LLMs tested, score ranging from 0 to 100, 100 meaning that the model answer every question correctly

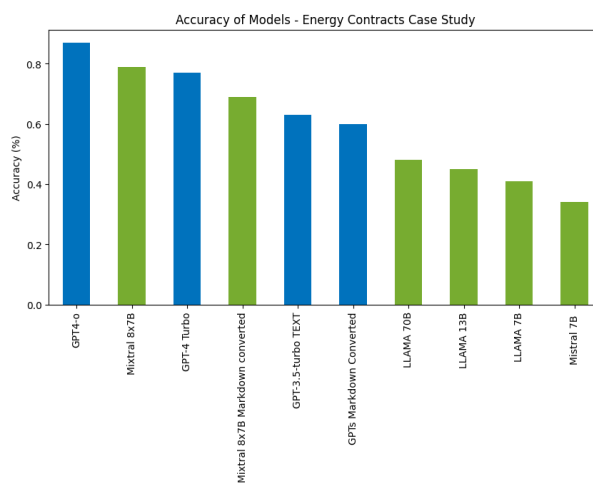
Om de derde onderzoeksvraag te beantwoorden, vergeleken we alle geteste modellen over de twee casestudies om te bepalen welke open-source modellen alternatieven kunnen zijn voor propri etaire modellen. Uit de nauwkeurigheidsmetingen van de casestudies blijkt duidelijk dat GPT-4-o alle andere modellen overtreft. Llama2-modellen toonden een nauwkeurigheid van minder dan 60%. Opmerkelijk is dat de Mixtral 7x8B-modellen superieure prestaties leverden in vergelijking met GPT-4 en GPT-3.5, ondanks dat ze minder parameters en minder VRAM gebruikt. Dit suggereert dat het Mixtral-model een open-source alternatief kan zijn voor propri etaire GPT-4 modellen. Figure 1 en Figure 2 tonen de nauwkeurigheid van de geteste large language models (LLM’s). In de grafieken staat blauw voor propri etaire modellen en groen voor open-source modellen.

<sup>7</sup><https://www.mijnenergie.be/>

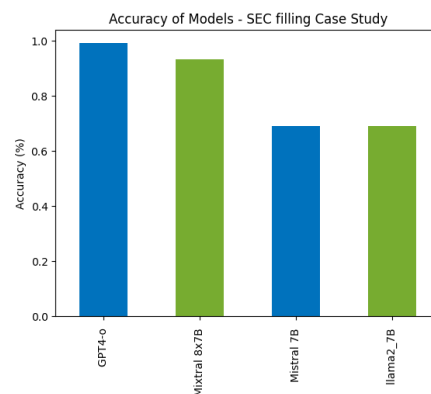
<sup>8</sup><https://investors.coca-colacompany.com/filings-reports/all-sec-filings>

Model Descriptions
L7B refers to LLAMA2 7B
L13B refers to LLAMA2 13B
L70B refers to LLAMA2 70B
M7B refers to Mistral 7B
M8x7B refers to Mixtral 8x7B
M8x7B MD refers to Mixtral 8x7B with input context converted to markdown
GPT3.5 refers to OpenAI's GPT3.5 turbo
GPT3.5 MD refers to OpenAI's GPT3.5 turbo with input context converted to markdown
GPT4 refers to OpenAI's GPT3.5
GPT4-o refers to OpenAI's GPT4 omni

**Table 4:** Full names and descriptions of the models tested.



**Figure 1:** Accuracy of Open-Source and Closed-Source Models on the Energy Contracts Case Study



**Figure 2:** Accuracy of Open-Source and Closed-Source Models on the SEC filling case study

## Beperkingen

Deze studie heeft ook verschillende beperkingen en doet suggesties voor toekomstig onderzoek. De eerste beperking is dat de geteste casestudies hoofdzakelijk slechts twee verschillende structuren van PDF-bestanden omvatten. Hoewel de resultaten van deze studie veelbelovend zijn, representeren ze niet volledig het potentieel van de geteste LLMs. Bovendien was de gebruikte dataset relatief klein. Het testen van de LLMs op twee verschillende datasets in deze casestudies heeft enige overlap laten zien. In de toekomst zou deze studie echter moeten worden uitgebreid naar grotere datasets. De ontwikkelde methoden en technieken kunnen worden toegepast naar mate nieuwe LLMs beschikbaar komen.

Een andere belangrijke beperking van de ontwikkelde LLM-pipeline en de Human-In-The-Loop (HITL) tool is dat ze alleen werken met batchverwerking. Dit betekent dat gebruikers uitsluitend een dataset kunnen laten voorverwerken door de LLM-pipeline, waarbij ze van tevoren de nodige key-value paren moeten specificeren. Daarnaast ondersteunt de HITL geen dynamische interactie met de LLM. Toekomstig werk moet zich richten op het direct verbinden van de HITL met de LLM-pipeline. Voor propriëtaire modellen is dit geen moeilijke taak omdat ze werken met API's. Echter, de manier waarop open-source modellen worden uitgevoerd op de VSC beperkt hen tot batchverwerking, aangezien de VSC geen interactieve node heeft voor de GPU's die voor text generatie worden gebruikt.



Een andere aspect dat niet in deze scriptie wordt behandeld, is het fine-tunen van modellen. Kleinere modellen die slechter presteren, kunnen worden gefinetuned met meer domeinkennis en specifieke prompts om hun prestaties in toekomstig werk te verbeteren. De ontwikkelde LLM-pipeline kan in de toekomst worden gebruikt om deze gefinetunede modellen te testen en te zien hoe ze presteren bij domeinspecifieke taken.

De laatste beperking heeft te maken met het contextvenster van de LLMs. In de geteste casestudies in deze scriptie hebben we de dataset handmatig doorgelicht en alleen de relevante pagina's van PDF's geselecteerd die in het contextvenster van de geteste LLMs passen. Dit proces kan zeer tijdrovend zijn, vooral voor grote datasets met diverse soorten PDF's. Een potentiële oplossing is het gebruik van Retrieval-Augmented Generation (RAG) systemen. Deze aanpak zou de LLM-pipeline in staat stellen om automatisch alleen de relevante invoercontext te extraheren die nodig is voor de huidige prompt die naar de LLM wordt gestuurd. Hoewel we deze aanpak hebben geprobeerd voordat we de relevante data handmatig extraheerden, voegde het extra dimensies van factoren toe die de prestaties van de LLM zouden kunnen beïnvloeden door een onjuiste invoercontext te bieden. Om de tests meer deterministisch te houden, hebben we besloten dit niet te implementeren in de LLM-pipeline en hebben in plaats daarvan vooraf geëxtraheerde PDF's gebruikt die consistent waren bij alle LLM-tests.

Concluderend biedt dit onderzoek aanzienlijke inzichten in de bruikbaarheid en nauwkeurigheid van LLMs, samen met hun inherente beperkingen. De ontwikkelde technieken tonen aan dat LLMs een optie kunnen zijn voor de extractiefase in data-integratietaken. Daarnaast illustreert dit onderzoek dat de ontwikkelde LLM-pipeline en het HITL-tool toekomstig werk kunnen vergemakkelijken door het testen van nieuwe LLMs en het vereenvoudigen van het selectieproces van geschikte LLMs.

# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Problem definition . . . . .	10
1.2	Outline . . . . .	11
<b>2</b>	<b>Background</b>	<b>12</b>
2.1	Neural Networks . . . . .	12
2.2	Natural Language Processing . . . . .	14
2.2.1	Large language model (LLM) . . . . .	15
2.3	Inferencing LLM . . . . .	16
2.3.1	Hyperparameters . . . . .	16
2.3.2	Message Types . . . . .	17
2.3.3	Prompting Techniques . . . . .	17
2.3.4	Providing context . . . . .	18
<b>3</b>	<b>Experiment Setup</b>	<b>19</b>
3.1	Vlaams Supercomputer Centrum (VSC) . . . . .	19
3.1.1	Tier system . . . . .	19
3.1.2	Tier-2 KUL Infrastructure . . . . .	20
3.2	LLM models . . . . .	20
3.2.1	OpenAI Close-source models . . . . .	21
3.2.2	Meta AI Open-source models . . . . .	21
3.2.3	Mistral AI open-source models . . . . .	22
3.2.4	Quantization . . . . .	22
3.2.5	Model specification . . . . .	22
3.3	LLM pipeline . . . . .	23
3.3.1	Data Extraction . . . . .	24
3.3.2	Validation Set Creation . . . . .	24
3.3.3	LLM Inferencer . . . . .	25
3.3.4	Post-Validation . . . . .	25
3.3.5	Visualization and Analysis . . . . .	26
<b>4</b>	<b>Initial Case Study - Energy contracts</b>	<b>27</b>
4.1	Data Selection and Initial Processing . . . . .	27
4.1.1	Initial Method . . . . .	27
4.1.2	Energy comparison website . . . . .	28
4.1.3	Structure of the PDF . . . . .	28
4.2	Experiment . . . . .	28
4.2.1	Goldset creation . . . . .	28
4.2.2	Test runs configurations . . . . .	29
4.2.3	Used models . . . . .	29
4.3	Inferencing the LLM . . . . .	30
4.3.1	Prompting . . . . .	30

4.3.2	Queues on VSC . . . . .	31
4.4	Results and Analysis . . . . .	32
4.4.1	GPT models . . . . .	33
4.4.2	LLAMA2 models . . . . .	33
4.4.3	Mistral models . . . . .	34
4.4.4	Conclusion . . . . .	34
<b>5</b>	<b>Case Study - SEC-filing</b>	<b>37</b>
5.1	Data Source . . . . .	37
5.1.1	Kruispuntbank van Ondernemingen . . . . .	37
5.1.2	SEC-filing . . . . .	38
5.2	Experiment . . . . .	39
5.3	Inferencing the LLM . . . . .	39
5.4	Results and Analysis . . . . .	40
5.4.1	GPT4-o . . . . .	40
5.4.2	Mistral 7x8B . . . . .	40
5.4.3	Conclusion . . . . .	41
<b>6</b>	<b>Human in the Loop Tool (HITL)</b>	<b>43</b>
<b>7</b>	<b>Discussion</b>	<b>48</b>
<b>8</b>	<b>Conclusion</b>	<b>51</b>
<b>9</b>	<b>Appendix A</b>	<b>58</b>
9.1	Prompt templates . . . . .	58
9.1.1	SEC filling prompt . . . . .	58
9.2	Initial case study results . . . . .	59
9.2.1	OpenAI Models . . . . .	59
9.2.2	LLAMA models . . . . .	62
9.2.3	Mistral models . . . . .	64
9.2.4	LLM output visualization in HTML file . . . . .	65
9.3	SEC case study results . . . . .	65
<b>10</b>	<b>Appendix B</b>	<b>67</b>
10.1	English summary . . . . .	67

# Chapter 1

## Introduction

Generative AI has the potential to revolutionize how we handle data, transforming it from a time-consuming and error-prone process into a highly efficient workflow. Case studies indicate that AI can significantly improve worker performance. For example, a study involving 700 consultants from the Boston Consulting Group showed that using ChatGPT<sup>1</sup> increased performance by up to 40%, with users completing 12% more tasks, 25% faster, and producing results of 40% higher quality [5]. In customer service, an IMF<sup>2</sup> study of 5,179 employees found an average productivity increase of 14%, with beginners and low-skilled employees seeing a 35% improvement [37]. This research also highlighted that generative AI shortens the learning curve for new hires, enhances customer experience, and boosts job satisfaction. In software development, a study by MIT, Microsoft, and GitHub on 95 software engineers found that GitHub Copilot increased productivity by 55.8%. This tool was particularly beneficial for beginners, improving their productivity and job satisfaction by automating tedious tasks [54].

While these productivity increases are impressive, AI tools can also provide incorrect answers and exhibit biases [55,65]. This is particularly challenging in environments where data accuracy and consistency are critical, such as the medical or financial sectors. Investigating these issues and understanding the limitations of AI models is the central focus of this thesis.

Verma et al. [34] (2023) introduced a significant contribution to the field of data management with their paper titled “Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes”. Their work tackles a common problem faced by organizations handling large volumes of diverse data: the challenge of extracting structured data from unstructured documents within heterogeneous data lakes. These lakes are large repositories that store a wide variety of data types and sources in their raw form.

By leveraging large language models (LLMs), Verma et al. [34] propose a novel system named Evaporate, which automates the extraction of key-value pairs from unstructured documents. This approach eliminates the need for manual intervention or specialized configuration, thereby streamlining the process of converting heterogeneous data lakes into structured tables.

The researchers explore two primary strategies within Evaporate: direct extraction and code synthesis. Direct extraction involves prompting the LLM to extract values directly from documents without specifying the keys, while code synthesis entails prompting the LLM to generate code for extraction tasks. Their evaluations reveal a trade-off between cost and quality, with direct extraction generally yielding higher accuracy but at a higher computational cost compared to code synthesis.

To address this trade-off, Verma et al. introduce an enhanced implementation called Evaporate-Code+, which generates multiple candidate extraction functions using code synthesis and en-

---

<sup>1</sup><https://openai.com/index/chatgpt/>

<sup>2</sup><https://www.imf.org/en/Research>

sembles their outputs using weak supervision techniques. This approach achieves higher accuracy than direct extraction while maintaining a lower computational cost.

We tested their approach of generating Python code for extraction using Evaporate-Code+. We initially tested it on a small dataset, the generated functions did not correctly answer all the asked questions. Additionally, running their code required numerous calls to OpenAI’s top-tier model, which makes their approach expensive.

## 1.1 Problem definition

Data integration combines data from multiple sources to provide a unified view, crucial for decision-making [25]. However, maintaining consistency and correctness across integrated datasets is challenging, especially as data volume and complexity increase.

In data integration tasks, data is extracted from various sources and processed for use in the complete integration workflow. Ensuring the accuracy of the extracted data significantly improves the overall success rate of the data integration process. This thesis will provide insights on the data extraction aspect of data integration tasks using LLMs.

Analyzing large and complex datasets, commonly referred to as “big data”, poses significant challenges, particularly when dealing with unstructured data that requires substantial manual work. Key challenges [46, 56, 59, 66] in big data analysis include:

- **Volume:** The massive scale of data being generated can overwhelm traditional storage and processing systems. Handling high volumes requires scalable solutions like distributed computing and cloud storage.
- **Velocity:** The speed at which data flows in from various sources, especially real-time streams, necessitates techniques like streaming, event-driven architectures, and auto-scaling to process high-velocity data
- **Variety:** The diversity of structured, semi-structured, and unstructured data from heterogeneous sources requires flexible data management strategies, data integration tools, and master data management to integrate and make sense of the data.
- **Veracity:** Ensuring data accuracy, reliability, and trustworthiness is critical, as poor data quality can lead to flawed insights. Maintaining data veracity requires good data governance, QA checks, standardized naming conventions, and a single source of truth.

Researchers have proposed various analytical methods and tools, such as machine learning, natural language processing, and distributed computing frameworks, to address these challenges. Ongoing research aims to develop more efficient and scalable techniques for managing big data complexities. Also, processing large amounts of unstructured data presents additional challenges [35, 39, 67]:

1. **Heterogeneous Data Formats and Lack of Standardization:** Data is often stored in various unstructured formats, making integration and analysis difficult.
2. **Difficulty in Extracting Meaningful Insights:** Extracting insights from unstructured data like text documents and multimedia is challenging due to the complexity of natural language processing and the need for domain-specific knowledge.
3. **Interoperability Issues:** Integrating data from various sources is hindered by different terminologies, coding systems, and data models.
4. **Data Quality and Reliability Concerns:** Unstructured data may contain errors and inconsistencies, affecting reliability and validity. Ensuring the quality and accuracy of the data is crucial for generating meaningful insights.

In summary, the main problems in processing large amounts of unstructured data for research include data heterogeneity, difficulty in extracting insights, interoperability issues, and concerns

about data quality and reliability. This thesis seeks to investigate the potential of Large Language Models (LLMs) in addressing challenges associated with data integration tasks, with a focus on evaluating their practical applicability.

## Research Questions

The proposed research will address several key questions concerning the performance of large language models (LLMs). These questions are designed to probe the capabilities and limitations of LLMs in executing critical data processing tasks, thus guiding our comprehensive investigation.

1. **Text Extraction Consistency:** Assessing the model’s consistency through multiple test runs to evaluate the consistency of generated responses and address potential hallucinations.
2. **Key-Value Pair Accuracy:** Evaluating the accuracy of extracting key-value pairs to understand the model’s performance in this task.
3. **Open vs. Closed Large Language Models Evaluation:** Comparing the performance of open-source and closed-source LLMs to provide insights into their comparative performance and potential advantages or limitations.
4. **Human Intervention Enhancement:** Exploring the role of human intervention in enhancing LLM outputs by creating a ‘human in the loop’ interface for developers and domain specialists to test and verify generated answers.

## 1.2 Outline

This thesis is organized into several chapters, each addressing a specific aspect of the research. The first chapter, Introduction, provided an overview of the research questions, objectives, and the overall structure of the thesis. Chapter 2 describes the relevant methodologies and foundational knowledge of language processing approaches employed to investigate the application of Large Language Models (LLMs) in data integration tasks. Chapter 3 elaborates on the specific configurations and methodologies utilized to evaluate LLM tests. Chapters 4 and 5 evaluate the accuracy and consistency of LLMs using two different datasets. Chapter 6 discusses the benefits and applications of the developed tool for analyzing results and detecting anomalies in the LLM-generated output. Chapter 7 critically analyzes the findings from the case studies and their implications. Finally, Chapter 8 encapsulates the thesis’ contributions, outlines its limitations, and offers recommendations for future research.

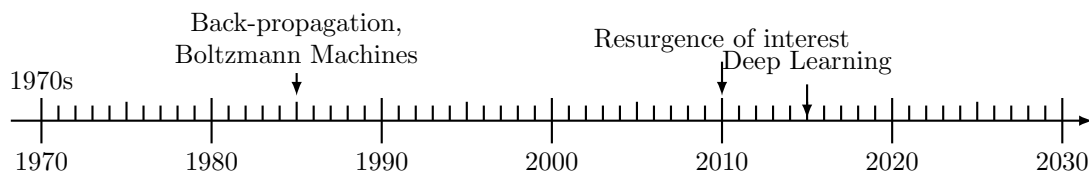
# Chapter 2

## Background

This chapter explains the methodology used to carry out the research in this thesis. We'll provide an overview of the systematic approach we employed and the essential concepts of large language models (LLMs) and machine learning, which are crucial for understanding the results.

### 2.1 Neural Networks

The concept of neural networks dates back to the 1940s with the introduction of the perceptron by McCulloch and Pitts [53]. However, it wasn't until the 1980s and 1990s, with the development of backpropagation and Boltzmann machines, that neural networks became widely used [32,48]. They were applied in various fields such as speech recognition, object recognition, and natural language processing [36,47].



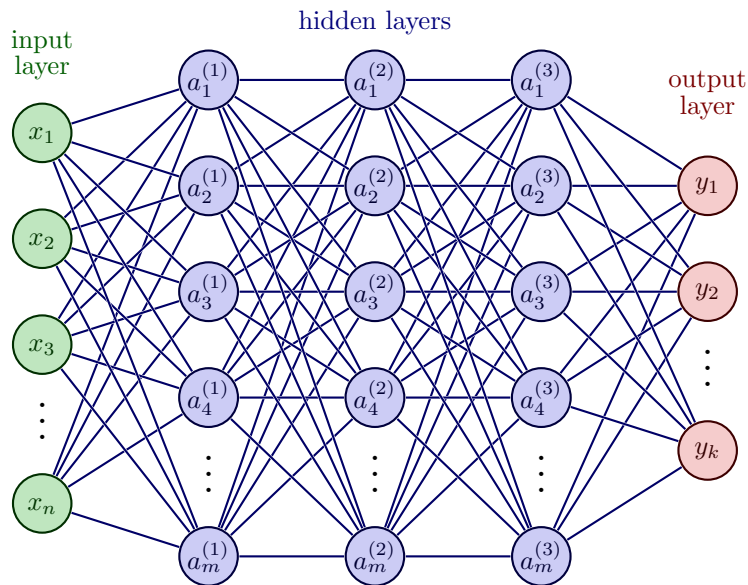
The fundamental idea behind neural networks is that if a phenomenon exists in nature, it can be modeled computationally [49]. This development is largely dependent on advancements in hardware [38]. Companies like NVIDIA<sup>1</sup> have provided powerful hardware solutions, that are optimized for deep learning and AI tasks [51]. These GPUs are efficient for AI model training and are used by major technology companies like Amazon, Google, and Microsoft. Deep learning algorithms have achieved state-of-the-art results in various domains, including computer vision, natural language processing, and drug discovery [52].

Neural networks (NNs) are a pivotal subdomain of machine learning and are significantly utilized in the space of artificial intelligence (AI). They are inspired by the structure and functioning of the human brain, mimicking its architecture by employing artificial neurons or nodes. These nodes serve as the fundamental building blocks of NNs, enabling pattern recognition in data by processing information through the network and learning via the adjustment of internal parameters (weights) during the training phase [9–11,24]. To understand this better, it is vital to dissect the layers that constitute a basic neural network [9,28,30]:

1. **Input Layer:** This layer serves as the entry point for data into the neural network. Each neuron in the input layer corresponds to one feature of the dataset. The input layer

---

<sup>1</sup><https://www.nvidia.com/>



**Figure 2.1:** Visualization of a Basic Neural Network, adapted from Tikz.net

simply passes the data attributes to the subsequent layers without any modification or processing.

2. **Hidden Layers:** Situated between the input and output layers, the hidden layers execute a series of transformations on the input data. These transformations are crucial for capturing complex data patterns. Each neuron in a hidden layer computes a weighted sum of its inputs, adds a bias, and applies an activation function to introduce non-linearity, which enhances the network's ability to learn complex patterns. Multiple hidden layers allow for a deep network structure, enabling the extraction and recognition of complex data patterns, which are essential for accurate prediction or classification tasks.
3. **Output Layer:** The purpose of the output layer is to generate the final predictions or classifications based on the transformations applied by the previous layers. For classification tasks, the Softmax activation function is commonly used in this layer, as it converts the raw output scores into probabilities, simplifying the interpretation and comparison of the model's confidence in various class predictions.

A simple feedforward neural network operates in two main phases: forward propagation and backward propagation.

During **forward propagation**, data flows sequentially from the input layer through the hidden layers to the output layer. Each node in a layer receives input from nodes in the preceding layer, processes this input through weighted connections, applies an activation function, and passes the output to the subsequent layer. The connections between nodes essentially determine the extent of influence each node has on the network's final output [50].

The weighted sum of inputs at each node is transformed using an activation function, such as the sigmoid<sup>2</sup> function, which maps the result to a range between 0 and 1. Each neuron has an associated bias term that adjusts the activation threshold, enhancing the model's flexibility. The network's architecture ensures that each neuron in one layer is connected to every neuron in the preceding layer, with each connection having its own weight and bias.

In the **training phase**, the network relies on labeled training data to adjust the weights and thresholds of its nodes to better fit the input data. The network's performance is evaluated

<sup>2</sup><https://www.sciencedirect.com/topics/computer-science/sigmoid-function>



using a cost function, which quantifies the difference between the network’s predictions and the actual values. The primary objective during training is to minimize this cost function. Different types of cost functions are employed based on the specific task; for instance, Mean Squared Error (MSE) is used for regression problems, binary cross-entropy for binary classification, and categorical cross-entropy for multi-class classification [63].

**Backward propagation** facilitates learning from mistakes using a technique known as gradient descent.<sup>3</sup> This method iteratively updates the model’s weights and biases by moving them in the direction that reduces the cost function. Gradient descent calculates the gradient of the cost function with respect to each weight and adjusts the weights proportionally, reducing the overall error. This iterative process continues until the model achieves minimal error on the training data [50].

While feedforward neural networks form the simplest type of NNs, the core concept remains consistent across different neural network architectures. However, these architectures can vary significantly in how nodes are connected and can include additional components to enhance the basic network. The next section will provide a detailed examination of neural network types that are specifically relevant to Natural Language Processing (NLP).

## 2.2 Natural Language Processing

This section consolidates adapted knowledge sourced from DeepLearning AI [27], an IBM blog post on natural language processing (NLP) [29], and the IBM Technology YouTube channel [64].

Natural Language Processing (NLP) is a branch of artificial intelligence focused on enabling computers to interact with human languages. Generally, NLP starts with unstructured text, such as “Hello, world!”, which lacks inherent meaning to a computer. Through NLP, machines can comprehend, interpret, and manipulate human languages.

NLP can be divided into two subfields: natural language understanding (NLU), which focuses on understanding the meaning of the provided text, and natural language generation (NLG), which focuses on generating text. NLP is used for a variety of language-related tasks, including sentiment analysis, machine translation, named entity recognition, spam detection, chatbots, autocomplete, etc. NLP allows us to create models that can find relationships in language. This is done by combining methods like **data preprocessing**, **feature extraction**, and **language modeling**.

To use textual data effectively in NLP, data preprocessing is essential for creating robust models. One common technique is tokenization.<sup>4</sup> This process converts a text input into a set of tokens, which are individual units of data used during model training. For example, “Hello, world!” might be tokenized into [“Hello”, “,”, “world”, “!”]. These tokens can also be represented as numerical values for use in deep learning models.

Features describe a document’s relationship to the corpus that contains it. These can be created using Bag-of-Words<sup>5</sup> or TF-IDF<sup>6</sup>. Another technique, Word2Vec<sup>7</sup>, uses neural networks to create high-dimensional word embeddings by predicting the surrounding words in a given context. These embeddings encode semantic meaning and relationships between words. GloVe<sup>8</sup> (Global Vectors for Word Representation) learns word embeddings by using matrix factorization techniques, building a matrix based on global word-to-word co-occurrence counts.

---

<sup>3</sup>[https://en.wikipedia.org/wiki/Gradient\\_descent](https://en.wikipedia.org/wiki/Gradient_descent)

<sup>4</sup><https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>

<sup>5</sup><https://www.ibm.com/topics/bag-of-words>

<sup>6</sup><https://en.wikipedia.org/wiki/Tf-idf>

<sup>7</sup><https://en.wikipedia.org/wiki/Word2vec>

<sup>8</sup><https://nlp.stanford.edu/projects/glove/>

Modeling data can be categorized into two types of techniques: **traditional machine learning** and **deep learning methods**.

In traditional machine learning, techniques like logistic regression, naive Bayes, and decision trees can be used for classification. For named entity recognition, hidden Markov models can be utilized.

In deep learning, techniques like CNNs, RNNs, and transformers can be applied to work with language. CNNs are typically used for image processing, but in the paper "Convolutional Neural Networks for Sentence Classification," a method was proposed that uses input sentences as a matrix of words instead of pixels to create text classification. CNNs use close proximity while doing classification, meaning the context provided by a particular text sequence is not captured, and the relation between the current word and words in the previous sentence is not considered.

Recurrent neural network (RNN) solve this problem by remembering information about previous words and sentences, using hidden states in the neural network. These states are limited in storing information, but the long short-term memory (LSTM) architecture solves this problem, allowing information to be remembered for extended periods. A variant of the LSTM, called bidirectional LSTM, can also keep information stored in both directions. This model has a better understanding of the relationship between sequences, as both the following and preceding words of a sentence are used to learn the meaning of the current word.

The paper "Attention Is All You Need" by Vaswani, Shazeer, Parmar, et al. introduced the Transformer architecture, a paradigm shift from traditional RNN-based models. Unlike RNNs, which rely on sequential data processing to retain information, the Transformer employs a self-attention mechanism, facilitating a global context comprehension between input and output. This architecture processes all tokens in parallel, thereby significantly reducing both training and inference times. As a result, the Transformer based models have become a popular choice in natural language generation (NLG) tasks.

This overview of NLP advancements highlights the evolution from conventional deep learning models to the Transformer architecture. The introduction of the Transformer has addressed numerous challenges inherent in traditional models, leading to the development of large language models (LLMs) such as ChatGPT.<sup>9</sup>

### 2.2.1 Large language model (LLM)

A language model is a probabilistic tool that predicts the next word in a sequence based on the provided input words. Foundation models, also known as base models, are pre-trained on extensive datasets that are either unlabeled or self-supervised.<sup>10</sup> These models develop a robust general grasp of language, enabling them to execute a wide range of tasks. Their primary goal is to predict the subsequent word and complete text sequences, rather than just responding to questions [42].

Large Language Models (LLMs) fall under the category of foundation models. These models have numerous parameters that are fine-tuned during training, which increases their ability to handle complex tasks. An LLM is composed of three key components: data, architecture, and training [23].

1. **Data:** LLMs require vast amounts of textual data to develop their capabilities. This data often spans diverse sources such as books, articles, websites, and other forms of written communication. The volume of data utilized can reach terabytes, allowing the model to capture a wide array of linguistic structures, nuances, and contexts. This extensive dataset forms the knowledge base upon which the model builds its understanding of language.

---

<sup>9</sup><https://openai.com/index/chatgpt/>

<sup>10</sup><https://www.ibm.com/topics/self-supervised-learning>

2. **Architecture:** The core structure of most Language Learning Models (LLMs) relies on neural networks. One major breakthrough in this field is the transformer architecture. This innovation has become essential for modern LLMs, allowing them to understand long-range dependencies and complex linguistic patterns more effectively.
3. **Training:** The training process is where the LLM learns to generate coherent and contextually relevant text. During training, the model is exposed to the vast dataset and tasked with predicting the next word in a given sentence. This is achieved through a process known as unsupervised learning, where the model adjusts its internal parameters based on the errors in its predictions. Each word prediction is influenced by previous words in the sequence, and the model continuously refines its understanding by iterating over the training data multiple times. The ultimate goal of the training phase is to fine-tune the model so that it can reliably generate coherent, fluent, and contextually appropriate sentences when given new input.

By integrating extensive datasets, advanced neural network architectures, and rigorous training methodologies, large language models (LLMs) acquire the ability to comprehend and generate human-like text. This capability makes them indispensable assets for various natural language processing (NLP) tasks.

LLMs can be used for general-purpose NLP tasks and can also be fine-tuned on smaller datasets to specialize in specific domain tasks. This process allows a general LLM to become more proficient at the specific task it is fine-tuned for, thus enhancing its accuracy.

It is crucial to recognize that large language models (LLMs) function as probabilistic machines, predicting the next token based on learned patterns. This mechanism can result in hallucinations, a phenomenon where models produce incorrect or fabricated responses. Hallucinations arise due to inherent aspects of LLM training, such as overfitting and biases present in the training data. These issues can have detrimental effects on real-world applications, rendering models vulnerable and unsuitable [22].

The transformation of foundational models into Large Language Models (LLMs) greatly influences their reliability and usability. Instruct models, also called assistant models, are enhanced foundational models trained on datasets with instruction-response pairs. This additional training helps the models better understand and perform tasks based on natural language prompts. Instruct models aim to generate outputs closely aligned with the given instructions [40, 42].

Another common model type is Chat models. These are foundational models trained on conversational data, like dialogue transcripts, to improve their ability to respond coherently and appropriately in chat settings. Chat models are particularly effective for interactive applications such as virtual assistants, where generating natural conversational responses is crucial [42].

## 2.3 Inferencing LLM

LLM inferencing allows users to generate responses from a Large Language Model (LLM). The process begins by sending a prompt to the LLM. This prompt serves as an input that the model uses to understand the user's question. The quality and structure of the prompt significantly affect the LLM's performance.

The model's weights, which are learned during training, cannot be changed during inferencing. Instead, we can influence the LLM's behavior by adjusting the prompt and fine-tuning a few hyperparameters, such as `top_k`, `top_p`, and `temperature`.

### 2.3.1 Hyperparameters

- **Top\_k:** Constrains the model to selecting the top  $k$  most probable next tokens, thereby narrowing the prospective output space and reducing variability.

- **Top\_p:** Alternatively termed nucleus sampling, this parameter includes the smallest set of tokens whose cumulative likelihood is at least  $p$ , allowing for a dynamic selection based on probabilistic weightings.
- **Temperature:** Modulates the entropy of the model's output distribution. Lower temperature values result in more deterministic outputs, whereas higher temperatures yield increased variability and creative responses.

### 2.3.2 Message Types

Chat models are trained to handle three primary types of messages: system, user and assistant. These message types are fundamental for organizing and directing interactions within a chat model, as elucidated in the Hugging Face documentation<sup>11</sup>.

1. **System Messages:** These messages offer system-level instructions and context to guide the interaction between the user and the assistant. System messages can define the behavior of the assistant, set the overall tone, and provide additional context that can influence how the LLM processes user inputs and generates responses.
2. **User Messages:** These messages are inputs from the user, providing context and queries that the LLM needs to respond to. User messages are critical as they set the stage for the LLM's response by specifying the task or question at hand.
3. **Assistant Messages:** These are the responses generated by the LLM. Assistant messages aim to provide accurate, coherent, and contextually relevant information based on the user's input. They are designed to simulate human-like conversation and are central to the interactive experience with the LLM.

By carefully crafting prompts and utilizing these hyperparameters and message types, we can significantly influence the LLM's output and behavior, providing a flexible and controlled interaction tailored to specific needs.

### 2.3.3 Prompting Techniques

Prompting techniques offer a way to effectively guide large language models to generate specific responses. These techniques leverage the structure of prompts to influence the model's output. Four common prompting techniques [13] include:

#### Zero-Shot Prompting

Zero-shot prompting involves providing a single prompt to the model without any additional training data. The model generates coherent responses based solely on the given prompt, demonstrating its ability to generalize and infer information.

#### Few-Shot Prompting

Few-shot prompting expands upon zero-shot prompting by providing a few example prompts with desired responses. The model learns from these examples to produce responses that align with the given prompts, allowing for more precise control over the output.

#### Chain-of-Thought Prompting

Chain-of-thought prompting involves constructing a sequence of prompts where each subsequent prompt builds upon the previous one. This technique helps the model maintain context and coherence throughout the conversation, resulting in more natural and engaging interactions.

---

<sup>11</sup>[https://huggingface.co/docs/transformers/main/en/chat\\_templating](https://huggingface.co/docs/transformers/main/en/chat_templating)

### **Tree of Thoughts (ToT)**

Tree of Thoughts (ToT) is a hierarchical prompting approach that organizes prompts in a tree-like structure. Each branch represents a different topic line, enabling branching conversations and the exploration of multiple concepts within a single interaction.

By utilizing these prompting techniques, researchers and developers can effectively guide LLMs towards generating desirable outputs across a variety of tasks and applications, from dialogue generation to content summarization.

### **2.3.4 Providing context**

In-Context Learning, detailed in the paper “In-Context Retrieval-Augmented Language Models” [57], involves enhancing large language models (LLMs) by prepending documents text to the input prompt without modifying the LLM architecture or requiring additional training.

This method allows the LLM to generate coherent and relevant answer based on a given prompt, including additional context that was not part of the initial training data. However, LLMs with smaller context windows face limitations, as they can only process a limited amount of tokens, reducing their ability to learn effectively from longer or more complex prompts. This approach demonstrates how supplementary data can enable the LLM to answer questions involving new contexts. All our designed prompts adhere to this principle of in-context learning.

## Chapter 3

# Experiment Setup

In this chapter, we explain the experimental setup designed to efficiently run tests on large language models (LLMs), ensuring reliable results and enabling insightful conclusions. This chapter will also describe the developed framework and methodologies, including the large language models selected for testing. Additionally, we provide detailed techniques used for executing memory-intensive LLMs on GPU hardware within a high-performance computing environment.

### 3.1 Vlaams Supercomputer Centrum (VSC)

VSC is a High-Performance Computing (HPC) platform enabling efficient and reliable parallel processing for advanced applications. It consists of tightly connected compute nodes with high bandwidth and low latency communication, managed by a resource manager and scheduler.

VSC is committed to advancing scientific and technical computing within the Flemish academic and industrial sectors. It offers critical infrastructure, specialized training, and comprehensive services, thus playing a pivotal role in scientific inquiries and the facilitation of knowledge transfer between academic research institutions and industry [17,20].

#### 3.1.1 Tier system

The European HPC landscape is structured through initiatives such as the Distributed European Infrastructure for Supercomputing Applications<sup>1</sup> (DEISA) and the Partnership for Advanced Computing in Europe<sup>2</sup> (PRACE). PRACE, which succeeded DEISA, aims to provide access to advanced computing and data management resources for large-scale scientific and engineering applications.

PRACE provides access to advanced computing and data management resources for large-scale scientific and engineering applications. It categorizes European high-performance [14] computing facilities into three tiers: Tier-0 for European centers with petaflop machines, Tier-1 for national centers serving individual countries, and Tier-2 for regional centers covering smaller areas. The VSC follows a similar structure: Tier-0 for large-scale European HPC infrastructure, Tier-1 for regional and national centers, and Tier-2 for resources at individual research institutions.

VSC primarily focuses on Tier-1 and Tier-2 facilities but also aids users in transitioning to Tier-0 by facilitating their preparation for PRACE HPC computing [18]. VSC's Tier-2 infrastructures

---

<sup>1</sup><https://cordis.europa.eu/project/id/508830>

<sup>2</sup><https://digital-skills-jobs.europa.eu/en/organisations/partnership-advanced-computing-europe-prace>

are located within four Flemish universities: UAntwerpen, VUB, HPC-UGent, and KU Leuven (KUL). The infrastructure at KU Leuven (KUL) is employed for our experimental setup of the LLM, facilitated by a shared infrastructure agreement with UHasselt [19].

In this experiment, we employ the Genius cluster<sup>3</sup> and the Wice cluster<sup>4</sup>. Although VSC provides powerful CPUs, we prioritize GPUs due to the primary specification for running open-source models being the amount of Virtual Memory (VRAM) [16].

### 3.1.2 Tier-2 KUL Infrastructure

To facilitate the execution of open-source Large Language Models (LLMs), the infrastructure at KU Leuven (KUL) is employed, comprising two GPU-enabled clusters: the Genius cluster and the wICE cluster. For our tests, we are prioritizing GPUs over CPUs due to the significant VRAM requirements of LLMs [16].

The Genius cluster, operational since 2018, consists of 230 nodes, including 10 high-memory nodes, 22 GPU nodes, and 4 AMD nodes [4]. Within the 22 GPU nodes, 20 nodes are configured with four NVIDIA P100 SXM2 GPUs, each offering 16 GiB of GDDR memory, and 2 nodes are equipped with eight NVIDIA V100 SXM2 GPUs, each providing 32 GiB of GDDR memory. Cumulatively, the Genius cluster incorporates 96 GPU devices, contributing a total of 1792 GiB of GPU memory (1280 GiB from P100 GPUs and 512 GiB from V100 GPUs). The maximum available GPU memory within a single node is 256 GiB, contingent on the use of V100 GPUs, underscoring the cluster’s potential for memory-intensive LLM tasks.

The wICE cluster, operational since February 2023, integrates 240 nodes, inclusive of 5 high-memory nodes, 1 node with extensive memory resources, 8 GPU nodes, and 5 interactive nodes [26]. The GPU nodes are classified into two categories: the first category consists of 4 nodes, each equipped with four NVIDIA A100 SXM4 GPUs, offering 80 GiB of GDDR memory per GPU, and the second category comprises 4 nodes, each featuring four NVIDIA H100 SXM5 GPUs with 80 GiB of GDDR memory per GPU. The aggregated GPU memory capacity of the wICE cluster stands at 2560 GiB (1280 GiB from A100 GPUs and 1280 GiB from H100 GPUs). A single node within this setup can provide up to 320 GiB of GPU RAM. Efficient job scheduling is imperative to leverage these resources optimally, especially due to the 32 GPU devices available.

When fully utilized, the wICE cluster offers a cumulative GPU memory capacity of 1280 GiB from the A100 GPUs and an additional 1280 GiB from the H100 GPUs. A single node can provide up to 320 GiB of GPU RAM. With a total of 32 GPU devices, efficient job scheduling is necessary to manage resources optimally.

A detailed overview of the hardware of the VSC is displayed in Table 3.1

Cluster	GPU Type	Number of Nodes	GPUs per Node	Total GPU RAM	Max GPU RAM per Node
Genius	P100	20	4	1280 GiB	64 GiB
	V100	2	8	512 GiB	256 GiB
wICE	A100	4	4	1280 GiB	320 GiB
	H100	4	4	1280 GiB	320 GiB

**Table 3.1:** Comparison of GPU clusters highlighting different GPU types, node counts, GPUs per node, and RAM configurations.

## 3.2 LLM models

This section describes the different models used in the case studies, focusing on their specifications and usability. Large Language Models (LLMs) are categorized into two main groups:

<sup>3</sup>[https://docs.vscenrum.be/leuven/tier2\\_hardware/genius\\_hardware.html](https://docs.vscenrum.be/leuven/tier2_hardware/genius_hardware.html)

<sup>4</sup>[https://docs.vscenrum.be/leuven/tier2\\_hardware/wice\\_hardware.html](https://docs.vscenrum.be/leuven/tier2_hardware/wice_hardware.html)

1. **Closed-Source Models:** These models have proprietary source code and model weights that are not publicly accessible [60].
2. **Open-Source Models:** These models offer publicly accessible source code, model architecture, and pre-trained weights. Such openness fosters transparency, reproducibility, and allows for extensive customization [43, 61].

To evaluate the consistency and accuracy of LLMs, a selection comprising both open-source and closed-source models is used. Within the closed-source category, the evaluation is exclusively focused on OpenAI’s GPT models.

For open-source models, numerous new options are launched monthly on Hugging Face’s model repository [45]. This thesis primarily examines models from two significant contributors. The first contributor is Meta AI, an artificial intelligence laboratory established by Meta Platforms (formerly Facebook) in 2015. And the second major contributor is Mistral AI, a French enterprise founded in 2023, offering both open-source and closed-source models. Their open-source models can be found on the Hugging Face repository, while the closed-source models are accessible via the Mistral API.

These companies were selected due to their leading open-source models. Meta’s Llama models stood out among the best available options at the time of selection, while Mistral models were chosen for their compliance with European laws, given that Mistral is a European company.

### 3.2.1 OpenAI Close-source models

Generative Pre-trained Transformer (GPT) is a type of large language model (LLM) developed by OpenAI. Several influential GPT foundation models have been released in the “GPT-n” series, with each iteration being significantly more capable than the previous due to increased parameters and training size [41]. For our case studies, we will focus exclusively on testing the chat models: GPT-3.5-turbo and GPT-4.

The GPT-3.5-turbo model exists in multiple versions provided by OpenAI. In this thesis, we use the latest version, gpt-3.5-turbo-0125. This model offers a context window of 16,385 tokens and has been trained up until September 2021. It can return a maximum of 4,096 output tokens [12].

The GPT-4 model (specifically, the gpt-4-0613 version) will be the second model tested. Compared to its predecessors, GPT-4 can solve more complex problems with higher accuracy. However, it is more expensive and has a context window of 8,192 tokens. Its training data goes up to September 2021 [12].

OpenAI recently released a newer model, GPT4-o, which provides a context window of 128K tokens. However, since this model was released at the end of this thesis project, we did not utilize the larger context window in our implementation [12].

All the models launched by OpenAI since the first release of the GPT-1 model in June 2018 are closed-source and proprietary. To access these models, an account on the OpenAI platform is required. Afterward, the models can be used via the platform or through the API. For initial testing, the platform is easier to use, but for running extensive tests, the API will be employed in the LLM-pipeline (see section 3.3).

### 3.2.2 Meta AI Open-source models

Llama2<sup>5</sup> models, developed by META AI, are a series of large language models that are open-source and available for both research and commercial use. These models are trained on 2 trillion tokens and support a context window of 4096 tokens.

---

<sup>5</sup><https://llama.meta.com/llama2/>



The Llama 2 series includes models with 7 billion (7B), 13 billion (13B), and 70 billion (70B) parameters. While these foundational models are useful, our study focuses on the chat variants of these models. These chat models are fine-tuned versions of the foundational models, optimized using techniques like supervised fine-tuning<sup>6</sup> and Reinforcement Learning with Human Feedback<sup>7</sup> (RLHF). RLHF uses human feedback to adjust the model’s weights, improving its ability to generate accurate and contextually appropriate responses.

Meta AI has recently released the LLAMA3 models, which are offered in parameter sizes of 8 billion (8B) and 70 billion (70B), featuring an expanded context window of 8,000 tokens. Due to their release being timed too closely to the end of our project, these models were not included in the scope of our study or the subsequent testing phases.

### 3.2.3 Mistral AI open-source models

Mistral specializes in the development of large language models.<sup>8</sup> They produce both open-source and closed-source models. This study exclusively examines Mistral’s open-source models.

The Mistral 7B model, which contains 7.3 billion parameters, is released under the Apache 2.0 license. Based on the transformer architecture, this model was made available in September 2023 and supports a context window of 8,000 tokens. We will use the fine-tuned “instruct” version of this model.

Additionally, we will employ the Mixtral 8x7B model, which uses a sparse mixture of experts architecture<sup>9</sup> and is also released under the Apache 2.0 license [31]. This model supports a context window of up to 32,000 tokens. For our study, we will also use the fine-tuned “instruct” variant of this model.

### 3.2.4 Quantization

Model quantization is a technique employed in deep neural networks to reduce memory usage and computational overhead by converting model weights from standard floating-point data types to lower precision data types. Typically, this involves changing 32-bit floating-point representations to 16-bit. Such conversion saves memory and increases inference speed, thereby reducing computational complexity. Quantization is essential for deploying models on edge devices, which usually have limited computational power and memory. It speeds up inference, making real-world applications feasible on these resource-constrained devices. However, using lower precision data types can decrease the model’s accuracy, as quantization reduces the model’s weight precision [62].

### 3.2.5 Model specification

For open-source models, specifications are crucial as they help determine the required hardware to run these models. In contrast, closed-source models can be used on any hardware, since they make requests to the vendor’s infrastructure using an API.

To check if a Large Language Model (LLM) can fit into the available GPUs, we need to calculate the total memory space in bytes. This helps us determine if a model can run for inference. Methods outlined by Quentin Anthony, Stella Biderman, and Hailey Schoelkopf [33] can be used to calculate the model’s inference memory size.

The following equations detail the memory requirements for different quantization levels:

<sup>6</sup>[https://klu.ai/glossary/supervised\\_fine\\_tuning](https://klu.ai/glossary/supervised_fine_tuning)

<sup>7</sup>[https://en.wikipedia.org/wiki/Reinforcement\\_learning\\_from\\_human\\_feedback](https://en.wikipedia.org/wiki/Reinforcement_learning_from_human_feedback)

<sup>8</sup><https://mistral.ai/>

<sup>9</sup><https://huggingface.co/blog/moe>

- In int 8:

$$\text{Memory} = (1 \text{ byte/param}) \times (\text{number of parameters})$$

- In floating point 16:

$$\text{Memory} = (2 \text{ bytes/param}) \times (\text{number of parameters})$$

- In floating point 32:

$$\text{Memory} = (4 \text{ bytes/param}) \times (\text{number of parameters})$$

To determine the GPU memory requirements, we utilized the application “can-it-run-llm”<sup>10</sup> developed by Vokture, which adheres to the methodology outlined previously. Table 3.2 shows the VRAM usage of each open-source model considered in this thesis, specifying their requirements for both full quantization (float 32) and half quantization (float 16).

Model	VRAM (f32)	VRAM (f16)
llama2_7B	≈ 30 GB	15 GB
llama2_13B	≈ 58 GB	29 GB
llama2_70B	≈ 308 GB	77 GB
mistral_7B	≈ 33 GB	16 GB
mistral_8x7B	≈ 209 GB	105 GB

**Table 3.2:** VRAM Usage for Different Models

### 3.3 LLM pipeline

This section details the construction of a pipeline designed for efficient execution of the case studies, herein referred to as the Large Language Model (LLM) pipeline. This section offers a comprehensive overview of the chosen technologies and the design principles applied in building the LLM-pipeline.

The pipeline comprises five main stages (see Figure 3.1): **1) Data Extraction**, **2) Validation Set Creation**, **3) LLM Inferencing**, **4) Post-Validation**, and **5) Visualization and Analysis**. It begins by accepting raw data files in either HTML or PDF format. This dataset is then input into the Data Extraction module. After processing, the pipeline outputs a JSON file containing the test results, which are ready for further analysis.

#### File Structure for Input Data

To evaluate various files within a dataset, it is essential to configure the directory where unstructured data is stored. The prescribed folder hierarchy for efficient execution of the LLM pipeline is as follows: `usecase/raw/[type]/`, where the `type` subdirectory can be `pdf`, `text`, or `excel`.

During the initial execution phase, the LLM pipeline systematically generates the required directory structure tailored to the specific case study. This structure comprises an `intermediate` directory, into which the data extraction module transposes raw data into standardized, usable formats. Additionally, it includes a `generated` directory, which the LLM inferencer populates with the resulting data outputs from the execution process.

Additionally, a `log` folder will be created to log any errors that occur while running the LLM for faster debugging. This folder will also keep track of the costs associated with API calls to closed-source models.

<sup>10</sup><https://huggingface.co/spaces/Vokturz/can-it-run-llm>

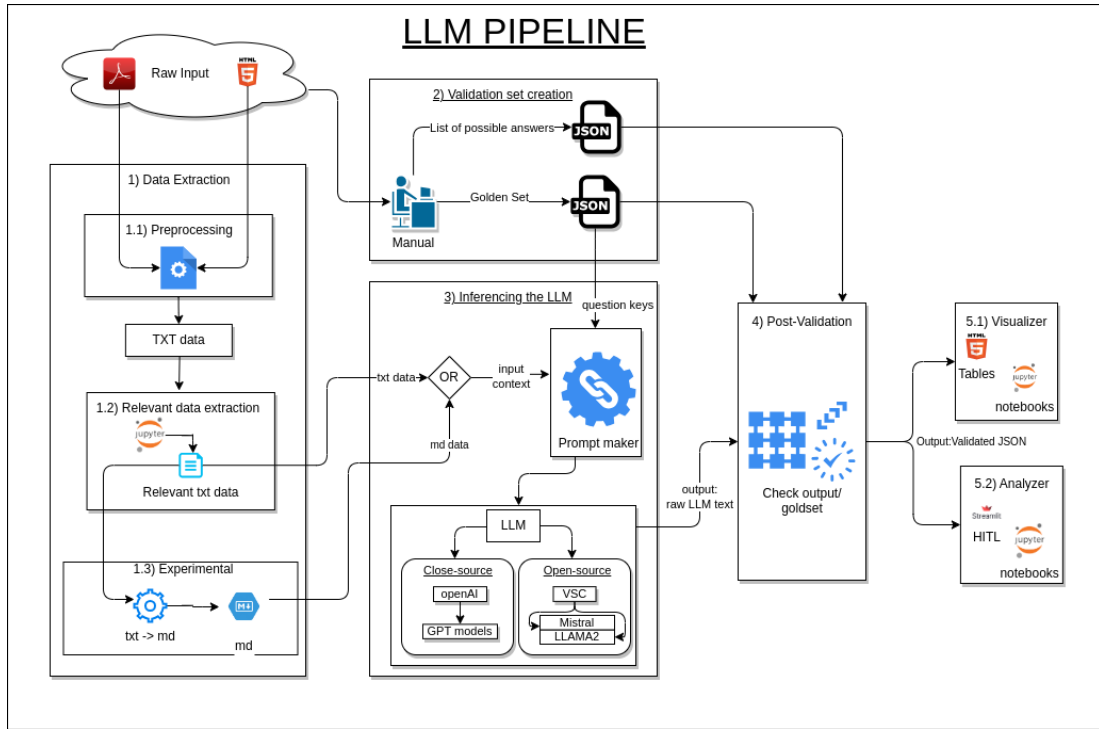


Figure 3.1: Five main modules of the LLM-pipeline

### 3.3.1 Data Extraction

This module has two main responsibilities: **1) preprocessing data**, and **2) acquiring relevant data**.

The input data is converted into textual form using python scripts, as shown in Figure 3.1. Currently, the LLM pipeline exclusively handles PDF documents and raw text files, as these formats are the most prevalent in the case studies we’ve examined. This preprocessing step includes data cleaning to ensure better quality and compatibility. Future work could expand the preprocessing module to support additional data formats.

Due to the limited context window size of the LLMs utilized, relevant data extraction is performed on the input files. This extraction is accomplished using scripting, which selectively extract the necessary pages for the testing phase. The extraction process involves partial manual intervention to streamline the processing. This phase effectively transforms the original dataset into a reduced dataset containing only the essential data.

An experimental module has been developed to convert relevant textual data into Markdown. This module utilizes an LLM to transform raw text files into Markdown format. In the case studies, we will investigate whether this approach yields any performance improvements over raw text formats as input data to the LLM.

### 3.3.2 Validation Set Creation

Creating a validation set is crucial for automatically checking the output of the Language Learning Model (LLM). This step requires extracting the ground truth key-value pairs from the raw input, where the key represents the question, and the value denotes the correct answer. If no metadata containing these key-value pairs is available for the unstructured raw input data, the extraction process is performed manually.

Two sets of JSON files must be created if this is completed manually. The first set is the

“goldenset” JSON, which contains key-value pairs for each document in the input dataset, corresponding to the specific questions and answers for that document. The second set is the “answerset” JSON, which assists the post-validator in identifying the type of key-value pair. For example, a value that can only be a month will have an item mapping the key to a dictionary of all the months. Similarly, if the value is a double or a string, this set will provide that information.

If metadata is available, the otherwise manual process can be automated. Currently, our implementation supports metadata in Excel or XLSX formats. It should be noted, however, that this process still necessitates some manual intervention. While there are scripts available to convert the metadata into a goldenset, these scripts may require modification tailored to the specific case study.

### 3.3.3 LLM Inferencer

This module is responsible for generating prompts for various Language Learning Models (LLMs) using the Promptmaker module. Parameters such as model type, temperature, test runs, and case study can be specified as arguments when initializing the LLM pipeline. The Promptmaker module uses the goldenset in combination with input context fed from the data extraction module to build the prompts. These prompts are a set of templates we developed that are automatically updated based on the selected model.

The LLM prompter module is the core component of the entire pipeline. All models inherit from a base LLM class we developed, which uses Langchain<sup>11</sup> for model inference. Common parameters across different models are set in the superclass. The subclassed modules include Mistral, LLAMA, and OpenAI prompters. By setting the model at runtime, the pipeline automatically detects the correct architecture needed to run the LLM. Newer types of models can be easily added by extending the LLM class. Predefined models listed in the config.ini file facilitate easy switching between models.

The LLM pipeline can be executed on local machines with sufficient GPU memory for open-source models, for close-source models the LLM pipeline uses the API provided by the vendor, as in the case of OpenAI models. For LLMs requiring execution on a VSC, the full codebase needs to be cloned into the VSC. Visualization and analysis can be performed either on the VSC or locally once post-validation is completed.

#### Utils

To run the LLM pipeline on the VSC, an additional wrapper script has been developed. This script accepts the same arguments as the LLM pipeline but includes extra parameters specific to the VSC environment. The script features an integrated configuration file, which we added to enable the wrapper script to automatically generate the SLURM<sup>12</sup> script required for the job. This script configures the necessary GPUs and the appropriate cluster based on the specified arguments. Once the SLURM script is built, it is automatically submitted to the VSC scheduler for execution.

### 3.3.4 Post-Validation

The post-validation step ensures that raw text data from the LLM is converted to structured data, which can then be easily visualized and analyzed to gain valuable insights.

We designed the prompts in a way that the LLM provides its responses in a valid JSON format. However, the LLM output is initially in a string format and needs conversion to JSON. Occasionally, the LLM returns additional text data, which cannot be converted directly to JSON the post-validation step also cleans the LLM output, ensuring it is in the proper format.

---

<sup>11</sup><https://www.langchain.com/>

<sup>12</sup>[https://docs.vscenrum.be/jobs/running\\_jobs.html](https://docs.vscenrum.be/jobs/running_jobs.html)

The formatted JSON object is compared against the *goldenset* and the *answerset*. This comparison identifies which keys were correctly extracted by the LLM and which key-value pairs were missed. Data cleaning procedures are applied to the generated values, including converting text to lowercase, removing extraneous spaces, and standardizing numeric representations.

### Validation metrics

Validation is based on the number of correct and incorrect answers. Initially, we tried calculating precision, recall, and the F-score, but these metrics did not provide the desired insights. We then used a majority vote system, where the answer with the most votes (e.g., 4 incorrect vs. 6 correct results in a correct answer) determines the final result. The downside of this method is that it does not allow for precise accuracy computation.

The final version of the validation operates using the number of correct and incorrect answers. Dividing the number of correct answers by the number of incorrect answers provides a better overview of the LLM's performance on each contract.

### 3.3.5 Visualization and Analysis

The visualization and analysis components of the LLM-pipeline can be considered as separate modules that do not interfere with the pipeline's primary function. The pipeline generates a validated JSON file, which serves as the basis for visualization. Initially, this visualization was displayed in an HTML table (see appendix subsection 9.2.4), showing the number of correct and incorrect answers for each contract.

To conduct a more detailed analysis, data analysis notebooks are employed. However, this traditional approach does not allow for easy or dynamic data exploration and is not very user-friendly. Consequently, we developed a user-friendly analyzer tool called "HITL". More details about this tool will be provided in chapter 6.

## Chapter 4

# Initial Case Study - Energy contracts

This initial case study aims to evaluate a range of techniques and methods for data extraction, with the primary goal of facilitating an evaluation of various Large Language Models. The study encompasses both open-source and closed-source models.

This chapter proceeds as follows: First, we examine the sources of our dataset. Next, we preprocess the dataset to ensure compatibility with our LLM pipeline. After that, we identify key metrics for comparison. Then, we test the data using various prompting techniques and different LLM models, both open-source and closed-source.

At the conclusion of this case study, we will evaluate all acquired knowledge and draw conclusions. The methods and techniques applied here will then be tested with a different dataset in the next chapter, where we will repeat the process with a different document layout to ensure unbiased findings.

### 4.1 Data Selection and Initial Processing

Any available datasets containing unstructured data can be used at this phase. For this case study, we chose PDFs as the primary raw dataset due to their easy accessibility. Considering the surge in energy prices [44] at the time of writing, energy contracts in PDF format were selected as an appropriate data source. To reduce complexity during LLM inferencing, we included only the relevant data while building and sending prompts to the LLM, considering the limited number of tokens provided by an LLM's context window.

#### 4.1.1 Initial Method

To gather a variety of contracts, we first reviewed different energy providers' offerings. For the initial case study, we manually searched several energy providers and sourced three contracts. We also developed a basic web automation tool to help extract energy contract PDFs, but encountered several challenges while scraping data from unstructured websites. These challenges included inconsistent structures, dynamic content, complex nested elements, inconsistent naming conventions, and lack of metadata [58]. Due to the encountered challenges, the creation of the dataset through the original method is considered unfeasible. Consequently, the dataset is created using the platform Mijnergie.be<sup>1</sup>.

---

<sup>1</sup>Mijnergie.be

### 4.1.2 Energy comparison website

Mijnenergie.be<sup>2</sup> is an independent energy comparison website in Belgium that allows users to compare energy prices from different suppliers and select the most cost-effective energy provider online. It provides a detailed overview of current energy prices across all cities and municipalities in Belgium, enabling users to assess if they are paying too much for their energy [6–8]. This website plays a crucial role in helping consumers navigate the liberalized gas and electricity market by simplifying the process of changing energy suppliers without hassle.

The PDF documents utilized in this case study are downloaded from this website, which offers a comprehensive overview of all the energy contracts of various suppliers over the past three months.

### 4.1.3 Structure of the PDF

The energy contract PDFs we download contain two main areas that we utilize: simple values in paragraphs and tabular values. Figure 4.1 shows an example PDF from the dataset, highlighting these two groups. By analyzing both simple values and tabular data, we can evaluate the LLM’s ability to manage and interpret various forms of unstructured information.

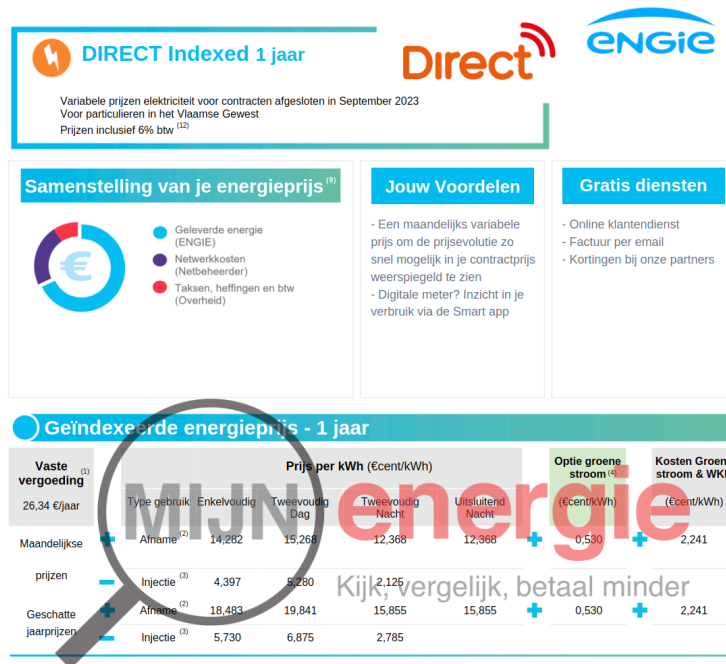


Figure 4.1: Engie’s energy contract, showing the structure of a contract. Source: Mijnenergie.be

## 4.2 Experiment

### 4.2.1 Goldset creation

To rigorously evaluate the precision and consistency of a large language model (LLM), we employed a “goldset”. This Goldset comprises a meticulously selected set of key values that act as a ground truth reference. Using this Goldset, we systematically compare the LLM generated outputs against the goldset to determine the model’s accuracy and consistency.

<sup>2</sup>Mijnenergie.be

The proposed method for creating a GoldenSet involves two main steps:

1. **Identifying keys easily found in the input text:** These keys should be present in a paragraph or section of the input document.
2. **Selecting keys present in tables:** Tables often contain important information in a structured format, but this structure is lost when PDFs are converted to text. By including keys found in tables, we can assess the LLM’s ability to understand and extract information from tabular data.

By combining these two types of keys, the GoldenSet provides a comprehensive evaluation of the LLM’s understanding of the input data, encompassing both textual content and structural elements.

To create the GoldenSet, we started with a single PDF, which involves significant manual effort, as detailed in the LLM pipeline (see Section 3.3). The primary limitation of this case study is the time-consuming process necessary to develop the GoldenSet for PDFs. This requires meticulous examination and extraction of relevant data.

VREG [21] uses the same PDF documents and generates an Excel sheet for their V-Test [15]. This Excel sheet can serve as a metadata set. However, the values in their Excel sheet do not match our PDF dataset. Therefore, we manually compile a GoldenSet by selecting 8 PDFs from different energy suppliers for this case study.

## 4.2.2 Test runs configurations

In this subsection, we detail the configurations employed for conducting the test runs. These configurations were designed to evaluate the performance of the Language Learning Models (LLMs) across various scenarios. The following configurations and types of tests were implemented to ensure a thorough assessment of the LLM’s capabilities. The number of PDF files used for testing was 8, and each file underwent 10 test runs.

Three types of tests were conducted. The first type was the verification of the LLM’s ability to reproduce text character by character. The second type involved querying individual keys, where the model’s response to absent keys was tested with the expectation of receiving a “not found” result. The third type of test involved the simultaneous querying of all keys within a contract.

## 4.2.3 Used models

This case study employs all models referenced in Section 3.2 to draw conclusions about their performance. Initially, we conducted local tests using the Ollama<sup>3</sup> ecosystem, built on llama.cpp<sup>4</sup>, an open-source, high-performance inferencing system. Ollama was chosen for local testing due to its cost-free nature and flexibility in handling diverse prompts and settings. Our LLM pipeline is implemented with LangChain, which simplifies model switching and inferencing processes.

A significant limitation of local testing with open-source models via Ollama is the hardware dependency. For example, smaller models such as the 7B models can operate on systems with 8 GB of RAM<sup>5</sup> using quantization, whereas larger models require more substantial resources and cannot be run locally. Consequently, for comprehensive experiments involving all models, we utilized the Vlaams Supercomputer Centrum (VSC) environment.

---

<sup>3</sup><https://www.ollama.com/>

<sup>4</sup><https://github.com/ggerganov/llama.cpp>

<sup>5</sup><https://github.com/ollama/ollama>



## 4.3 Inferencing the LLM

### 4.3.1 Prompting

In this subsection, we examine and assess the performance of the prompts we tested. To ensure fair comparisons, we use the same prompt for all LLAMA2 and Mistral models, with slight syntax modifications for the GPT models.

Each prompt includes a system message (as detailed in 2.3.2), specifying the purpose of the LLMs. The following prompt is used in all models after extensive testing:

---

```
System {
    This agent is designed to extract key values from the provided input
    context related to [Domain]. If the agent does not have an answer for a
    specific key, it should respond in the desired JSON format, indicating
    that there is no value for that key to recommend at the moment. Let's
    proceed step by step.

    Desired format = JSON format: {<key>: <value>}. Return only this format;
    the question is always the key; do not change that; if there is no
    value for a key, return {<key>: "Not found"}.
}
```

---

Subsequently, an example input context is provided, representing the raw output that the LLM-pipeline data extractor module converted from the input PDF. The few-shot prompting technique is utilized to facilitate in-context learning, wherein the LLM is supplied with a sample question and the appropriate response it should generate. The following prompt is appended to the system message, thus completing the prompt assembly.

---

```
User {
    inputcontext = [Example input Context from contract]
    Q: naam van contract, afname prijs tweevoudig nacht
}

Assistant {
    { "naam van contract": "Luminus Comfy Elektriciteit",
      "afname prijs tweevoudig nacht": "20,30"      }
}

User {
    Above where example's use same approach to answer question below; Do not
    use the input context defined above; Use input context provided in the
    input context field below.

    inputcontext= [Input Context for the tested contract]
    Q: [Set of keys that we want to extract]
}
```

---

A key sentence in our prompt is:

---

```
Above where examples use the same approach to answer the question below; Do not use
the input context defined above; Use the input context provided in the input
context field below.
```

---

We crafted this sentence to prevent the LLM from mixing the given context with the example context. Context mixing happens when the LLM confidently generates an incorrect response

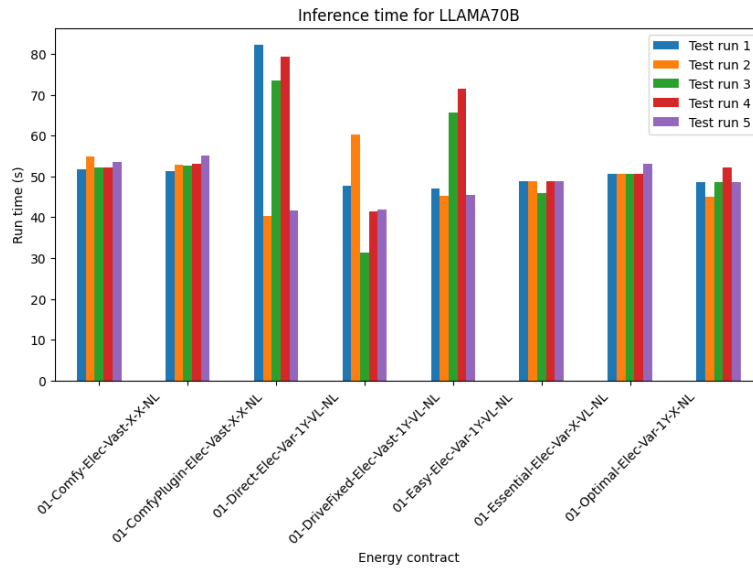
based on examples rather than the provided input context. This issue was observed only in open-source models and could not be replicated in closed-source models. We addressed this by iteratively testing and refining the prompt, which was challenging due to the lack of extensive documentation on prompt structure for open-source models.

To mitigate the issue of hallucinations, we explicitly instruct the Large Language Model (LLM) to return the phrase “Not found” when it encounters scenarios where the information is not found in the input context.

The template for prompts is crafted for adaptability across all models we utilize. Although the specific prompt sent to the Language Learning Model (LLM) might differ, the core principle remains unchanged. The Huggingface library provides a chat template<sup>6</sup> functionality that can effectively adapt this prompt for use with other models.

### 4.3.2 Queues on VSC

Testing larger models on the VSC presents several challenges. One major issue is the longer inference times required for prompt testing by these models (see Figure 4.2). This necessitates scheduling resources for longer durations. We observe that VSC resources, particularly V100 and A100 GPUs, are heavily used during weekdays. However, testing on weekends significantly improves our progress, as tests run faster and can be analyzed more quickly.



**Figure 4.2:** Inference times on the VSC for the LLAMA2 70B parameters, with 5 test runs for each contract and a total of 7 contracts

Running the LLM pipeline to test 7 contracts with 5 test runs each took around 35 minutes. Inference times vary between models; running the same test on the Mixtral model took 40 minutes. It is noteworthy that the execution duration may escalate substantially when larger datasets are employed.

In the initial phase, we ran larger models on wICE A100 GPUs, using 3 GPUs per job. However, because of the long test times and the limited number of A100 GPUs available, we switched to the Genius cluster. To address prolonged queue times, we then transitioned to V100 GPUs, allocating 8 GPUs per job. Currently, all tests are conducted using V100 and P100 GPUs.

<sup>6</sup>[https://huggingface.co/docs/transformers/main/en/chat\\_templating](https://huggingface.co/docs/transformers/main/en/chat_templating)

## 4.4 Results and Analysis

In this section, we delve into the results obtained from the test runs. The primary aim of these tests is to assess the accuracy of the language models (LLMs) and identify any patterns across all the test runs. Each subsection provides a detailed examination of the results of the specific LLM used. The outputs are detailed in appendix section 9.2. By distinguishing the result outputs from the analysis, we articulate a clearer interpretation of each LLM model's performance metrics.

### Initial tests on the LLM

Several sub-questions were formulated to gain preliminary insights into the models. Initially, we aim to evaluate if a Large Language Model (LLM) can generate output purely based on the input provided, achieving a one-to-one correspondence between input and output. This specific evaluation is conducted exclusively on the GPT-3.5 model and the Llama 7B model, as the larger models are derivatives of these foundational models. Our tests reveal that both models achieve 100% accuracy in this task. It is particularly notable that the input text stays within the 4000-token limit.

Subsequently, we perform a subtest by prompting the LLM with one question to obtain a single key value. Figure 4.3 displays the results of this test performed using GPT-3.5-turbo.

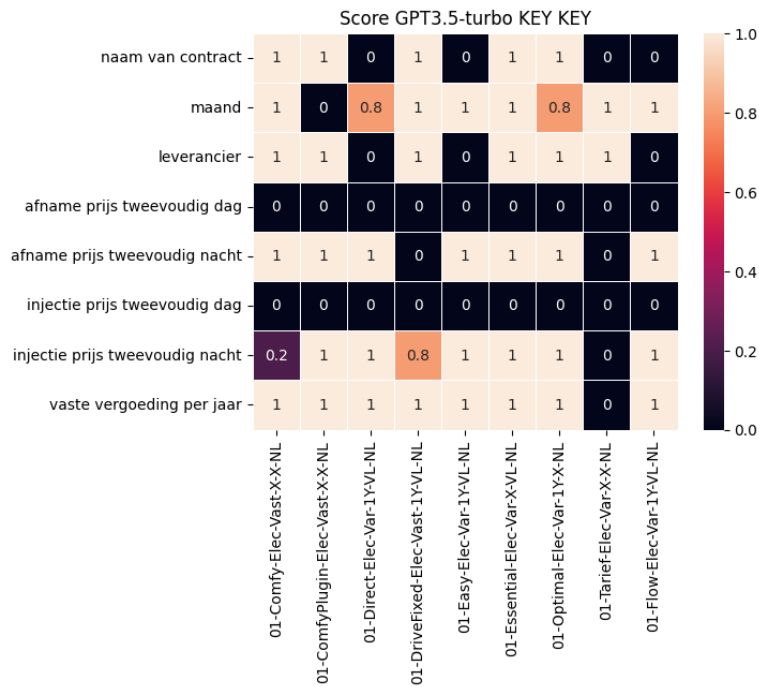


Figure 4.3: Key by key output of GPT-3.5 turbo for the energy use case

This matrix illustrates the accuracy of the test. Each cell displays the percentage reflecting the LLM's performance across multiple test runs. In this test, each document key was prompted five times, and the percentage of 0 or 1 indicates whether the LLM correctly identified the answer (1) or failed to do so (0) compared to the goldset for that energy contract. Overall, we conducted 5 tests per key, with 8 keys per file and 9 sets of PDFs, totaling 360 prompts sent to the OpenAI API.

One limitation of this approach is its inefficiency, as it requires constructing the prompt with the entire PDF context for each call, resulting in unnecessary API calls.

Following these initial tests, subsequent tests employed a different method of prompting the LLMs. In this new approach, we extended the prompts to answer a set of key questions. This significantly reduced the number of inferences, potentially by a factor of 8 if there are 8 keys. We applied this method to tests on both closed-source models and open-source models.

#### 4.4.1 GPT models

In our experimentation to evaluate the effectiveness of various OpenAI models, namely GPT3.5 turbo, GPT4, and the GPT4-o model, we followed a systematic approach. This involved prompting the Large Language Model (LLM) with the document’s context and a predefined set of keys, as outlined earlier. To ensure objectivity and mitigate any potential biases, we executed multiple test runs over the same document.

For each column in Figure 4.3, an intermediate score was calculated by summing all the accuracies from the test runs and then dividing this total by the number of keys for that column. This provided the accuracy for each specific contract. The process was repeated for all contracts to obtain an overall score.

The resulting scores for each contract, along with the respective LLM utilized, are presented in the table below:

Contract Name	GPT-3.5 MD	GPT-3.5 Turbo	GPT-4	GPT-4.O
01-Comfy-Elec-Vast-X-X-NL	90	61	90	88
01-ComfyPlugin-Elec-Vast-X-X-NL	60	63	88	100
01-Direct-Elec-Var-1Y-VL-NL	35	86	85	100
01-DriveFixed-Elec-Vast-1Y-VL-NL	55	88	85	93
01-Easy-Elec-Var-1Y-VL-NL	95	41	83	95
01-Essential-Elec-Var-X-VL-NL	55	53	50	93
01-Optimal-Elec-Var-1Y-X-NL	88	62	98	98
01-Tarief-Elec-Var-X-X-NL	10	52	38	30
<b>Total Score</b>	<b>61</b>	<b>63</b>	<b>77</b>	<b>87</b>

**Table 4.1:** Accuracy Scores for OpenAI Models (Refer to Appendix A, Section 9.2.1 for detailed accuracy matrices), Representing Model Performance

The scoring system ranges from 0 to 100, with 100 representing the case where the model correctly answered all questions. Analysis of the table reveals that the GPT-4-o model outperforms all other tested models.

A notable pattern observed in less effective models, such as GPT-3.5 Turbo, is that key-value pairs within tables (e.g., ‘afname prijs tweevoudig dag,’ ‘afname prijs tweevoudig nacht,’ ‘injectie prijs tweevoudig dag,’ and ‘injectie prijs tweevoudig nacht’) present significant challenges. Refer to Appendix A section 9.2.1 for a comprehensive accuracy matrix.

#### 4.4.2 LLAMA2 models

The Llama2 model’s scoring methodology aligns closely with that of the GPT models. Comparative analysis indicates a marginal performance enhancement, with the largest model (70-billion parameters) achieving only a 7-point increase in accuracy over the smallest model (7-billion parameters). Despite the substantially higher VRAM and memory demands, alongside increased inference times for the LLAMA 70B model.

An observed consistent pattern across all LLAMA models, is their suboptimal accuracy in processing keys embedded within tables. Moreover, these models occasionally interchange keys located in adjacent columns (see Appendix A section 9.2.2).

<b>Contract</b>	<b>7B</b>	<b>13B</b>	<b>70B</b>
01-Comfy-Elec-Vast-X-X-NL	48	50	70
01-ComfyPlugin-Elec-Vast-X-X-NL	32	50	65
01-Direct-Elec-Var-1Y-VL-NL	48	50	38
01-DriveFixed-Elec-Vast-1Y-VL-NL	28	40	27
01-Easy-Elec-Var-1Y-VL-NL	55	52	35
01-Essential-Elec-Var-X-VL-NL	50	62	75
01-Optimal-Elec-Var-1Y-X-NL	32	50	50
01-Tarief-Elec-Var-X-X-NL	32	08	25
<b>Total Score</b>	<b>41</b>	<b>45</b>	<b>48</b>

**Table 4.2:** Accuracy Scores for LLAMA2 Models (Refer to Appendix A, Section 9.2.2 for detailed accuracy matrices), Representing Model Performance

### 4.4.3 Mistral models

<b>Contract</b>	<b>7B</b>	<b>8x7B</b>	<b>8x7B MD preprocessed</b>
01-Comfy-Elec-Vast-X-X-NL	47	92	90
01-ComfyPlugin-Elec-Vast-X-X-NL	41	100	75
01-Direct-Elec-Var-1Y-VL-NL	22	80	72
01-DriveFixed-Elec-Vast-1Y-VL-NL	44	75	72
01-Easy-Elec-Var-1Y-VL-NL	19	48	70
01-Essential-Elec-Var-X-VL-NL	41	92	78
01-Optimal-Elec-Var-1Y-X-NL	28	92	55
01-Tarief-Elec-Var-X-X-NL	45	50	38
<b>Total Score</b>	<b>36</b>	<b>79</b>	<b>69</b>

**Table 4.3:** Accuracy Scores for Mistral Models (Refer to Appendix A, Section 9.2.3 for detailed accuracy matrices), Representing Model Performance

For the Mistral line of LLMs, two models were tested: the Mistral 7B model and the Mixtral 8x7B model. Solely looking at the scores of these models, there is a significant performance boost. The 8x7B model achieves an accuracy that is 45 points higher than the 7B mistral model.

Examining the 7B model, it struggled in both to identify the plain text key-value pairs and the keys within tables. A closer inspection of the score heatmap in Figure 4.5 reveals that the values in tables contain more zeros, indicating that the model frequently failed to provide the correct answers across all test runs.

In contrast, the Mixtral 8x7B model, shown in Figure 4.4, performs well at identifying key-value pairs in paragraphs. However, it consistently struggles across all documents to find the “injectie prijs tweevoudig dag” and “injectie prijs tweevoudig nacht”, which lowers the score for key-value pairs in tables.

In tests on both GPT and Mixtral models, we evaluate whether converting raw text input to Markdown improves model accuracy. Initial tests with a single document show that Markdown-converted input achieves a higher accuracy score. However, when we extend the experiment to all input PDF documents, both models display a decrease in accuracy compared to using raw text, as detailed in Table 4.2 and Table 4.3.

### 4.4.4 Conclusion

To determine the efficacy of the proposed method, a series of experiments were performed using a Large Language Model (LLM) without any fine-tuning. The experiments allowed for an

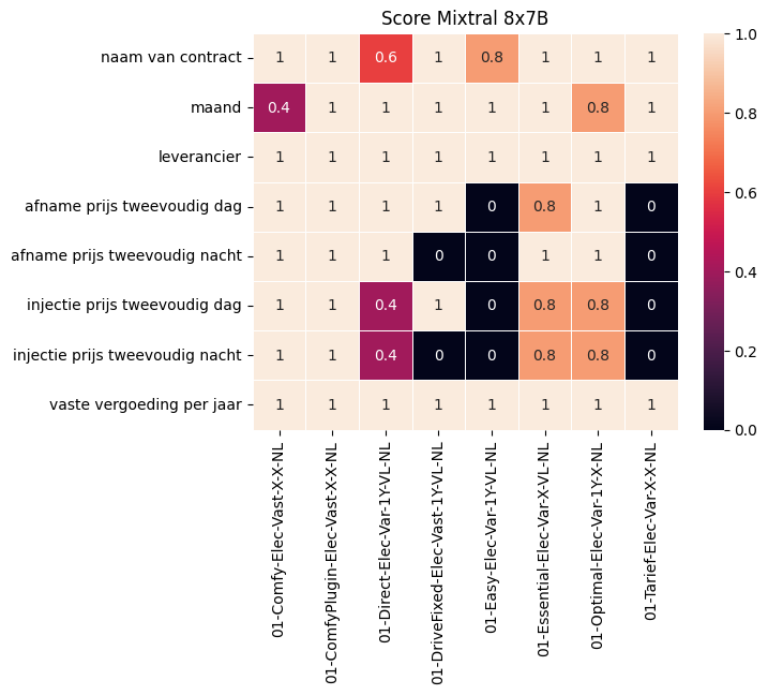


Figure 4.4: Output of Mixtral 8x7B model for the energy use case

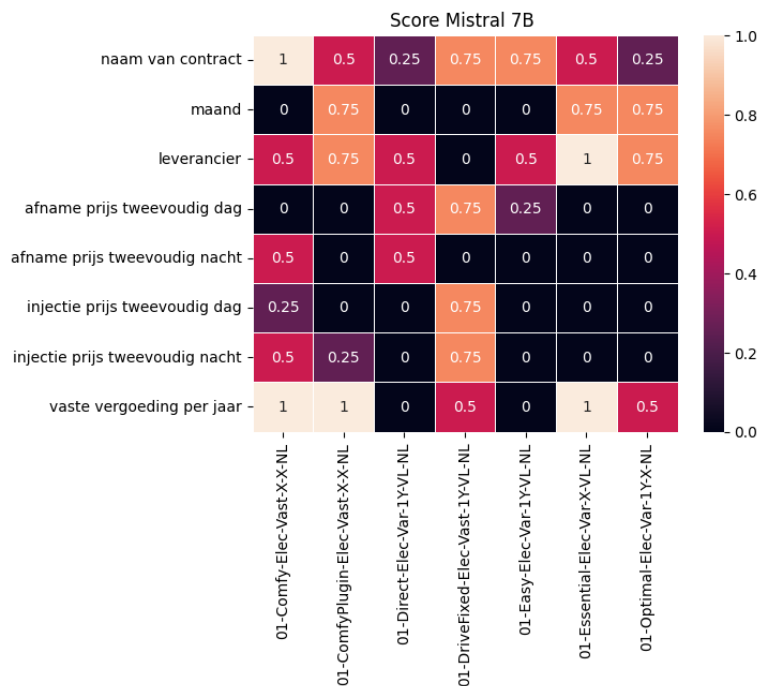


Figure 4.5: Output of Mistral 7B model for the energy use case

isolated assessment of the LLMs’ capabilities in understanding and processing textual content exclusively. The findings from these experiments reveal that the goldenset, generated via the proposed methodology, can successfully gauge the accuracy and consistency of LLM outputs provided.

In our comparative assessment of multiple large language models, the closed-source GPT-4-o from OpenAI exhibited the highest accuracy, registering a score of 87%. Each model was subjected to identical prompts and input documents to maintain objective evaluation conditions. Open-sourced LLAMA2 models universally scored below 50% accuracy, with the Mistral 7B model demonstrating the lowest performance at 34% accuracy. Conversely, the Mistral 8x7B model not only outperformed all other open-source models with a score of 79% but also surpassed the performance metrics of OpenAI’s closed-source GPT-3.5 turbo and GPT-4 models.

M7B	L7B	L13B	L70B	GPT3.5 MD	GPT3.5
34	41	45	48	61	63

**Table 4.4:** Score of LLMs tested (Part 1), score ranging from 0 to 100, 100 meaning that the model answers every question correctly.

M8x7B MD	GPT4	M8x7B	GPT4-o
69	77	79	87

**Table 4.5:** Score of LLMs tested (Part 2), score ranging from 0 to 100, 100 meaning that the model answers every question correctly.

Model Descriptions
L7B refers to LLAMA2 7B
L13B refers to LLAMA2 13B
L70B refers to LLAMA2 70B
M7B refers to Mistral 7B
M8x7B refers to Mixtral 8x7B
M8x7B MD refers to Mixtral 8x7B with input context converted to markdown
GPT3.5 refers to OpenAI’s GPT3.5 turbo
GPT3.5 MD refers to OpenAI’s GPT3.5 turbo with input context converted to markdown
GPT4 refers to OpenAI’s GPT3.5
GPT4-o refers to OpenAI’s GPT4 omni

**Table 4.6:** Full names and descriptions of the models tested.

During our testing and validation of LLM outputs, we encountered several issues. One major problem was that the LLM did not consistently provide output in JSON format, despite clear instructions given in the system prompt. This inconsistency made the validation process quite challenging. Another issue was that the LLM’s output did not fully match the expected answers in the goldenset. This discrepancy required us to iteratively update the validator in the LLM pipeline to correctly align the generated answers with the goldenset.

# Chapter 5

## Case Study - SEC-filling

The initial case study successfully showcases the potential of large language models (LLMs). However, the model may have performed well solely on this specific dataset, and its effectiveness might vary with other datasets. To validate the initial findings, we will test the LLMs on a different dataset. This will enable us to draw reliable conclusions and illustrate how well the LLM can adapt to different domains.

In this section, we follow the same structure as the previous one. First, we describe the new dataset used in this study. Next, we apply the techniques and methods discussed in Chapter 4 to this dataset. The main difference lies in the dataset's format in this case study, necessitating changes to the prompt template for querying the LLM. Despite this, the overall procedure remains the same.

### 5.1 Data Source

Finding publicly available datasets with PDFs and matching metadata is challenging. Metadata is crucial for handling large datasets and for building the goldenset. The goldenset for the initial case study was created manually by reviewing the PDFs. For this case study, we explored several open data sources which can be used as and second case study. Below, we discuss the data sources and the challenges they presented.

#### 5.1.1 Kruispuntbank van Ondernemingen

Every publicly traded company must report its financial performance to the government, providing transparency and information to investors, analysts, and regulators. In Belgium, this is managed by the KBO<sup>1</sup> (Kruispuntbank van Ondernemingen). The KBO, or Kruispuntbank van Ondernemingen, maintains detailed information about all registered businesses in Belgium, such as their legal form, address, economic activities, financial information, and more [3].

Data on companies can be retrieved via the KBO web interface. Nevertheless, a significant issue is encountered when selecting this dataset: the reuse of data obtained through KBO's public search feature is explicitly prohibited. The interface allows only narrowly-targeted, purpose-specific queries aimed at individual entities. [2]. An attempt was made to create an account to gain broader data access, but the application was subsequently not approved. As a result, we are unable to utilize this dataset.

---

<sup>1</sup><https://kbo.be>



### 5.1.2 SEC-filing

In our study, we analyze datasets from the U.S. SEC (Securities and Exchange Commission) filings, which encompass financial reports from major publicly traded entities such as Google, Microsoft, and Apple. These datasets are publicly accessible in both PDF and XML formats, making them suitable for our study. For the case study, we concentrate on processing Coca-Cola's quarterly SEC filings [1].

In order to utilize the data effectively, we undertake a series of preprocessing steps. Initially, we employ a web scraping script to download the data. Subsequently, we extract the relevant pages from the PDFs and convert the XML files into the required format by the LLM-pipeline. Finally, we refine our prompting strategy to handle SEC filings specifically.

#### Structure of the PDF

The SEC filing documents usually consist of about 50 to 60 pages. The initial pages contain the company's information, followed by many pages with financial data presented in table form.

In this case study, we focus on the first page (see Figure 5.1) and a page with financial data (see Figure 5.2). As in the initial case study, this will allow us to test two distinct data formats: 1) textual data embedded within paragraphs and 2) numerical data structured within tables.

**UNITED STATES  
SECURITIES AND EXCHANGE COMMISSION  
WASHINGTON, D.C. 20549  
FORM 10-Q**

☑ QUARTERLY REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934  
For the quarterly period ended June 27, 2014

OR

☐ TRANSITION REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934  
For the transition period from \_\_\_\_\_ to \_\_\_\_\_  
Commission File No. 001-02317

*The Coca-Cola Company*  
(Exact name of Registrant as specified in its Charter)

State or other jurisdiction of incorporation or organization: **GA-0020405**  
Other Coca-Cola Plans: **(1033 Employee Identification No.)**  
Atlanta, Georgia: **30313**  
(Address of principal executive offices) (Zip code)

Registrant's telephone number, including area code: **(404) 479-2121**

Indicate by check mark whether the Registrant (1) has filed all reports required to be filed by Section 13 or 15(d) of the Securities Exchange Act of 1934 during the preceding 12 months (or for such shorter period that the Registrant was required to file such reports), and (2) has been subject to such filing requirements for the past 90 days. Yes  No

Indicate by check mark whether the Registrant has submitted electronically and posted on its corporate Web site, if any, every Interactive Data File required to be submitted and posted pursuant to Rule 405 of Regulation S-T (17 CFR 232.405) of this chapter during the preceding 12 months (or for such shorter period that the Registrant was required to submit and post such files). Yes  No

Indicate by check mark whether the Registrant is a large accelerated filer, an accelerated filer, a non-accelerated filer, or a smaller reporting company. See the definitions of "large accelerated filer," "accelerated filer" and "smaller reporting company" in Rule 12b-2 of the Exchange Act.

Large accelerated filer  Accelerated filer   
Non-accelerated filer  Smaller reporting company   
(Do not check if a smaller reporting company)

Indicate by check mark if the Registrant is a shell company (as defined in Rule 12b-2 of the Exchange Act). Yes  No

Indicate the number of shares outstanding of each of the issuer's classes of common stock as of the latest practicable date:

Class of Common Stock	Outstanding as July 25, 2014
Not Applicable	4,385,023,884 Shares

**Figure 5.1:** First page used in the experiments, with textual data embedded within paragraphs

Part I. Financial Information

Item 1. Financial Statements (Unaudited)

**THE COCA-COLA COMPANY AND SUBSIDIARIES  
CONDENSED CONSOLIDATED STATEMENTS OF INCOME  
(UNAUDITED)**  
(In millions except per share data)

	Three Months Ended		Six Months Ended	
	June 27, 2014	June 28, 2013	June 27, 2014	June 28, 2013
<b>NET OPERATING REVENUES</b>	\$ 12,874	\$ 12,749	\$ 23,450	\$ 23,784
Cost of goods sold	4,819	4,909	8,982	9,211
<b>GROSS PROFIT</b>	7,705	7,709	14,148	14,473
Selling, general and administrative expenses	4,264	4,385	8,373	8,567
Other operating charges	281	182	339	293
<b>OPERATING INCOME</b>	3,170	3,243	5,446	5,653
Interest income	144	129	287	295
Interest expense	197	122	311	224
Equity income (loss) — net	284	246	328	333
Other income (loss) — net	(773)	29	(108)	(158)
<b>INCOME BEFORE INCOME TAXES</b>	3,384	3,525	5,680	5,899
Income taxes	779	831	1,288	1,406
<b>CONSOLIDATED NET INCOME</b>	2,605	2,694	4,392	4,493
Less: Net income attributable to noncontrolling interests	10	18	17	36
<b>NET INCOME ATTRIBUTABLE TO SHAREOWNERS OF THE COCA-COLA COMPANY</b>	\$ 2,595	\$ 2,676	\$ 4,374	\$ 4,457
<b>BASIC NET INCOME PER SHARE*</b>	\$ 0.59	\$ 0.60	\$ 0.96	\$ 0.99
<b>DILUTED NET INCOME PER SHARE*</b>	\$ 0.58	\$ 0.59	\$ 0.95	\$ 0.98
<b>DIVIDENDS PER SHARE</b>	\$ 0.305	\$ 0.289	\$ 0.610	\$ 0.566
<b>AVERAGE SHARES OUTSTANDING</b>	4,391	4,446	4,396	4,450
Effect of dilutive securities	65	81	68	76
<b>AVERAGE SHARES OUTSTANDING ASSUMING DILUTION</b>	4,456	4,527	4,464	4,526

\* Calculated based on net income attributable to shareholders of The Coca-Cola Company.  
Refer to Notes to Condensed Consolidated Financial Statements.

**Figure 5.2:** Second page used in the experiments, with numerical data structured within tables

In our analysis, we will be focusing on several key questions to gain a comprehensive understanding of the financial data. Key aspects of the SEC filing document that we are interested in include:

- **Simple in-text keys** Company name and Document Fiscal Year
- **Keys in tables:** Cost of Goods Sold, Gross Profit, Operating Income, Income Taxes, Interest Income, Interest Expense

Due to the nature of historical documents in terms of structure and contained information, we selected the standardized keys that are common across all documents. We subsequently downloaded the entire dataset available, spanning from 2013 to 2023.

## 5.2 Experiment

Since this case study extends our initial research, we focus on testing the top-performing closed-source and open-source models identified in the initial case study. Conducting this second experiment will also allow us to assess how easily the LLM-pipeline can be adapted to different datasets.

### Data Extraction

The Coca-Cola website allows downloading SEC filings as PDFs and Excel sheets containing metadata for a specific period. We will be using the quarterly reports, also known as 10-Q reports. To handle the approximately 31 PDFs and 31 Excel sheets, a Python script was developed to download this data. After downloading, the Excel sheets are analyzed and preprocessed to ensure the consistency of the metadata. From the metadata Excel sheets, only a subset of key-value pairs frequently appearing across all the downloaded PDF documents is used.

The PDFs are also preprocessed to be used by the LLM. The original PDFs are split, and only two pages, as shown in Figure 5.2 and Figure 5.2, are used as the input dataset.

### Goldenset creation

The goldenset for this use case is built automatically using the metadata from the Excel sheets. The keys detailed above in section 5.1.2 are extracted from this metadata and used as the goldenset. The answerset is built manually by examining the goldenset and defining the data types for the key-value pairs.

### Test runs configurations

In this case study, we configure 31 PDF files for performance testing of the LLM. Each document undergoes 10 independent test runs, where all keys are queried simultaneously by the LLM. The initial dataset comprises these 31 files, but any files the model fails to process through the LLM pipeline are excluded from the final analysis.

## 5.3 Inferencing the LLM

In the inference phase of the LLM-pipeline in this case study, we employ the closed-source GPT-4-O model. For the open-source counterpart, we utilize the Mixtral 8x7B model.

### Prompting

The prompts used for this case study are identical to the prompt template used in the first use case study (subsection 4.3.1), with a minor update to the domain in the system prompt to “SEC filing reports”. The few-shot prompting examples are also slightly updated. These were updated manually for the current case study, but can be automatically updated in the future using the goldenset and the input dataset to build the few-shot examples in the prompt.

The following text provides an updated instance of few-shot prompting:

---

```
User    {
        inputcontext = [Example input Context from SEC filling doc]
        Q: date, operating income
    }

Assistant {
        {"date": "Sep. 27, 2013", "operating income": "2472"}
    }
```

```

User {
    Above where example's use same approach to answer question below; Do not
    use the input context defined above; Use input context provided in the
    input context field below.

    inputcontext= [Input Context for the tested SEC filling doc]
    Q: [Set of keys that we want to extract]
}

```

## 5.4 Results and Analysis

The calculation of the score, which indicates how accurately the LLM answers the questions, is done using the same methodology as explained in section 4.4, with test parameters defined in section 5.2.

### 5.4.1 GPT4-o

The GPT-4o model achieves an accuracy rate of 99.30%. The only area where the GPT-4o model consistently struggles is in recognizing the “name of the company,” where it typically responds with “Not Found.” The results of this experiment are illustrated in Figure 5.3

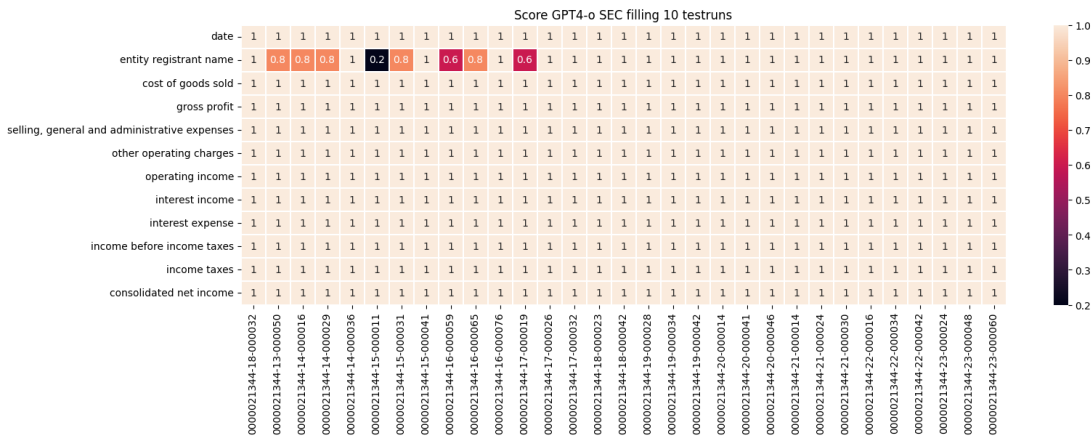


Figure 5.3: Results from Testing Entire SEC Document with GPT-4-o LLM

### 5.4.2 Mistral 7x8B

A notable issue with the open-source models during query processing was that the LLM frequently failed to retrieve the correct values from the intended column when queried with keys present in tables, as illustrated in Figure 5.4. Upon detailed analysis using the “HITL” (see chapter 6) of the raw output JSON, we discovered that the Mistral 7X8B model exhibited significant difficulties in accurately determining the “consolidated net income” across all documents. It is crucial to mention that we maintained a consistent prompt across both closed and open-source models to evaluate their generalization capabilities. The model demonstrated an overall accuracy of 93.36%.

To test the hypothesis that the LLM selects values from incorrect columns when answering financial data questions, we evaluated the smaller model, Mistral 7B, which showed the lowest accuracy in the initial case study. We kept the prompts and test configurations unchanged. This smaller model achieved an accuracy of 69.34%. A detailed analysis of the raw output JSON revealed that this model also sourced answers from incorrect columns. We used the “HITL” tool to gain insights into the LLM output. As shown in Figure 5.6, the model incorrectly selected

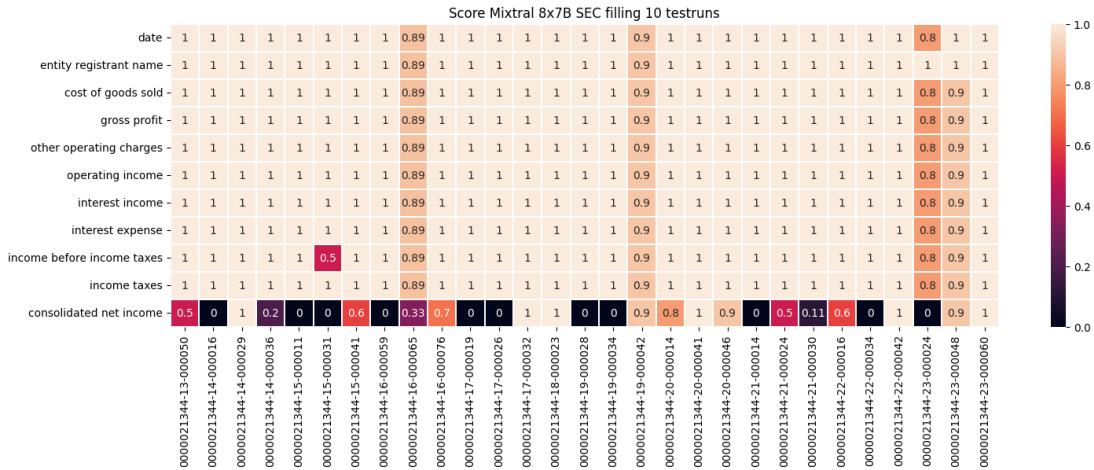


Figure 5.4: Results from Testing Entire SEC Document with Mixtral 7x8B LLM

the third and last columns instead of the first column. The highlighted text denotes the LLM’s output as responses to the questions. For example, the visualization indicates that for the key ‘Consolidated net income,’ the LLM responded with “7,196”, whereas the correct answer in the goldset is “2453”, found in the first column.

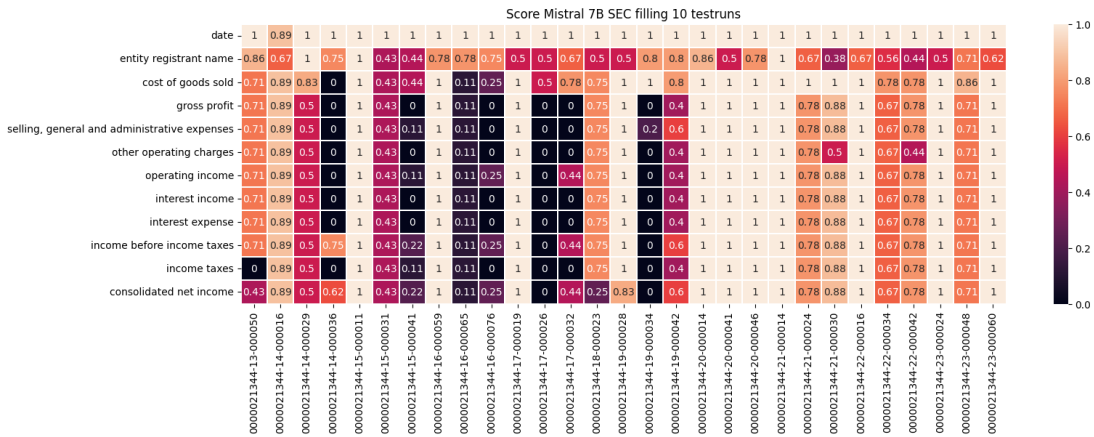


Figure 5.5: Results from Testing Entire SEC Document with Mistral 7B LLM

### 5.4.3 Conclusion

The GPT4-o model demonstrated the highest accuracy in answering questions, with minor issues related to the company name. This model effectively utilized few-shot prompting to automatically identify the correct column, resulting in the highest overall accuracy among the tested models.

In contrast, the Mistral models, particularly the smaller Mistral 7B, showed significant inaccuracies due to sourcing values from incorrect columns. This model did not learn from the examples which columns were being queried, leading to a decrease in overall accuracy. The larger Mistral 7x8B model also suffered from this issue but was primarily affected by the “Consolidated net income” key, where it consistently retrieved values from the wrong column. However, the overall score of the open-source Mistral 7x8B model was close to the GPT4-o model, with only a 5.94% difference.

THE COCA-COLA COMPANY AND SUBSIDIARIES CONDENSED CONSOLIDATED STATEMENTS OF INCOME (UNAUDITED) (In millions except per share data)					
	Three Months Ended		Nine Months Ended		
	September 27, 2012	September 28, 2012	September 27, 2012	September 28, 2012	
<b>NET OPERATING REVENUES</b>	\$ 12,030	\$ 12,340	\$ 35,814	\$ 36,562	
Cost of goods sold	4,793	4,853	14,106	14,425	
<b>GROSS PROFIT</b>	7,237	7,487	21,708	22,137	
Selling, general and administrative expenses	4,424	4,630	12,991	13,308	
Other operating charges	341	64	594	233	
<b>OPERATING INCOME</b>	2,472	2,793	8,123	8,596	
Interest income	136	118	381	345	
Interest expense	90	102	314	302	
Equity income (loss) — net	204	252	537	637	
Other income (loss) — net	658	23	522	156	
<b>INCOME BEFORE INCOME TAXES</b>	3,380	3,084	9,249	9,432	
Income taxes	925	755	2,331	2,236	
<b>CONSOLIDATED NET INCOME</b>	2,455	2,329	6,918	7,196	
Less: Net income attributable to noncontrolling interests	8	18	44	43	
<b>NET INCOME ATTRIBUTABLE TO SHAREOWNERS OF THE COCA-COLA COMPANY</b>	\$ 2,447	\$ 2,311	\$ 6,874	\$ 7,153	
<b>BASIC NET INCOME PER SHARE<sup>1</sup></b>	\$ 0.55	\$ 0.51	\$ 1.55	\$ 1.58	
<b>DILUTED NET INCOME PER SHARE<sup>1</sup></b>	\$ 0.54	\$ 0.50	\$ 1.52	\$ 1.56	
<b>DIVIDENDS PER SHARE</b>	\$ 0.28	\$ 0.255	\$ 0.84	\$ 0.765	
<b>AVERAGE SHARES OUTSTANDING</b>	4,426	4,502	4,442	4,513	
Effect of dilutive securities	72	85	76	80	
<b>AVERAGE SHARES OUTSTANDING ASSUMING DILUTION</b>	4,498	4,587	4,518	4,593	

<sup>1</sup> Calculated based on net income attributable to shareowners of The Coca-Cola Company.

**consolidated net income**

Correct Answers: 2,455

Precision over all test runs: 3/7

Wrong in these test run: [3, 4, 5, 6]

**Figure 5.6:** Visual interpretation of a test run on the 2012 SEC filing PDF. Screenshot captured from the developed “HITL” tool, incorporating the Mixtral 7B LLM output results.

Mistral 7B	Mixtral 7x8B	GPT4-o
69,34	93,36	99,30

**Table 5.1:** Performance Evaluation scores of all tested Large Language Models (LLMs), with scores ranging from 0 to 100. A score of 100 indicates that the Model answered every question correctly.

## Chapter 6

# Human in the Loop Tool (HITL)

A human-in-the-loop interface is designed to assist both developers and end users in understanding the output of the LLM for integration tasks. This interface provides a clear evaluation of the LLM’s performance in data extraction. In this section, the term “user” refers to both developers and end users who utilize the tools to support their work.

### Functionality

The tool is built with Streamlit<sup>1</sup> and uses the HTML PDF renderer to display the PDFs on the interface. It is a standalone tool and does not automatically run tests on VSC or use the OpenAI API. This means that the user must manually run the LLM-pipeline, and download the output JSON file provided by the LLM-pipeline. Once the user has the output JSON, it can be uploaded into the tool for analysis.

The interface features two primary pages. The first page enables users to compare the output of the LLM with the goldenset to assess its accuracy. The second page is intended for scenarios where the goldenset is unavailable; in this case, the tool creates an aggregated set to assist users in constructing the goldenset. It also offers a visual representation to evaluate the usability of the input dataset.

### Page 1 - Analyze Using Goldenset

Users can upload a PDF they want to analyze, along with the complete output from the LLM pipeline. The tool automatically extracts relevant data from the LLM output JSON file and visually maps this data onto the PDF. This feature allows users to immediately see how well the LLM has performed. When dealing with multiple key-value pairs, users can select specific keys through an input box, which dynamically adjusts the PDF display to highlight these filtered outputs. The layout of this page is shown in Figure 6.1.

The interface also provides a comparative analysis between the goldenset and the LLM output. Correctly identified keys (as determined by the goldenset) are both highlighted and underlined, depending on the user’s selection of test runs. Keys that are only highlighted indicate instances where the LLM did not produce the correct response. This annotation is especially useful for reviewing specific test runs (see Figure 6.2). Users can click on the highlighted text to access more details, such as the expected correct answer from the goldenset, the frequency of

---

<sup>1</sup><https://streamlit.io/>

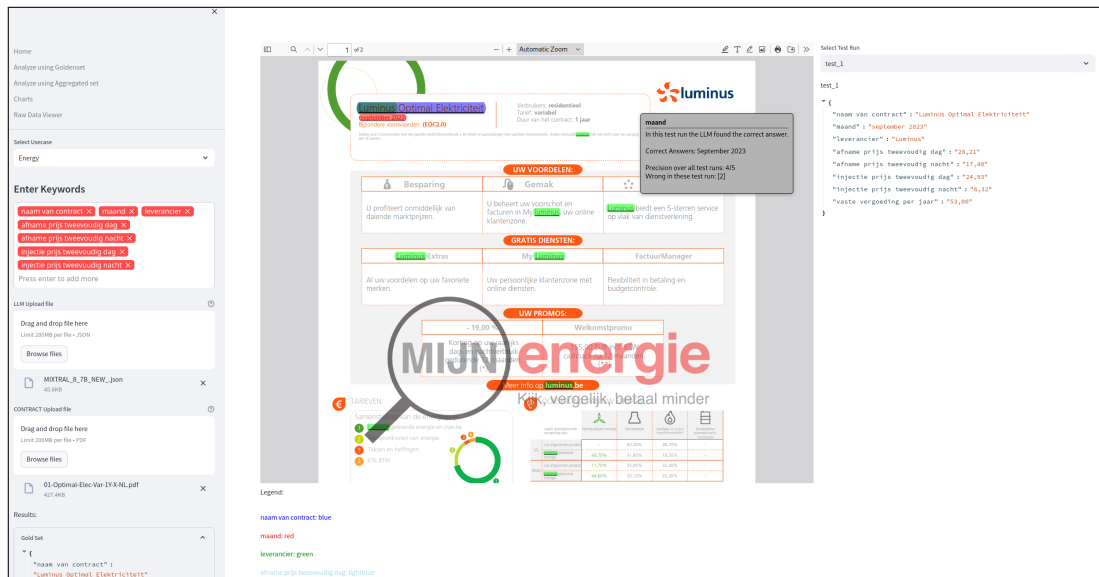


Figure 6.1: Interface of the first page

correct predictions over all test runs, and other test runs where the LLM failed to find the right answer.

Users can select a specific case study from the sidebar interface to analyze. The tool then automatically retrieves the corresponding goldenset and populates the predefined search keys associated with that case study. Additionally, a color-coding scheme for the entered keys enhances the clarity of the results visualization. This feature facilitates the rapid identification and interpretation of discrepancies in the LLM output.

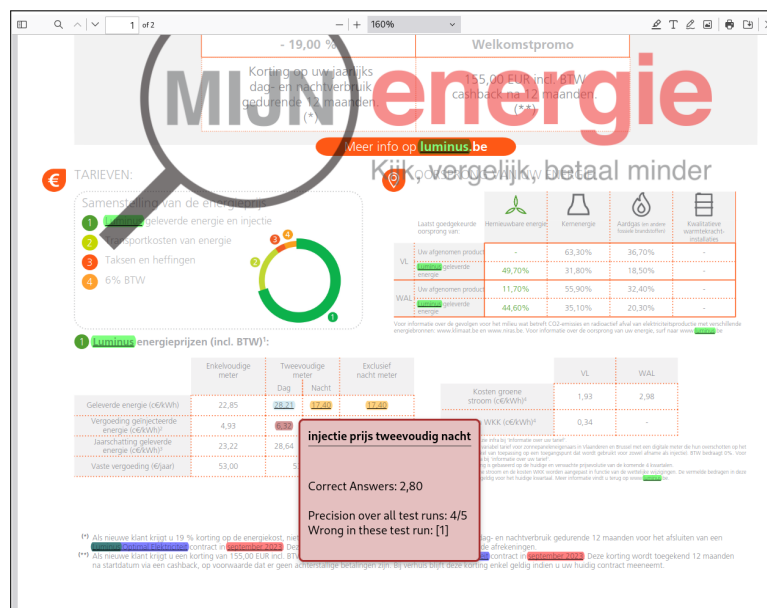


Figure 6.2: Illustration of PDF highlighting incorporating LLM output and subsequent annotations.

## Page 2 - Analyze Using Aggregated Set

The design of this page, as shown in Figure 6.3, closely resembles the first page. An additional sidebar displays how confident the LLM’s output is regarding the selected test runs, categorized by key.

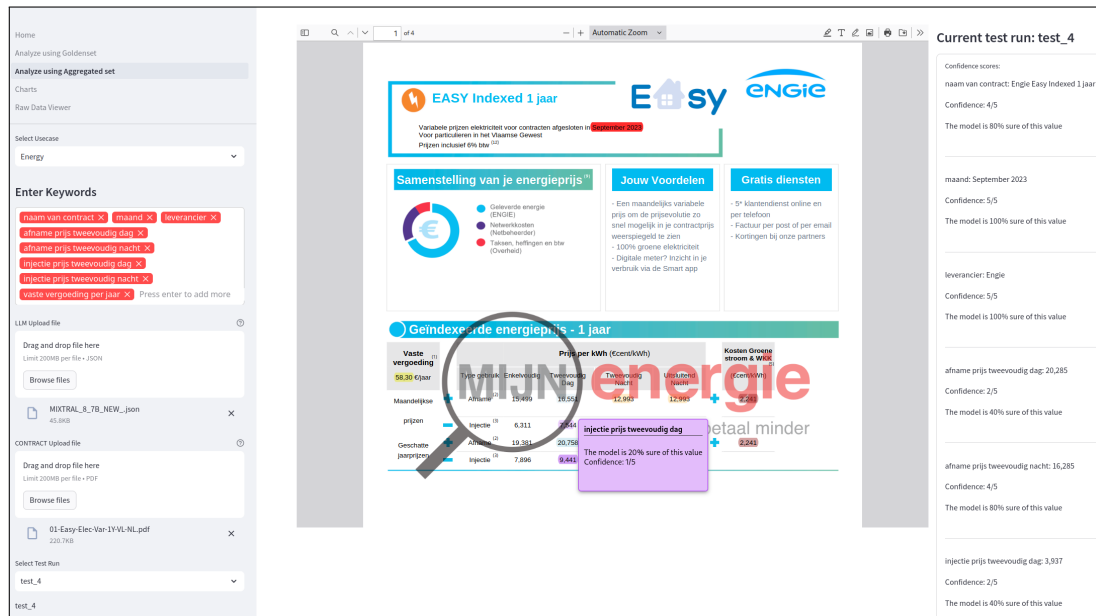


Figure 6.3: Interface of the second page

For datasets that lack metadata, the provided page offers an alternative approach. This approach retains the main features of the first page but removes the underline feature, as comparing the LLM output to a goldenSet is not possible. Such scenarios occur when either a goldenSet has not been developed by the engineer or when there is a need to evaluate a dataset’s viability for testing purposes.

This page enables users to analyze the LLM output without a GoldenSet, by comparing differences between test runs. The tool will create an aggregated GoldenSet through majority voting, where the most frequent value for each key across all test runs forms the aggregated set (see Figure 6.4).

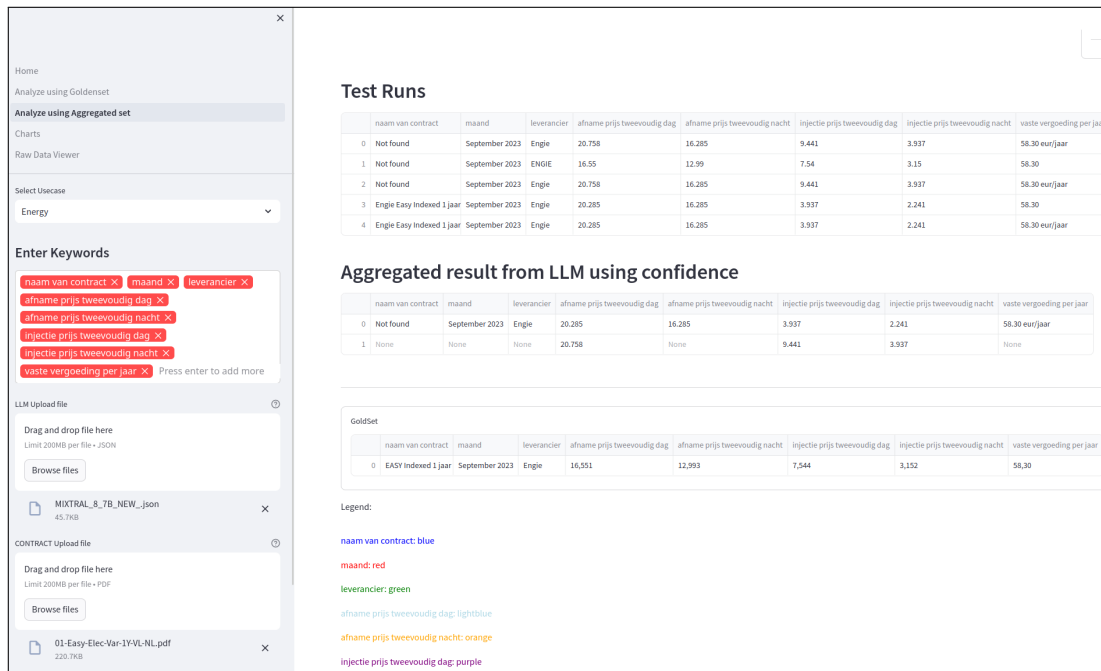
The aggregated set is not adopted as the definitive goldenSet in this analysis due to the likelihood of the LLM repetitively producing wrong values for a given key, thereby resulting in false key-value pairs. For instance, should the LLM output “Not found” three times and “EASY Indexed 1 jaar” only once (where “EASY Indexed 1 jaar” is the accurate value) for the “contract name”, the aggregated goldenSet would erroneously report “Not found” as the contract name. This situation is illustrated in Figure 6.4.

The user can also utilize this page to construct the GoldenSet by analyzing the aggregated data. The highlighting feature and the aggregated set can be used in the GoldenSet construction process. It is important to note that the aggregated output may contain errors. Therefore, this page should be used as an auxiliary tool to suggest potential correct answers, and developers should proceed with caution.

## Additional pages

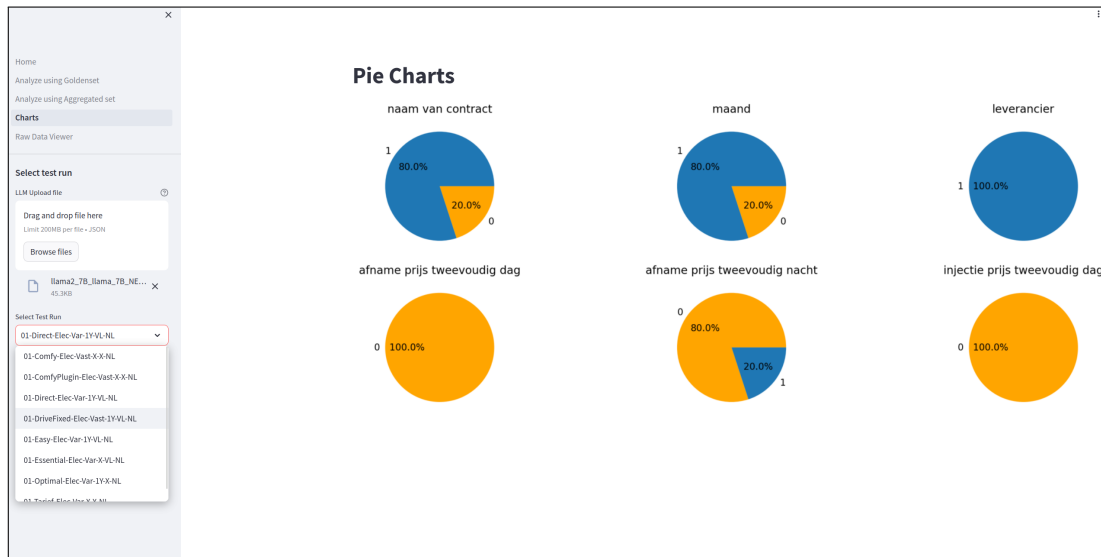
A chart page (see Figure 6.5) is also constructed to provide an overview of the LLM’s performance across all documents for each key. This visual representation highlights the overall consistency for each key. In these charts, the blue segment of the pie chart shows the percentage





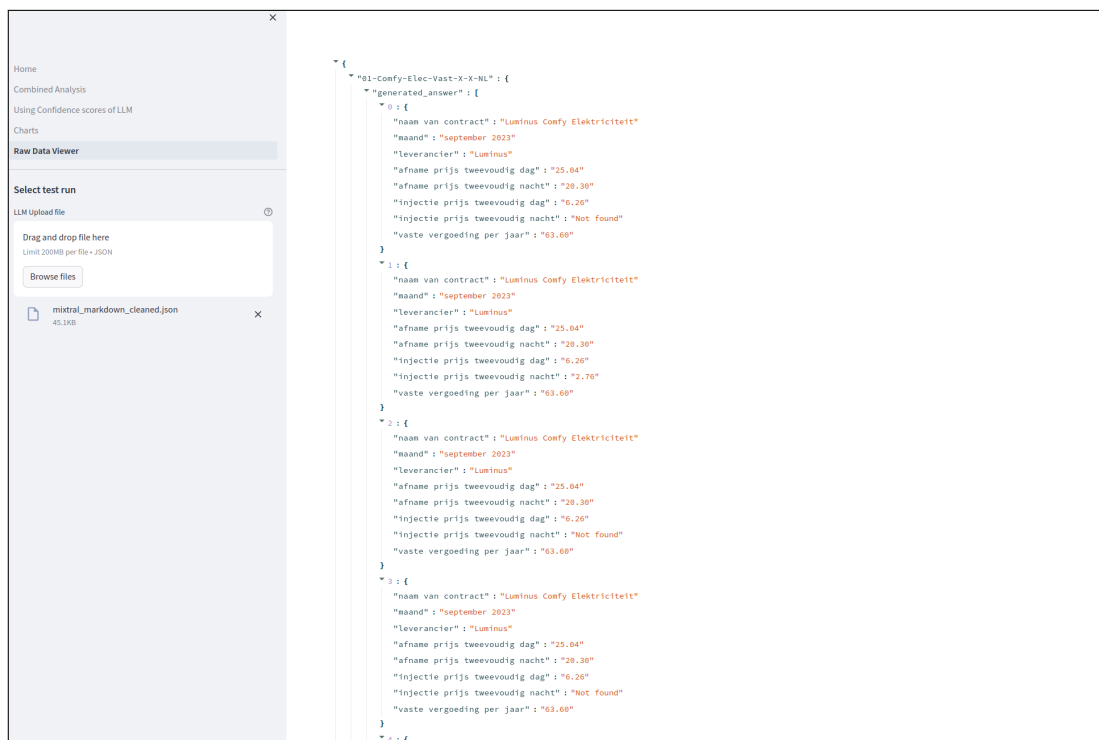
**Figure 6.4:** Illustration of the test runs of the selected LLM model, alongside the aggregated dataset constructed from all test runs and the golden set for the selected PDF document.

of keys correctly identified by the LLM, whereas the orange segment represents the instances of incorrect identification. These values are computed by comparing the goldenSet with the LLM output.



**Figure 6.5:** Overview of the LLM’s performance across all documents for each key

Additionally, another page is included for debugging purposes, allowing the developer to view the LLM output in JSON format (see Figure 6.6). This is particularly useful for identifying issues within the data, such as errors introduced by the post-validator or instances where the LLM generated plain text instead of JSON.



The screenshot shows a web interface for LLM analysis. On the left, there's a sidebar with navigation links: Home, Combined Analysis, Using Confidence scores of LLM, Charts, and Raw Data Viewer. Below this is a 'Select test run' section with an 'LLM Upload file' area. A file named 'mitral\_markdown\_cleaned.json' (45.1KB) is listed. On the right, a JSON output window displays the following data:

```

{
  "01-Comfy-Elec-Vast-X-X-NL": {
    "generated_answer": [
      0: {
        "naam van contract": "Luminus Confy Elektriciteit"
        "maand": "september 2023"
        "leverancier": "Luminus"
        "afname prijs tweevoudig dag": "25.04"
        "afname prijs tweevoudig nacht": "20.30"
        "injectie prijs tweevoudig dag": "6.26"
        "injectie prijs tweevoudig nacht": "Not found"
        "vaste vergoeding per jaar": "63.60"
      }
      1: {
        "naam van contract": "Luminus Confy Elektriciteit"
        "maand": "september 2023"
        "leverancier": "Luminus"
        "afname prijs tweevoudig dag": "25.04"
        "afname prijs tweevoudig nacht": "20.30"
        "injectie prijs tweevoudig dag": "6.26"
        "injectie prijs tweevoudig nacht": "2.76"
        "vaste vergoeding per jaar": "63.60"
      }
      2: {
        "naam van contract": "Luminus Confy Elektriciteit"
        "maand": "september 2023"
        "leverancier": "Luminus"
        "afname prijs tweevoudig dag": "25.04"
        "afname prijs tweevoudig nacht": "20.30"
        "injectie prijs tweevoudig dag": "6.26"
        "injectie prijs tweevoudig nacht": "Not found"
        "vaste vergoeding per jaar": "63.60"
      }
      3: {
        "naam van contract": "Luminus Confy Elektriciteit"
        "maand": "september 2023"
        "leverancier": "Luminus"
        "afname prijs tweevoudig dag": "25.04"
        "afname prijs tweevoudig nacht": "20.30"
        "injectie prijs tweevoudig dag": "6.26"
        "injectie prijs tweevoudig nacht": "Not found"
        "vaste vergoeding per jaar": "63.60"
      }
    ]
  }
}

```

Figure 6.6: LLM output from the LLM-pipeline in JSON output for raw data analysis

# Chapter 7

## Discussion

In this section, we will analyze the results and conclusions of the case studies discussed in chapter 4 and chapter 5.

### Patterns in LLM output

By examining the outputs of both case studies, we observed certain patterns. In the conclusion of each case study, we interpreted the results individually. Now, we will identify the recurring patterns across the case studies to better understand the effectiveness of LLMs in data integration tasks. In both case studies, we analyzed two types of data: in-text data and tabular data.

#### In-text data

In our comparative analysis of key-value pairs, it was observed that Large Language Models (LLMs) demonstrate superior accuracy in extracting and interpreting values embedded within paragraphs as opposed to those formatted within tables. The comprehensive accuracy metrics underpinning this observation are detailed in the results section of the case studies.

#### Data in tables

A clear pattern observed from the first case study is that the tested LLMs struggle with extracting data from tables, showing the lowest overall accuracy on these key-value pairs. We examined the raw PDF document and found that the energy contracts contain complex table structures. Converting this semantic PDF to raw text fails to retain these table structures, leading to incorrect answers from the LLMs. Notably, the LLMs do not hallucinate, meaning they do not generate answers not present in the input context. Using the “HITL” visualization tool, we noticed that the LLM misinterprets the data in the tables, often extracting answers from adjacent columns.

A similar pattern appeared in the second case study with SEC filing reports. Initially, we tested the best open and closed-source models and found that the GPT-4-o model performed the best, with the Mistral 7x8B model also performing well. These models did not exhibit major problems with tables, likely due to the simpler underlying structure of the SEC financial sections, which remains intact when converted to text format. However, smaller models, like the llama 7B and de mistral 7B, performed poorly again, often extracting answers from the wrong columns.

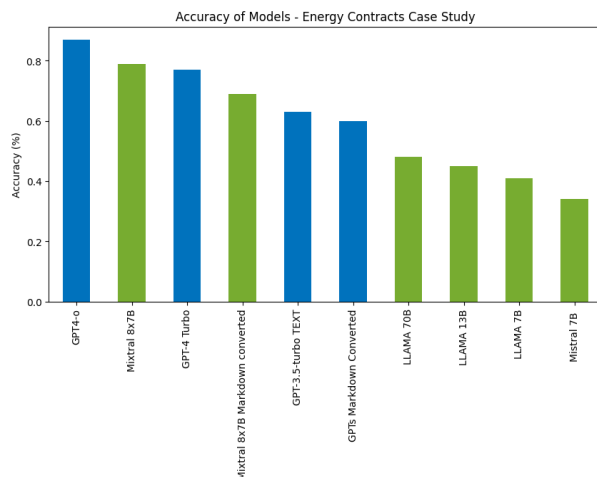
Our experiments revealed that high-accuracy (GPT4-o and Mixtral) models can infer the appropriate column for extraction from provided few-shot examples. We intentionally did not specify which column contained the correct answers to evaluate the LLMs’ learning capabilities. The

top-performing models successfully identified this pattern, whereas the smaller models did not. The overall accuracy and consistency of these models across the datasets in both case studies clearly address the first and second research questions.

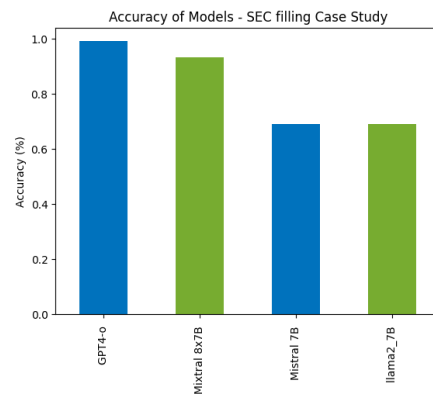
To improve accuracy, we experimented with converting PDFs to text and then to markdown, as markdown files are more structured than raw text files. Initial tests showed increased accuracy compared to raw text prompting. However, large-scale tests revealed decreased performance compared to normal LLM prompting.

## Open- and close-source models

To answer the third research question, we compared all the models tested across the two case studies. This evaluation helps determine which open-source models are viable alternatives to the closed-source models. Additionally, the model accuracy provides insights into the second research question regarding the key-value accuracy of the tested models.



**Figure 7.1:** Accuracy of Open-Source and Closed-Source Models on the Energy Contracts Case Study



**Figure 7.2:** Accuracy of Open-Source and Closed-Source Models on the SEC filling case study

Figure 7.1 and Figure 7.2 provide an overview of the accuracy of the tested LLMs. The charts are color-coded: blue represents closed-source models, while green denotes open-source models. From the accuracy metrics, it is evident that GPT-4-o outperforms the other tested models. The Llama2 models we tested performed poorly in comparison, showing less than 60% accuracy overall. Notably, the Mixtral 7x8B models demonstrated superior performance compared to GPT-4 and GPT-3.5 turbo, despite using fewer parameters and less VRAM than larger models like LLama2 70B. This suggests that Mixtral model could serve as viable open-source alternatives to closed-source GPT-4 model.

During our testing, we also observed a pattern where smaller models performed poorly, as explained in the previous section. A small test was done on a smaller model (Llama2 7B) where we utilized them for simple tasks with more specific prompting rather than generic prompting templates, this aims to improve their accuracy. The more general a prompt is, the worse smaller models generate the correct answer. We also found that the parameter count alone does not determine how powerful an LLM is. For instance, we compare the LLama2 70B parameter model with the Mixtral 7x8B model, which has only 56B parameters. Surprisingly, the Mixtral model not only performs comparably to the closed-source model from OpenAI but also outperforms the LLama2 70B model.

## Cost analysis

For the open-source models, we used the VSC, which provides computing credits to UHasselt for academic research. These credits were allocated for use in this thesis.

For the closed-source models, OpenAI charges based on the number of tokens processed. The cost per test run of the GPT models can be analyzed by examining token usage. The LLM pipeline has a built-in logging system that logs each call to OpenAI’s API, which can be used to calculate the mean cost of all the test runs. The results are provided in Table 7.1.

**Table 7.1:** Cost Comparison of GPT3.5 and GPT4 Models

GPT Model	Cost per Test Run (\$)	Cost Factor
GPT3.5-turbo	~0.001	~1
GPT4-o	~0.015	~15
GPT4	~0.03	~30

For a single test run with the GPT-3.5 model, the cost incurred for processing the provided prompt was approximately \$0.001. In contrast, using the GPT-4 model for the same prompt resulted in a significantly higher cost of approximately \$0.03 for an equivalent number of tokens. This represents a substantial cost difference, with the GPT-4 model being approximately 30 times more expensive for the same text generation task.

Our tests showed that the GPT-4-o model performs the best and is better than the GPT-4 model. The GPT-4-o model costs approximately \$0.015, meaning it has better accuracy at half the cost of the GPT-4 model.

The cost disparity between the GPT4-o and GPT4 model underscores the considerable difference in financial implications, highlighting the importance of carefully considering the cost-effectiveness of utilizing newer versions of AI models for text generation tasks.

## Issues

Several technical issues arose while building the LLM pipeline to systematically and automatically run tests on the open-source models. The validator module of the developed LLM pipeline requires that the input fed into the validator is valid JSON. It is essential that the LLMs provide the generated answers in valid JSON format. We experienced many issues getting this part to work because LLMs have non-deterministic output each time we ran the test. For example, the validator needs dates in a certain format, but the LLMs often failed to provide answers in the required format, necessitating updates to the validator.

One significant challenge involves the JSON output of low-parameter LLMs, which frequently contains errors. This happens despite enforcing JSON output through system prompt templates. To handle this, we revise our validation pipeline to include post-processing steps to ensure the output format is valid. If the validation mechanism fails, the validator marks the generated answer as invalid. By considering this factor in LLM output analysis, we can correlate high accuracy with the ability to produce valid JSON files.

Another substantial issue is validating the data generated by the LLM. In both case studies, the generated data often deviates from the expected format. This problem forces us to augment the validation process with rule-based matching and data cleaning. While the answer set alleviates this issue to some extent, the LLM pipeline still requires review when processing new datasets. We address this by developing extensive unit tests to ensure the pipeline’s robustness for comprehensive testing.

## Chapter 8

# Conclusion

In this conclusion, we summarize the key findings of this thesis and reflect on possible improvements. We will also discuss the limitations of this work and suggest areas for future enhancement.

The primary objective of this thesis was to explore the efficiency of Large Language Models (LLMs) in data integration tasks. The research questions formulated at the beginning guided us throughout the thesis. We tested both the LLM pipeline and the case studies with these research questions in mind. Our findings addressed all the research questions using the results of the two case studies, “HITL” tool and the developed LLM pipeline.

The first research question investigated whether LLMs could consistently extract text. Our case studies showed that OpenAI’s GPT4-o and the Mixtral 7x8B model exhibited the highest consistency. Due to their high consistency, these models also achieved the highest key-value pair accuracy, thereby addressing the first and second research questions. Among the tested models, GPT4-o model performed the best overall. Comparing open and closed-source models, only the Mixtral 7x8B open-source model performed comparably to OpenAI’s GPT-4 models, while Llama2 models scored below 60%. To address the third research question, models such as Mixtral 7x8B were identified as viable alternatives to OpenAI’s proprietary models.

The Human-In-The-Loop (HITL) approach was successfully utilized during debugging and testing phases, as well as for visualizing the LLM output. This tool allowed us to easily identify patterns, such as smaller parameter models struggling with key-value pairs in tables. Before developing this interface, identifying such patterns in raw LLM output was challenging. Using the visualization tool, we found that smaller models could be enhanced with more specific prompts and used for simpler extractions. This successfully answers the final research question about the usefulness of a human intervention tool in interpreting LLM outputs.

Despite gathering useful insights, we faced numerous challenges throughout the end-to-end process of this thesis. The main objective was to evaluate the efficacy of Large Language Models (LLMs) and to apply the findings from an initial successful case study to a more sensitive dataset from Jessa Hospital.<sup>1</sup> The ultimate goal was to determine if open-source models could be viable in the medical sector. Due to the inherent limitations and security concerns associated with open-source large language models (LLMs), we select the Vlaams Supercomputer Centrum (VSC) platform for conducting our testing procedures. Their platform facilitates the secure processing of sensitive medical data acquired from Jessa Hospital by ensuring data remain isolated. However, we encounter a significant challenge as the anticipated sensitive medical data from Jessa Hospital fail to arrive, compelling us to seek an alternative dataset late in the development phase.

---

<sup>1</sup><https://www.jessazh.be/>

Identifying a suitable dataset for the second case study was time-consuming, as we required a specific format: a set of PDF documents accompanied by metadata that could serve as a goldenset. After an extensive search, we selected the SEC filing dataset, which met our requirements. However, the conclusions drawn from the first case study did not fully align with those from the SEC filing dataset. While the first case study demonstrated that LLMs struggle with data in tables, the best-performing LLM achieved over 90% accuracy on data in tables in the second case study. This discrepancy is likely due to the simpler nature of the tables in the second case study compared to the first case study.

Another challenge we faced was the time-consuming nature of developing the LLM pipeline, as detailed in the Discussion chapter. Due to changes in the LLM pipeline code, we had to rewrite the Human-In-The-Loop (HITL) tool to match the output of the LLM pipeline. This resulted in a continuous loop of updating both codebases, causing significant trouble and delays.

Initially, we also underestimated the complexity of the LLM pipeline and used a simple codebase to test the initial hypothesis. However, as we started testing multiple LLMs, the codebase became very unstable. This led to a complete refactoring of the codebase, which we redeveloped from the ground up in an object-oriented manner. This approach allows the LLM pipeline to be easily updated in the future to test other LLMs.

## Limitations and Future Directions

This study identifies several limitations that require further exploration in future research. The first limitation is that the case studies tested primarily involved only two different structures of PDF files. Although the results obtained in this study are promising, they do not fully represent the potential of the tested LLMs. Moreover, the dataset used in the study was relatively small. Testing the LLMs on two different datasets in these case studies has shown some overlap. However, in the future, this study should be extended to larger datasets. The developed methods and techniques can be applied to new LLMs as they are released to test their performance.

Another significant downside of the developed LLM pipeline and the Human-In-The-Loop (HITL) approach is that they work only with batch processing. This means that users can only request a dataset to be preprocessed by the LLM pipeline, requiring them to specify the key-value pairs they need in advance. Additionally, the “HITL” does not support dynamic interaction with the LLM. Future work should focus on directly connecting the “HITL” to the LLM inferencing. For closed-source models, this is not a difficult task as they work with APIs. However, the way open-source models are run on the VSC limits them to batch processing only, as the VSC does not have an interactive node for the GPUs that are used for inferencing.

One aspect not covered in this thesis is the fine-tuning of models. Smaller models that perform worse can be fine-tuned with more domain knowledge and specific prompting to enhance their performance in future work. The developed LLM pipeline can be used to test these fine-tuned models to see how they perform on domain-specific tasks.

The last limitation is related to the context window of the LLMs. In the case studies tested in this thesis, we manually reviewed the dataset and selected only the relevant pages of PDFs that would fit into the context window of the LLMs being tested. This process can be very time-consuming, especially for large datasets with diverse types of PDFs. A potential solution is the use of Retrieval-Augmented Generation (RAG) systems. This approach would allow the LLM pipeline to automatically extract only the relevant input context needed for the current prompt sent to the LLM. Although we tried this approach before manually extracting the relevant data, it added extra dimensions of factors that could influence the LLM’s performance by providing incorrect input context. To keep the testing more deterministic, we chose not to implement this in the LLM pipeline and instead used pre-extracted PDFs that were consistent across all LLM tests.

In conclusion, this research provides significant insights into the usability and accuracy of LLMs,

along with their inherent limitations. The developed technique demonstrates that LLMs can be a viable option for the extraction phase in data integration tasks. Additionally, this research illustrates that the developed LLM pipeline and “HITL” framework can facilitate future work of testing new LLMs and simplifying the selection process of appropriate models.



# Bibliography

- [1] All sec filings. <https://investors.coca-colacompany.com/filings-reports/all-sec-filings>. Coca-Cola Company, [Online; accessed 3. Jun. 2024].
- [2] Cbe public search. <https://economie.fgov.be/en/themes/enterprises/crossroads-bank-enterprises/services-everyone/consultation-and-research-data/cbe-public-search>. FPS Economy, [Online; accessed 14. Apr. 2024].
- [3] European e-justice portal - business registers in eu countries. [https://e-justice.europa.eu/106/EN/business\\_registers\\_in\\_eu\\_countries?BELGIUM&member=1](https://e-justice.europa.eu/106/EN/business_registers_in_eu_countries?BELGIUM&member=1). [Online; accessed 14. Apr. 2024].
- [4] Genius hardware. [https://docs.vscentrum.be/leuven/tier2\\_hardware/genius\\_hardware.html](https://docs.vscentrum.be/leuven/tier2_hardware/genius_hardware.html). [Online; accessed 11. May. 2024].
- [5] How generative ai can boost highly skilled workers' productivity | mit sloan. <https://mitsloan.mit.edu/ideas-made-to-matter/how-generative-ai-can-boost-highly-skilled-workers-productivity>. Online; accessed 12. Jun. 2024.
- [6] Mijnergie.be. <https://www.mijnergie.be>. [Online; accessed 4. May. 2024].
- [7] Mijnergie.be - energieleveranciers. <https://www.mijnergie.be/energieleveranciers/>. [Online; accessed 4. May. 2024].
- [8] Mijnergie.be - energieleveranciers vergelijken. <https://www.mijnergie.be/energieleveranciers-vergelijken->. [Online; accessed 4. May. 2024].
- [9] Neural network (machine learning). [https://en.wikipedia.org/wiki/Neural\\_network\\_%28machine\\_learning%29](https://en.wikipedia.org/wiki/Neural_network_%28machine_learning%29). [Online; accessed 13. Apr. 2024].
- [10] Neural networks. [https://link.springer.com/referenceworkentry/10.1007/978-3-642-20617-7\\_6563](https://link.springer.com/referenceworkentry/10.1007/978-3-642-20617-7_6563). [Online; accessed 13. Apr. 2024].
- [11] Neural networks. <https://www.interaction-design.org/literature/topics/neural-networks>. [Online; accessed 13. Apr. 2024].
- [12] Openai platform. <https://platform.openai.com/docs/models>. [Online; accessed 23. May. 2024].
- [13] Prompting techniques | prompt engineering guide. <https://www.promptingguide.ai/techniques>. [Online; accessed 30. May. 2024].
- [14] Supercomputing in europe. [https://en.wikipedia.org/wiki/Supercomputing\\_in\\_Europe](https://en.wikipedia.org/wiki/Supercomputing_in_Europe). [Online; accessed 11. May. 2024].
- [15] V-test resultaat. <https://vtest.vreg.be/vtest/resultaat/117bfaf3-f80a-4f63-8ebf-6010eb213cae>. [Online; accessed 4. May. 2024].

- [16] Vlaams supercomputer centrum - compute. <https://docs.vscentrum.be/compute.html>. [Online; accessed 11. May. 2024].
- [17] Vlaams supercomputer centrum - high-performance computing (hpc). <https://www.vscentrum.be/hpc>. [Online; accessed 11. May. 2024].
- [18] Vlaams supercomputer centrum - tiering. <https://www.vscentrum.be/tiering>. [Online; accessed 11. May. 2024].
- [19] Vlaams supercomputer centrum - vsc account. [https://docs.vscentrum.be/access/vsc\\_account.html](https://docs.vscentrum.be/access/vsc_account.html). [Online; accessed 11. May. 2024].
- [20] Vlaams supercomputer centrum - what are standard terms used in hpc? [https://docs.vscentrum.be/what\\_are\\_standard\\_terms\\_used\\_in\\_hpc.html](https://docs.vscentrum.be/what_are_standard_terms_used_in_hpc.html). [Online; accessed 11. May. 2024].
- [21] Vlaamse regulator van de elektriciteits- en gasmarkt (vreg). <https://www.vreg.be/nl>. [Online; accessed 4. May. 2024].
- [22] What are ai hallucinations? <https://www.ibm.com/topics/ai-hallucinations>. IBM, [Online; accessed 13. Jun. 2024].
- [23] What are large language models (llms)? | ibm. <https://www.ibm.com/topics/large-language-models>. [Online; accessed 1. Jun. 2024].
- [24] What is a neural network? <https://www.spotfire.com/glossary/what-is-a-neural-network>. [Online; accessed 13. Apr. 2024].
- [25] What is data integration? | ibm. <https://www.ibm.com/topics/data-integration>. [Online; accessed 12. Jun. 2024].
- [26] Wice hardware. [https://docs.vscentrum.be/leuven/tier2\\_hardware/wice\\_hardware.html](https://docs.vscentrum.be/leuven/tier2_hardware/wice_hardware.html). [Online; accessed 11. May. 2024].
- [27] Natural Language Processing (NLP) - A Complete Guide. <https://www.deeplearning.ai/resources/natural-language-processing>, January 2023. [Online; accessed 15. Jun. 2024].
- [28] The Essential Guide to Neural Network Architectures. <https://www.v7labs.com/blog/neural-network-architectures-guide>, June 2024. [Online; accessed 15. Jun. 2024].
- [29] What Is NLP (Natural Language Processing)? | IBM. <https://www.ibm.com/topics/natural-language-processing>, June 2024. [Online; accessed 15. Jun. 2024].
- [30] 3Blue1Brown. But what is a neural network? | Chapter 1, Deep learning. <https://www.youtube.com/watch?v=aircAruvnKk>, October 2017. [Online; accessed 15. Jun. 2024].
- [31] Mistral A. I. Mixtral of experts. <https://mistral.ai/news/mixtral-of-experts>. [Online; accessed 13. Jun. 2024].
- [32] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. Learning representations by back-propagating errors: Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 1:77, 1985.
- [33] Quentin Anthony, Stella Biderman, and Hailey Schoelkopf. Transformer math 101. *EleutherAI Blog*, mar 2024. [Online; accessed 30. May. 2024].
- [34] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avani Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes, 2023.
- [35] Ashwin Belle, Raghuram Thiagarajan, S. M. Reza Soroushmehr, Fatemeh Navidi, Daniel A. Beard, and Kayvan Najarian. Big data analytics in healthcare. *Biomed Res. Int.*, 2015, 2015.

- [36] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- [37] Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. Generative ai at work, 2023.
- [38] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2018.
- [39] Lorena Carlo, Herbert S. Chase, and Chunhua Weng. Aligning structured and unstructured medical problems using umls. *AMIA Annu. Symp. Proc.*, 2010:91–95, nov 2010.
- [40] AWS Community. What is an instruct model? <https://community.aws/content/2ZVa61RxToXUFzcuY8Hbut6L150/what-is-an-instruct-model?lang=en>. [Online; accessed 26 May. 2024].
- [41] Contributors to Wikimedia projects. Generative pre-trained transformer - wikipedia. [https://en.wikipedia.org/w/index.php?title=Generative\\_pre\\_trained\\_transformer&oldid=1223768820](https://en.wikipedia.org/w/index.php?title=Generative_pre_trained_transformer&oldid=1223768820). [Online; accessed 23. May. 2024].
- [42] Microsoft Developer. State of gpt | brk216hfs. <https://www.youtube.com/watch?v=bZQun8Y4L2A>. YouTube, [Online; accessed 3. Jun. 2024].
- [43] Data Science Dojo. Open source llms vs closed source llms. <https://datasciencedojo.com/blog/open-source-llms-vs-close-source-llms/>. [Online; accessed 11. May. 2024].
- [44] Eurostat. Electricity and gas prices stabilise in 2023. *Eurostat*, oct 2023.
- [45] Hugging Face. Hugging face model repository. <https://huggingface.co/models>. [Online; accessed 11. May. 2024].
- [46] Amirreza Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International journal of information management*, 35(2):137–144, 2015.
- [47] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
- [48] Geoffrey E Hinton and Terrence J Sejnowski. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [49] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, and Hartwig ... Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, 2017.
- [50] Mbali Kalirane. Gradient descent vs. backpropagation: What’s the difference? *Analytics Vidhya*, April 2023.
- [51] Madawa Manathunga, Hasan Aktulga, Andreas Götz, and Kenneth Merz. Quantum mechanics/molecular mechanics simulations on nvidia and amd graphics processing units. *Journal of Chemical Information and Modeling*, 2023.
- [52] Warren McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 1943.
- [53] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [54] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot, 2023.

- [55] Associated Press. Wrong responses of google's new ai tool leave experts worried. <https://www.dailysabah.com/business/tech/wrong-responses-of-googles-new-ai-tool-leave-experts-worried>. [Online; accessed 08. Jun. 2024].
- [56] Foster Provost and Tom Fawcett. Data science and its relationship to big data and data-driven decision making. *Big data*, 1(1):51–59, 2013.
- [57] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *arXiv*, jan 2023.
- [58] ScrapingBee. The challenges of web scraping unstructured data. <https://example.com>. [Online; accessed 4. May. 2024].
- [59] Uthayasankar Sivarajah, Mustafa M Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70:263–286, 2017.
- [60] SpringsApps. Open source vs. closed source llms: Pros and cons. <https://springsapps.com/knowledge/open-source-vs-closed-source-llms-pros-and-cons>. [Online; accessed 11. May. 2024].
- [61] DS Stream. Open source vs. closed source in language models: Pros and cons. <https://dsstream.com/open-source-vs-closed-source-in-language-models-pros-and-cons/>. [Online; accessed 11. May. 2024].
- [62] Kartik Talamadupula. A Guide to Quantization in LLMs | Symbl.ai. *Symbl*, February 2024. [Online; accessed 15. Jun. 2024].
- [63] Ibm Technology. Neural networks explained in 5 minutes. <https://www.youtube.com/watch?v=jmmW0F0biz0>. [Online; accessed 29. May. 2024].
- [64] Ibm Technology. What is NLP (Natural Language Processing)? <https://www.youtube.com/watch?v=fLvJ8VdHLA0>, August 2021. [Online; accessed 15. Jun. 2024].
- [65] Catherine Thorbecke. Ai tools make things up a lot, and that's a huge problem. *CNN*, aug 2023.
- [66] Gopinath Velivela, Krishna Rao, Yallamanda Challa, and Kasaraneni Purna Prakash. The journey of big data: 3 v's to 3 2 v's. mar 2016.
- [67] Dongdong Zhang, Changchang Yin, Jucheng Zeng, Xiaohui Yuan, and Ping Zhang. Combining structured and unstructured data for predictive models: a deep learning approach. *BMC Med. Inf. Decis. Making*, 20, 2020.

# Chapter 9

## Appendix A

### 9.1 Prompt templates

Throughout our testing phase, we evaluated multiple variations of the prompt. The optimal prompt, which demonstrated superior performance across all tested LLMs, is presented below.

#### 9.1.1 SEC filing prompt

---

```
System {
    The agent's objective is to recognize questions within the provided
    context and answer them. In cases where the agent lacks information
    for a specific key,
    it is required to respond with 'Not found.' Agent must respond in a
    valid JSON format. Let's proceed step by step.

    response must be in valid desired JSON format, and without any
    additional information.

    Desired format:
    JSON format: {{Questions>: <value>}}. Return only this format; the
    question is always the key; do not change that;
    if there is no value for a key, return {{<key>: "Not found"}}. and
    there is always only one question per input context.
}
User {
    inputcontext = [Example input Context from contract]
    Q: date, operating income
}
Assistant {
    {"date": "Sep. 27, 2013", "operating income": "2472"}
}
User {
    Above where example's use same approach to answer question below; Do not
    use the input context defined above; Use input context provided in the
    input context field below.

    inputcontext= [Input Context for the tested contract]
```

```

Q: [Set of keys that we want to extract]
}
    
```

---

## 9.2 Initial case study results

Below, we present the accuracy metrics obtained from the case study on energy contracts that we evaluated.

### 9.2.1 OpenAI Models

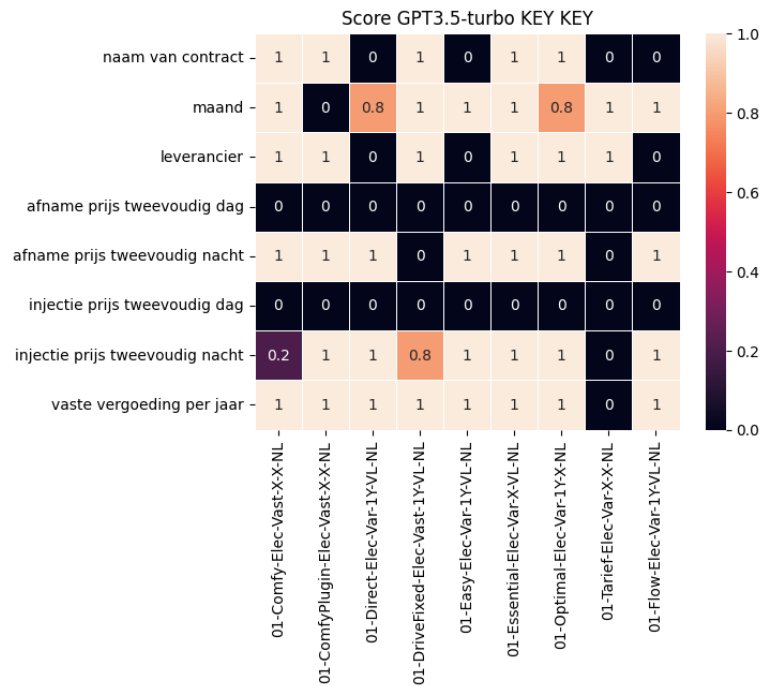


Figure 9.1: Accuracy of GPT-3.5 turbo for the energy use case (Key by Key)

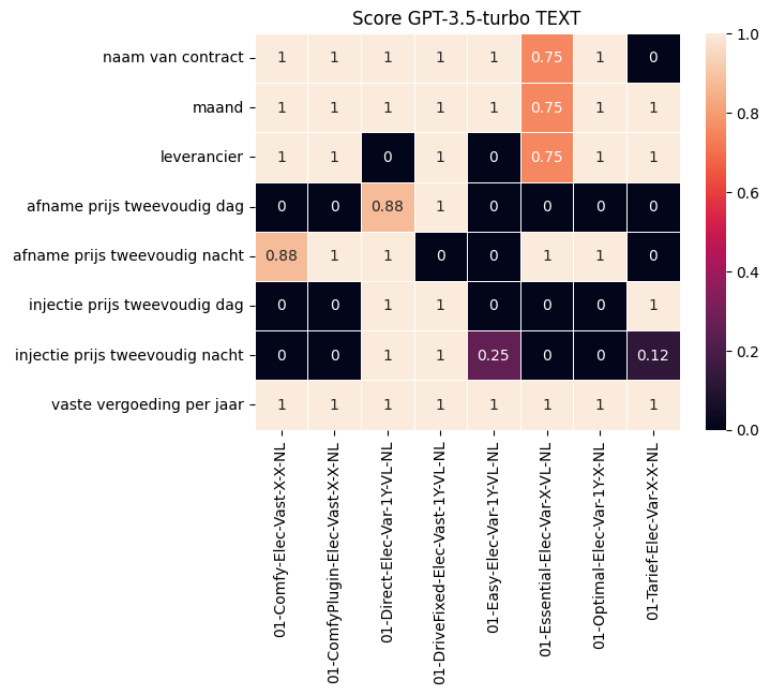


Figure 9.2: Text output of GPT-3.5 turbo for the energy use case

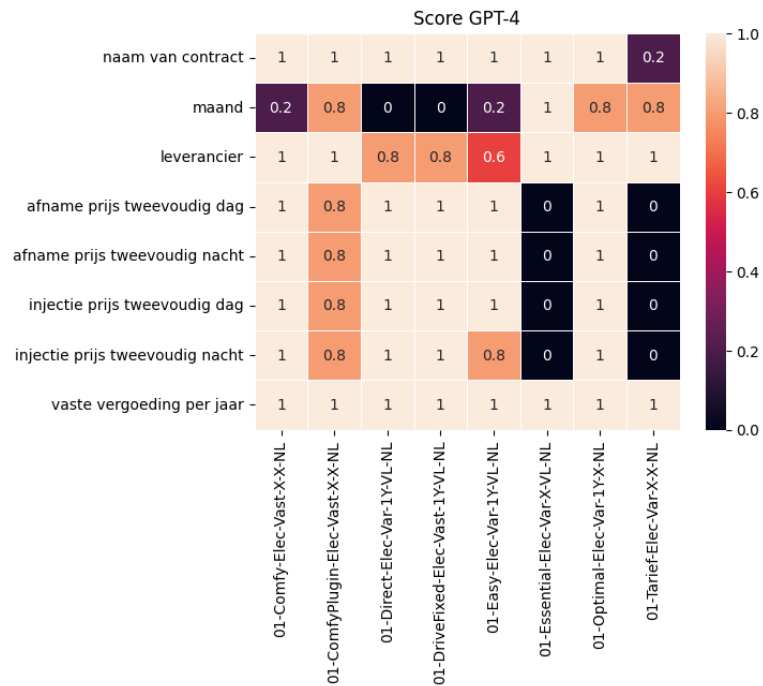


Figure 9.3: Performance of GPT-4 for the energy use case

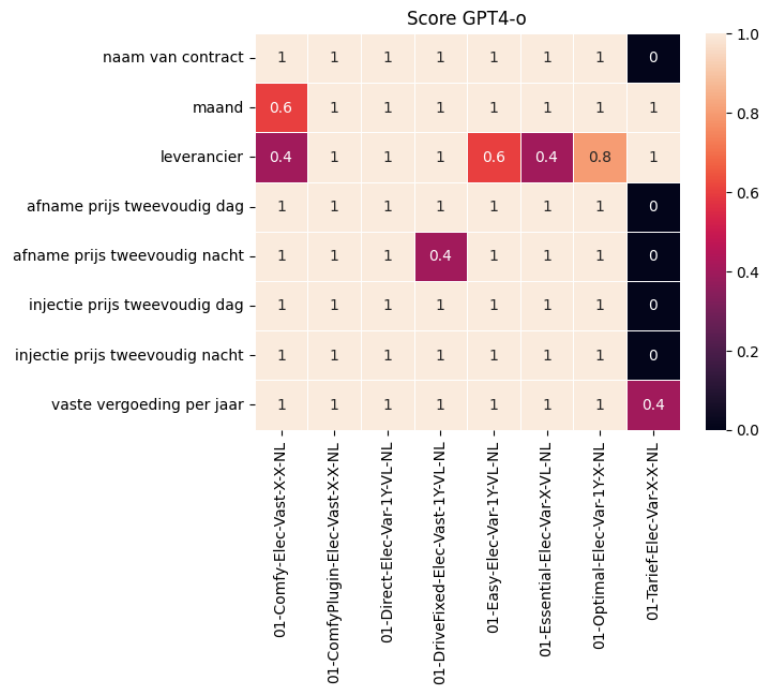


Figure 9.4: Output comparison of GPT-4-o for the energy use case



Figure 9.5: Markdown conversion output of GPT models



## 9.2.2 LLAMA models

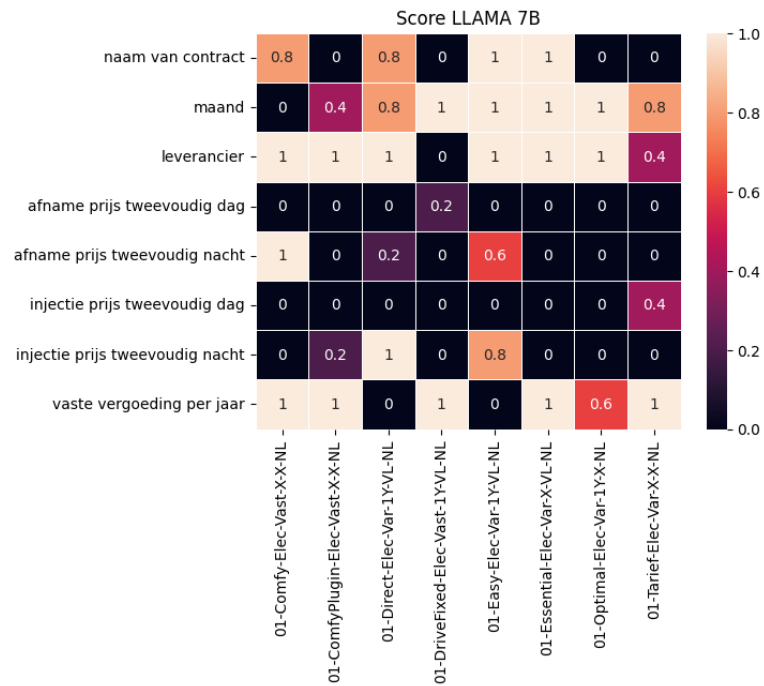


Figure 9.6: Output of LLAMA 7B model for the energy use case

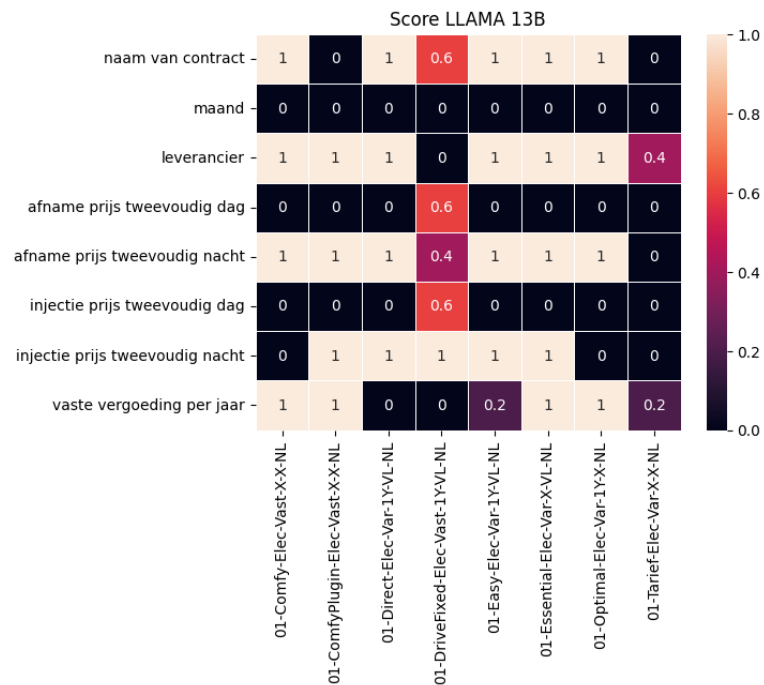


Figure 9.7: Output of LLAMA 13B model for the energy use case

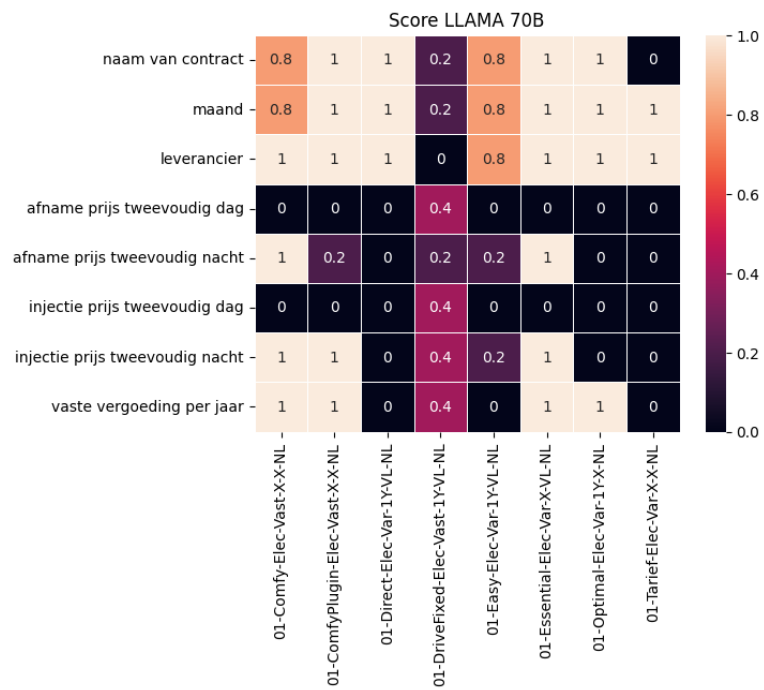


Figure 9.8: Output of LLAMA 70B model for the energy use case

## 9.2.3 Mistral models

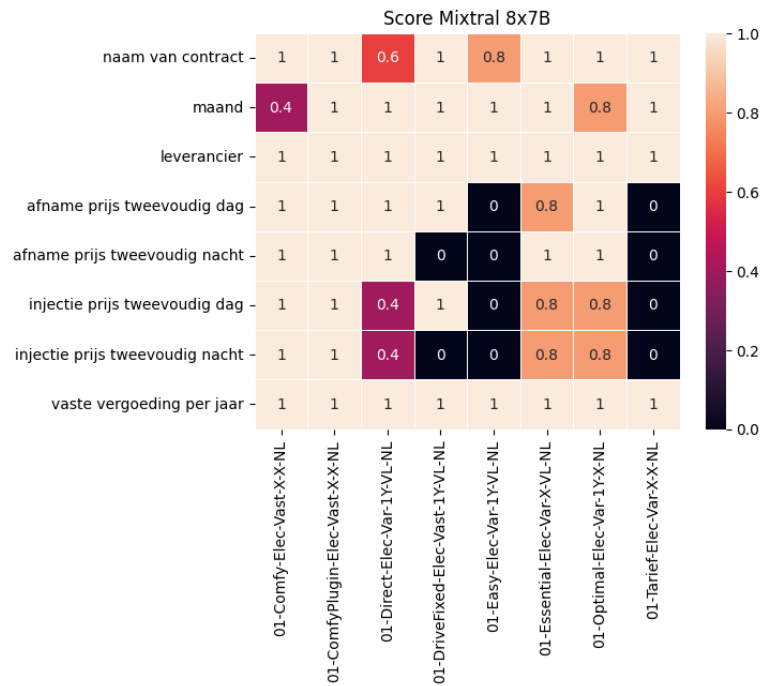


Figure 9.9: Output of Mixtral 8x7B model for the energy use case

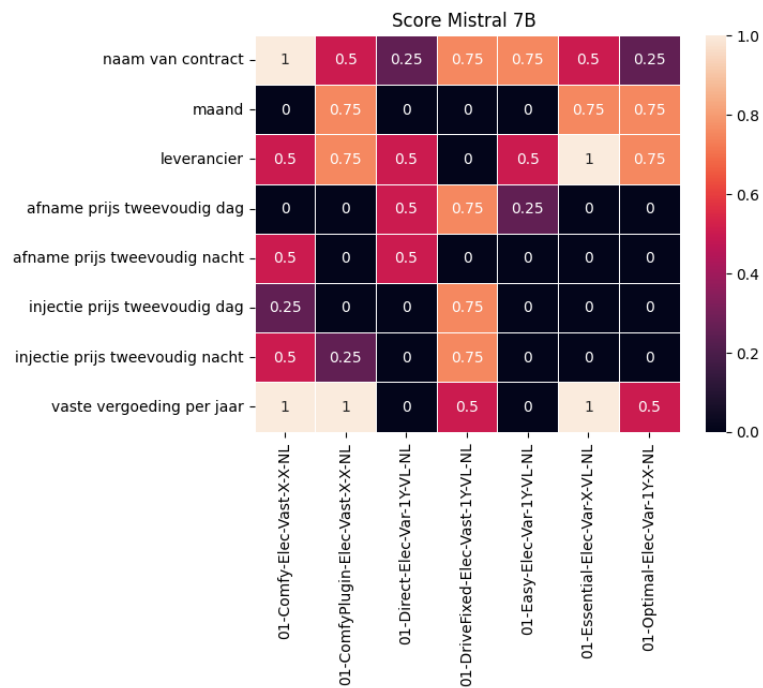


Figure 9.10: Output of Mistral 7B model for the energy use case

### 9.2.4 LLM output visualization in HTML file

**KEY by KEY generated answers[alternative to majority vote] - number of tests: 5**

Contract: 01-Comfy-Elec-Vast-XX-NL			
Key	# of wrong values	# of correct values	# of missing values
naam van contract	0	5	0
maand	0	5	0
leverancier	0	5	0
afname prijs tweevoudig dag	1	0	0
afname prijs tweevoudig nacht	0	5	0
injectie prijs tweevoudig dag	1	0	2
injectie prijs tweevoudig nacht	2	1	2
vaste vergoeding per jaar	0	5	0

Contract: 01-ComfyPlugin-Elec-Vast-XX-NL			
Key	# of wrong values	# of correct values	# of missing values
naam van contract	0	5	0
maand	0	5	0
leverancier	1	4	0
afname prijs tweevoudig dag	1	0	0
afname prijs tweevoudig nacht	1	0	0
injectie prijs tweevoudig dag	1	0	0
injectie prijs tweevoudig nacht	0	5	0
vaste vergoeding per jaar	0	5	0

Contract: 01-Direct-Elec-Var-1Y-VL-NL			
Key	# of wrong values	# of correct values	# of missing values
naam van contract	0	5	0
maand	0	5	0
leverancier	1	4	0
afname prijs tweevoudig dag	1	0	0
afname prijs tweevoudig nacht	1	0	0
injectie prijs tweevoudig dag	1	0	0
injectie prijs tweevoudig nacht	1	0	0
vaste vergoeding per jaar	0	5	0

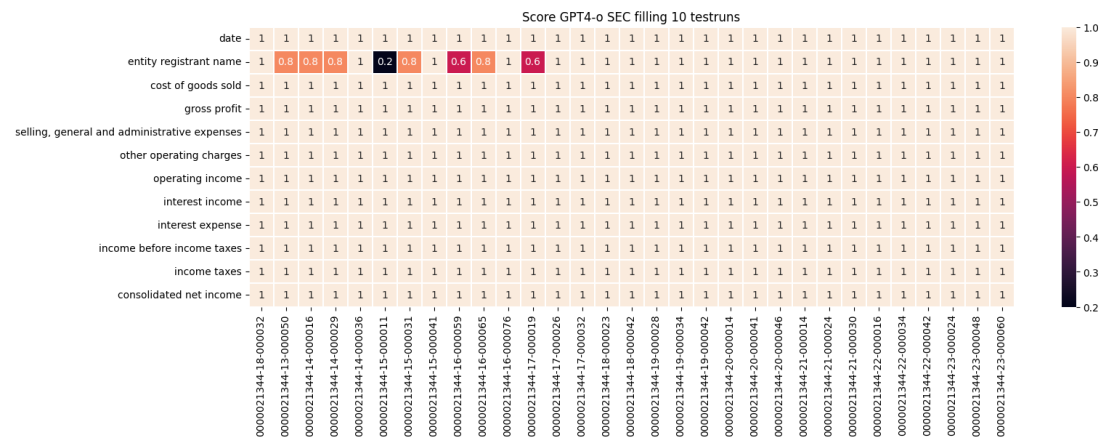
Contract: 01-DirectElec-Elec-Vast-1Y-VL-NL			
Key	# of wrong values	# of correct values	# of missing values
naam van contract	0	5	0
maand	0	5	0
leverancier	1	4	0
afname prijs tweevoudig dag	1	0	0
afname prijs tweevoudig nacht	1	0	0
injectie prijs tweevoudig dag	1	0	0
injectie prijs tweevoudig nacht	1	0	0
vaste vergoeding per jaar	0	5	0

Contract: 01-DirectElec-Elec-Vast-1Y-VL-NL

**Figure 9.11:** An example of the first visualization of the LLM output in a simple HTML file

## 9.3 SEC case study results

Below, we present the accuracy metrics obtained from the case study on SEC filling of Coca-Cola that we evaluated.



**Figure 9.12:** Results from Testing Entire SEC Document with GPT-4-o LLM



# Chapter 10

## Appendix B

### 10.1 English summary

The primary objective of this thesis is to evaluate the effectiveness and usability of large language models (LLMs) in data integration tasks. Data integration involves extracting data from various sources and processing it for use in an integration workflow. This thesis focuses on the data extraction component of data integration tasks, leveraging LLMs to enhance this process. LLMs are a type of artificial intelligence where neural networks are trained on vast amounts of data to generate human-like responses to queries.

To guide this investigation, several research questions have been formulated. The first research question seeks to determine whether LLMs can consistently extract text from large input documents by evaluating their performance over multiple test runs.

The second question is related to the first and focuses on assessing the accuracy of LLMs in generating key-value pairs during question answering tasks. The “key” is the question, and the “value” is the generated answer. Accuracy is measured by comparing the generated answer with the correct response.

LLMs can be categorized into two types: open-source models, which are publicly available and can be downloaded and tested locally, and proprietary models, which are accessible only via vendor APIs. The third research question examines the performance and usability of both open-source and proprietary models, and whether they can be interchangeably used.

Finally, since LLMs can generate incorrect answers, the fourth research question evaluates whether human intervention can enhance the usability of LLMs in data integration tasks.

#### Experiments setup

To answer the research questions, two case studies were analyzed, testing both open and closed-source LLMs to evaluate their performance and limitations. For closed-source LLMs, OpenAI’s GPT-3 and GPT-4 models were tested, while for open-source LLMs, models from Meta AI (Llama2 models) and Mistral AI (Mistral and Mixtral models) were examined. An end-to-end testing framework called the LLM-pipeline (see section 3.3) was developed to facilitate these experiments. This pipeline is responsible for testing and validating the results for both open and closed-source models.

The LLM-pipeline starts with raw input data, which can be either PDF or HTML files, depending on the case study. The Data Extractor module extracts and converts this input data into text that the LLM can process. Due to the limited context window of LLMs, the Data Extractor module also converts the input dataset into a relevant-only dataset, sometimes requiring manual intervention based on case study requirements.

Before the data is sent to the LLM, an intermediate step involves creating a validation set. In this step, a goldenset is built from the original input dataset, indicating what key-value pairs are present in the PDFs. If the dataset includes a metadata file, it can be converted into the goldenset; otherwise, the goldenset is manually constructed.

The core functionality of the LLM-pipeline is the LLM Prompter, which combines the output of the Data Extractor module and the goldenset to prompt the LLM. The goldenset is used to update the prompts to answer questions related to the selected case study. This prompt template was designed iteratively to limit the LLM from hallucinating and providing incorrect answers. Hallucination occurs when the LLM generates output that is not present in the input context. For closed-source models, the OpenAI API is used to interact with the GPT models. To test open-source models the infrastructure of the Vlaams Supercomputer Centrum (VSC) is used, the entire LLM-pipeline is transferred to the VSC for testing. As Open-source models need more GPU memory than a standard computer can provide. The VSC makes its infrastructure available for research purposes, and this thesis leverages that capability.

The output from the LLM Prompter module is fed into the Validation module, along with the validation set, for post-processing the raw textual LLM output into JSON format and calculating the LLM's accuracy using the goldenset from the validation set. The validation process counts correct and incorrect answers and generates a JSON file that can be visualized with the Human-In-The-Loop (HITL) tool or analyzed further using specialized notebooks.

### Human-in-the-loop (HITL)

In addressing the final research question, we designed a human-in-the-loop interface to help developers and end users better comprehend the LLM's output for data integration tasks. The interface visually assesses the LLM's data extraction performance, making it easier for users to evaluate.

The tool is built with Streamlit and uses the HTML PDF renderer to display the PDFs on the interface. It is a standalone tool and does not automatically run tests on VSC or use the OpenAI API. Users must manually execute the LLM-pipeline and download the resulting JSON file. This JSON file can then be uploaded into the tool for further analysis.

The interface features two primary pages. The first page enables users to compare the output of the LLM with the goldenset to assess its accuracy. The second page is intended for scenarios where the goldenset is unavailable; in this case, the tool creates an aggregated set to assist users in constructing the goldenset. Both pages offer a visual representation of the output to evaluate the usability of the input dataset.

Users can visualize and analyze errors and patterns in the LLM output through an intuitive interface, facilitating user-friendly exploration of the LLM's performance.

### Case studies

Two case studies were conducted to analyze the performance and limitations of Large Language Models (LLMs). The first case study involved energy contracts as the input dataset. This dataset was manually built by downloading 8 energy contract PDF documents from [mijnenergie.be](https://www.mijnenergie.be)<sup>1</sup>, where we tested 8 different key-value pairs. The second case study focused on the financial sector, specifically using SEC filing documents of Coca-Cola. We downloaded 31 documents, which provided insights into the company's financial performance, and tested 12 different key-value pairs. In both case studies, the LLMs were tested on two main areas of the PDFs: questions with answers in plain text and questions with answers in tables.

We conducted two case studies to analyze the performance and limitations of Large Language Models (LLMs). The first case study involved using energy contracts as the input dataset.

---

<sup>1</sup><https://www.mijnenergie.be/>

This dataset was manually constructed by downloading a set of energy contract PDF documents from [mijnenergie.be](http://mijnenergie.be). The second case study focused on the financial sector, using SEC filing documents of Coca-Cola, which detail the company’s financial performance.<sup>2</sup> In both case studies, we tested the LLMs on two main areas of the PDFs: questions answerable from paragraphs and questions where answers are embedded in tables.

A clear pattern that was observed is that the tested LLMs struggled with extracting data from tables, showing the lowest accuracy on these key-value pairs. Using the “HITL” visualization tool, we noticed that the LLMs often misinterpreted the data and retrieved answers from adjacent columns. We intentionally did not specify which column contained the correct answers to test the learning capacity of the LLMs. Top models (GPT4-o and Mixtral 7x8B) successfully learned this pattern, while smaller models (LLAMA 7B, 13B, and Mistral 7B) did not. The detailed accuracies are available in Table 10.1, Table 10.2, and Table 10.4. The results show that close-source models have the highest accuracy and consistency among all tested models, answering the first two research questions.

M7B	L7B	L13B	L70B	GPT3.5 MD	GPT3.5
34	41	45	48	61	63

**Table 10.1:** Score of LLMs tested (Part 1), score ranging from 0 to 100, 100 meaning that the model answers every question correctly.

M8x7B MD	GPT4	M8x7B	GPT4-o
69	77	79	87

**Table 10.2:** Score of LLMs tested (Part 2), score ranging from 0 to 100, 100 meaning that the model answers every question correctly.

Model Descriptions
L7B refers to LLAMA2 7B
L13B refers to LLAMA2 13B
L70B refers to LLAMA2 70B
M7B refers to Mistral 7B
M8x7B refers to Mixtral 8x7B
M8x7B MD refers to Mixtral 8x7B with input context converted to markdown
GPT3.5 refers to OpenAI’s GPT3.5 turbo
GPT3.5 MD refers to OpenAI’s GPT3.5 turbo with input context converted to markdown
GPT4 refers to OpenAI’s GPT3.5
GPT4-o refers to OpenAI’s GPT4 omni

**Table 10.3:** Full names and descriptions of the models tested.

Mistral 7B	Mixtral 7x8B	GPT4-o
69,34	93,36	99,30

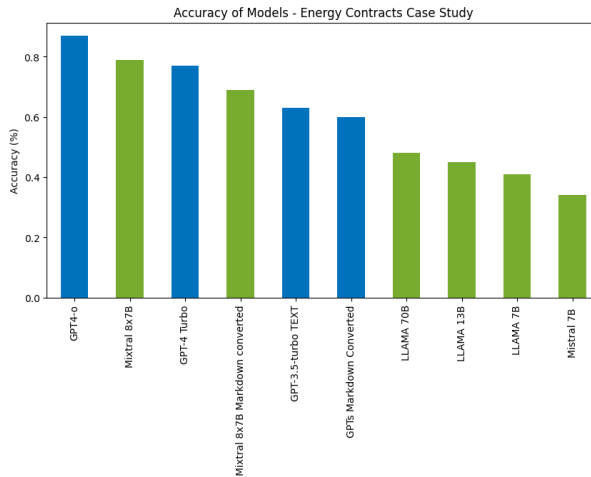
**Table 10.4:** Score of all the LLMs tested, score ranging from 0 to 100, 100 meaning that the model answer every question correctly

To answer the third research question, we compared all tested models across the two case studies to determine which open-source models could be alternatives to proprietary models. The accuracy measurements from the case studies clearly show that GPT-4-o outperforms all

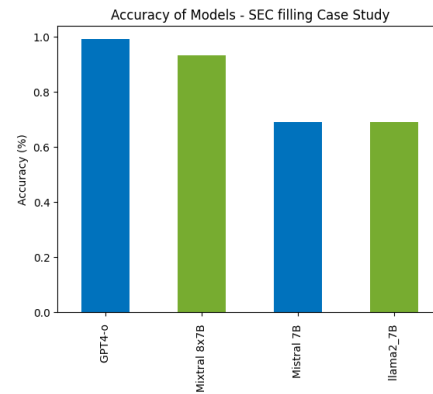
<sup>2</sup><https://investors.coca-colacompany.com/filings-reports/all-sec-filings>



other models. Llama2 models showed an accuracy of less than 60%. Remarkably, the Mixtral 7x8B models demonstrated superior performance compared to GPT-4 and GPT-3.5, despite using fewer parameters and less VRAM. This suggests that the Mixtral model could be an open-source alternative to close-source GPT-4 models. Figure 10.1 and Figure 10.2 show the accuracy of the tested large language models (LLMs). In the graphs, blue represents close-source models and green represents open-source models.



**Figure 10.1:** Accuracy of Open-Source and Closed-Source Models on the Energy Contracts Case Study



**Figure 10.2:** Accuracy of Open-Source and Closed-Source Models on the SEC filling case study

## Limitations

This study identifies several limitations that warrant further exploration in future research. The first limitation is that the case studies tested primarily involved only two different structures of PDF files. Although the results obtained in this study are promising, they do not fully represent the potential of the tested LLMs. Moreover, the dataset used in the study was relatively small. Testing the LLMs on two different datasets in these case studies has shown some overlap. However, in the future, this study should be extended to larger datasets. The developed methods and techniques can be applied as new LLMs are released.

Another significant downside of the developed LLM-pipeline and the Human-In-The-Loop (HITL) approach is that they work only with batch processing. This means that users can only request a dataset to be preprocessed by the LLM pipeline, requiring them to specify the key-value pairs they need in advance. Additionally, the HITL does not support dynamic interaction with the LLM. Future work should focus on directly connecting the HITL to the LLM inferencing. For closed-source models, this is not a difficult task as they work with API's. However, the way open-source models are run on the VSC limits them to batch processing only, as the VSC does not have an interactive node for the GPUs that are used for inferencing.

One aspect not covered in this thesis is the fine-tuning of models. Smaller models that perform worse can be fine-tuned with more domain knowledge and specific prompting to enhance their performance in future work. The developed LLM-pipeline can be used to test these fine-tuned models to see how they perform on domain-specific tasks.

The last limitation is related to the context window of the LLMs. In the case studies tested in this thesis, we manually reviewed the dataset and selected only the relevant pages of PDFs that would fit into the context window of the LLMs being tested. This process can be very time-consuming, especially for large datasets with diverse types of PDFs. A potential solution is the use of Retrieval-Augmented Generation (RAG) systems. This approach would allow the LLM-pipeline to automatically extract only the relevant input context needed for the current

prompt sent to the LLM. Although we tried this approach before manually extracting the relevant data, it added extra dimensions of factors that could influence the LLM's performance by providing incorrect input context. To keep the testing more deterministic, we chose not to implement this in the LLM-pipeline and instead used pre-extracted PDFs that were consistent across all LLM tests.

In conclusion, this research offers valuable insights into the usability and accuracy of LLMs, highlighting both their potential and limitations. The developed technique shows that LLMs can effectively be used for the extraction phase in data integration tasks. Moreover, the study demonstrates that the LLM-pipeline and HITL framework can aid in future testing of new LLMs and streamline the selection of suitable models.