Large Language Models for Structuring and Integration of Heterogeneous Data

Henrik Bongertmann
Institute of Computer Science, University of Rostock
Rostock, Germany *henrik.bongertmann@gmail.com*

Benjamin Nast
Institute of Computer Science, University of Rostock
Rostock, Germany *benjamin.nast@uni-rostock.de*

Leon Griesch
Institute of Computer Science, University of Rostock
Rostock, Germany *leon.griesch@uni-rostock.de*

Henry Rotzoll
Sachgebiet GEO, Datenverarbeitungszentrum MV GmbH
Schwerin, Germany *h.rotzoll@dvz-mv.de*

Kurt Sandkuhl
Institute of Computer Science, University of Rostock
Rostock, Germany *kurt.sandkuhl@uni-rostock.de*

Abstract

The implementation of artificial intelligence (AI) in the public sector offers great potential. Repetitive and labor-intensive tasks can be automated to improve overall efficiency. Generative AI, in particular, opens up new possibilities for structuring and integrating heterogeneous data sources. At the same time, AI introduces challenges such as technical complexity and ethical issues that must be addressed during development and implementation. This paper investigates the potential and challenges of using AI in the extract, transform, load (ETL) process in a public sector study. Our findings demonstrate that open-source large language models (LLMs) can efficiently transfer over 5,000 unstructured documents into the structured format of a relational database, achieving a success rate of approximately 96%. The quality of the results was significantly improved through optimization measures, particularly in terms of prompt engineering and post-processing. While the results are encouraging, challenges remain, including processing extensive documents and adapting the data model to greater complexity.

Keywords: Large Language Model, Open-source, Data Heterogeneity, Data Structuring, Data Integration, ETL Process.

1. Introduction

The implementation of artificial intelligence (AI) in the public sector offers great potential [29]. It can help to improve value creation [26], support decision-making processes [10], or enhance efficiency by automating repetitive and labor-intensive tasks [2], [21]. Generative AI, in particular, is opening up new possibilities for structuring and integrating heterogeneous data sources [18]. At the same time, AI introduces challenges such as technical complexity [16] and ethical issues [24] that must be addressed during development and implementation.

This paper explores the potential of large language models (LLMs) on-premises to facilitate the efficient transfer of heterogeneous data sets into relational databases. An exploratory case

study is conducted, which involves the transfer of unstructured documents into a structured format using descriptive documents from an art collection of churches in northern Germany. Different documentation methods employed by the authors make it impossible to use traditional ETL approaches for replacing the manual transfer by humans.

The primary objective of this work is to develop a solution for the requirements of the case study and, in the course of the development process, assess the efficiency and effectiveness of LLMs in structuring and integrating heterogeneous data sets. Due to data protection regulations, the use of public cloud-based LLMs is not permitted, i.e., the data processing has to be conducted on-premises using open-source LLMs. In this context, a comparative analysis of diverse open-source LLMs was performed to identify suitable pre-trained models that align with the data structuring and integration requirements of the case study.

Our work includes various activities to improve the LLM outcomes (e.g., prompt engineering and post-processing) and leads to recommendations for a comprehensive approach to leveraging open-source LLMs in data structuring. Subsequent to this, the case study results are evaluated, and findings derived from the methodological approach are discussed. The following research questions (RQs) are addressed:

- RQ1: To what extent can LLMs support the structuring of heterogeneous data and integration into a common data schema?
- RQ2: Which open-source LLMs are particularly suitable for converting unstructured text documents into a structured format?
- RQ3: What measures can be implemented to improve the results?
- RQ4: What general recommendations for LLM-based data structuring and integration can be derived from the case study?

The paper is structured as follows: Section 2 provides an overview of LLMs and their potential for use in the extract, transform, load (ETL) process. Section 3 introduces the case study and compares selected open-source LLMs in terms of their ability to structure data. The developed approach for an LLM-based ETL process is described in Section 4. The results and general findings are discussed in Section 5. A conclusion and an outlook are given in Section 6.

2. Background and Related Work

2.1. Large Language Models

A key feature of LLMs is their ability to generate coherent, diverse, and contextually relevant text over long passages. They can be used for translation, summarization, and question-answering without task-specific training data [14]. The well-known GPT (generative pre-trained transformer) models from OpenAI, equipped with the chatbot frontend ChatGPT, can also be used for translation, grammar correction, or email writing [11]. The challenges and limitations of the development and application of LLMs can roughly be divided into two groups [19]: From a technical point of view, these may encompass substantial resource requirements, scalability, data dependency, or reliability. Ethical considerations include discrimination, data protection, potential for misuse, and explainability.

Prompt engineering enables the targeted control of models to generate precise answers and to extend the range of applications beyond traditional question-and-answer scenarios [30]. The optimization of prompts can, e.g., be achieved through the use of patterns [28], self-correction mechanisms to avoid undesired behavior [30], or the chain-of-thoughts method [27] to enhance logical reasoning by introducing intermediate steps in natural language. Brown et al. [8] describe contextual learning as a key concept for this. In this context, an LLM is instructed to

adopt a specific behavioral paradigm through the provision of a limited number of examples. A distinction is made between zero-shot (no examples), one-shot (one example), and few-shot (multiple examples) learning.

In scenarios where the limited context length provided by the prompt is not sufficient to provide the LLM with the necessary context information, techniques such as fine-tuning or retrieval augmented generation (RAG) emerge as essential tools to provide pre-trained models with new data. Fine-tuning entails the recalibration of the weights of a pre-trained model through extensive training with a substantial volume of sample data drawn from a specific application domain. This approach can enhance performance on specific tasks; however, it is essential to note that it requires a substantial data set for each new task, which carries the risk of incorrect generalization due to overfitting the sample data [8]. RAG is an approach in which pre-trained LLMs are integrated with an external knowledge base. This facilitates the LLM in producing precise and well-founded responses based on the knowledge base provided without the necessity of training the model with additional data [15].

2.2. LLMs in the ETL Process

A substantial body of existing literature addressed the structuring and integration of heterogeneous data. Remadi et al. [18] showed how unstructured data could be successfully integrated into a graph database. Vijayan [25] compared different prompt engineering techniques in structuring heterogeneous data into a given SQL format and showed that iterative prompts are effective for this. In [13], the performance of GPT-3.5 in extracting and subsequently structuring in JSON format is compared with conventional natural language processing techniques. An LLM-based approach for extracting specific attributes from data lakes was proposed in [6]. Omar [17] investigated the performance of ChatGPT in extracting and translating entities and relationships into an entity-relationship model. Another publication successfully demonstrates the extraction of relational triples using a zero-shot approach that does not require fine-tuning [31]. Similar successes were achieved in extracting relationships from Holocaust testimonies with GPT-3.5, which can be used to build knowledge graphs [5]. The precise extraction of relational triples from tweets using open-source LLMs was demonstrated in [22]. An interesting alternative is the approach of using vector databases for direct interaction with unstructured data [20]. The results of these studies show the diverse application possibilities of LLMs in data structuring and point to a promising development in this field.

Two additional papers illustrate the potential of LLMs to enhance the efficiency of the ETL process by automating specific tasks of data extraction, transformation, and integration. The work of Anu Mohan et al. [4] presents a user-friendly method for performing automated data transformations in natural language that reduces technical barriers in database management. An LLM, in combination with Python, is used to transform CSV data into JSON objects and then manipulate them based on user queries. In [23], an approach that uses LLMs to generate SQL code and automatically transform source data into target data is described. This method integrates domain-specific knowledge in the formulation of SQL statements and optimizes the statements to the LLM in an iterative process based on an error detection mechanism. The approach shows promising results and can be used in different areas regardless of the application.

The above research shows that LLMs are capable of generating structured formats from unstructured text. This capability has the potential to significantly reduce the manual effort involved in integrating heterogeneous data sources into relational databases by automating the steps of data extraction, transformation, and integration. Concurrently, a necessity for research in the domain of data structuring was identified, particularly with regard to the structuring of complex document content and its conversion into a suitable database model for subsequent integration into a relational database.

3. Application Context and LLM Selection

3.1. Case Study

The case study described in this section is about the application of open-source LLMs for structuring heterogeneous text documents and integrating the extracted data into a relational database. Additional material, e.g., example documents, prompts, and the complete results, is provided as a dataset to ensure reproducibility [7]. The documents originate from an art collection belonging to churches in northern Germany. The objective is to facilitate the creation of an inventory and to make the art objects accessible via an interactive map[1]. The database contains 6,509 Word (DOC and DOCX) and PDF documents, which lack a uniform structure and include some duplicates in different formats. The documents are divided into four folders according to the region of the art object, which were recorded differently during documentation.

The use of different data collection methodologies in the documentation process of the art objects, results in a variety of data inconsistencies among the documents, making it difficult to integrate them into the relational database. These are essentially two types of structural heterogeneities [12]: Firstly, the different naming of attributes represents a definitional conflict between the documents. Secondly, the art objects were documented to different extents, resulting in inconsistent domain coverage. Furthermore, representation conflicts may arise at the semantic level when documents contain different formats for the same data elements (e.g. dates). These discrepancies must be resolved prior to integrating the documents into a relational database. Although the works identified in Section 2.2 describe examples for structuring and implementation, this specific use case is not covered. A particular concern here is the use of open-source LLMs, because the underlying documents contain sensitive information.

3.2. Comparison of Open-source LLMs

This section compares several open-source LLMs suitable for implementing the case study. Open-source LLMs have a significant advantage over commercial models as they can be run locally, ensuring complete control over sensitive data and minimizing potential data breaches. A pre-selection of models was made based on benchmark tests, focusing on their ability to convert texts into structured formats alongside criteria such as performance, context length, training data, and multilingual support. They were evaluated using various benchmarks in six categories: General Knowledge (MMLU, BBH), Reasoning (ARC-C, GPQA), Multilingual (MI MMLU, MGSM), Math (GSM8k, MATH), Long Context (Qasper, SQuALITY), and Coding (HumanEval, MBPP) [9].

A fair comparison was ensured by testing the models under identical conditions with few-shot prompts and a temperature of zero. In addition to several open-source models, two commercial models were compared. In the study of [1], *Llama 3.1* performed well across multiple benchmark tests, suggesting stable capabilities. The *Gemma-2* model from Google DeepMind showed above-average performance, particularly in question answering, logical reasoning, and coding. *Phi-3.5-mini* displayed solid results for its size, especially in MMLU and reasoning benchmarks, while the *Phi-3.5-MoE* model performed significantly better due to its mixture-of-experts architecture. The *Mistral-7B* model performed well but trailed behind larger models like *Mistral-Nemo (12B)* and *GPT-4o mini*, the latter scoring the highest overall, so we will compare our results with *GPT-4o mini* as a benchmark later. Open-source models averaged over 60% in benchmark scores, while *Mistral-7B* lagged at just under 50%.

For the case study, we had an NVIDIA GeForce GTX 1070 with 8GB graphics memory (VRAM) (dedicated memory), an Intel Core i7-6820HK CPU (2.70 GHz) processor, and a working memory of 32GB RAM at our disposal. Based on discussions with the relevant experts, this

[1] `https://www.geoportal-mv.de/gaia`

is a realistic limitation, as such organizations or enterprises do not usually have better hardware available in practice. Following the benchmark tests and our available hardware, we selected *Mistral-7B-Instruct-v0.3*, *Llama-3.1-8B*, *Gemma-2-9B*, and *Phi-3.5-mini* for further investigation. These models demonstrate high performance, reflecting the current state of AI research. After analyzing a larger number of documents, we selected 30 representative test documents (ten DOC, ten DOCX, and ten PDF files). In our tests, the models were able to structure up to 7% of these in CSV (Mistral and Llama), 23% in SQL (Llama), and 80% in JSON (Mistral). For the setup, a uniform temperature of 0.7 was used with no optimization via pre- or post-processing. The prompt included formatting instructions and a template in the required format [7].

As these success rates alone are inconclusive, the generated responses' performance and quality were also analyzed. Phi demonstrates the fastest performance (50 seconds, 21 tokens per second), and Gemma shows both the longest response time (87 seconds) and the lowest value of tokens processed per second (twelve). However, the quality of these two models is significantly reduced due to the quality of the model responses, especially in summarizing (Gemma) and simplifying (Phi) the information. In contrast, the Llama model truncated long sections of text in the documents, while Mistral rephrased sentences. Although Mistral has a higher success rate in generating valid JSON formats than Llama (80% vs. 66%), the quality of the model response ultimately plays a more decisive role in model selection.

Because Llama was the only model that could transfer information without hallucination or modification, we used it and the target format JSON in our case study. The issue of response length requires further attention, which can be addressed by prompt engineering, while the success rate can be improved by post-processing the responses (see Section 4).

4. LLM-based ETL Process

4.1. Implementation of an LLM-based ETL Process

For successful integration into the database, the documents must undergo the three phases of the ETL process (see Figure 1). The starting point is a data source with text documents and a suitable data model for transferring the information. As some documents existed in multiple file formats, these were first filtered out. DOCX files were prioritized over PDF files and PDF files over DOC files to ensure the best possible text extraction. This reduced the data volume from 6,509 to 5,670 documents. In preparation for data structuring, a JSON template (see [7]) was created, which represents the one-dimensional table with 29 columns in the database given by the case study and contains the attribute names of the art objects recorded in the documents (e.g., *description*, *location*, or *artist*). It supports bringing the unstructured documents into a standardized format. The implementation process is comprised of four steps:

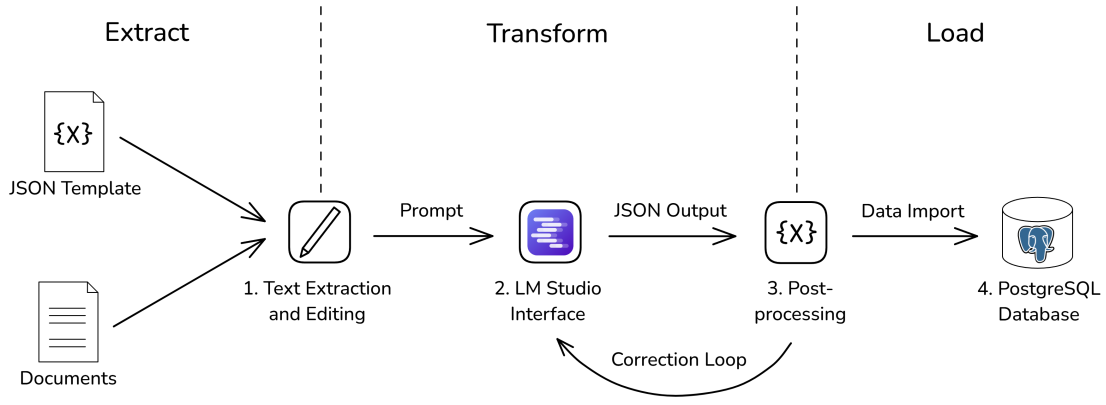

Fig. 1. LLM-based ETL process

1. Text Extraction and Editing: In the initial phase, the text is extracted from the documents, which involves a pre-processing step of data cleaning. The Python package *textract* is used to extract Word documents (DOC and DOCX files) and *PyMuPDF* for PDFs. DOCX files preserve headers and paragraphs during text extraction, while DOC files lose headers and extract content line by line, leading to information loss issues as the LLM is unable to recognize where the paragraphs end clearly.

For PDFs, the text is also extracted line by line, resulting in redundant headers from each page being inserted within the extracted text. Finally, unnecessary strings, spaces, and line breaks have been removed to enhance the text quality.

2. LM Studio Interface: In the second step, the extracted text is transmitted to the selected LLM, together with the JSON template of the data model and the instructions for structuring the information, thus forming the three-part prompt:

Prompt for structuring documents in a valid JSON format

The following object description contains information about a specific art object:

{document_text}

Task: Transfer all information from the object description into a valid JSON object using the following template:

{json_template}

Note: The information should be transferred complete and unchanged. Transfer the entire text for each feature until the next feature begins. No information should be lost. Use the specified keys from the template for the JSON object in the response. Format all values in the JSON object as a text string. Enclose the keys and values in double quotes to ensure that the response corresponds to the correct JSON syntax. The response should only contain the JSON object and no additional text.

The open-source LLM *Llama-3.1-8B* is employed, which is hosted on a local server using the *LM Studio* tool. The request is directed to the local LLM via REST API. The value 0.4 was selected in the configuration for the temperature to generate more deterministic model responses so that the information from the document remains unchanged. Our experiments with different temperature values (see [7]) showed that lower creativity led to LLM adopting the attribute names from the documents instead of the template due to its stochastic nature (e.g., *photo no.* instead of *photo number*), and higher creativity led to invented attribute names that did not appear in either the document or the template (e.g., *church district* instead of *parish*).

In the pre-tests [7], we found that approximately 2,000 input and 1,000 output tokens are sufficient for the efficient processing of the documents that only comprise a few pages (see example documents in [7]). We therefore set the maximum context length to 8,000 tokens. It also enables the interception of longer documents in cases where the information from the object description cannot be reliably structured. Furthermore, a general system prompt is stored in the configuration, which prompts the model to respond in German. This ensures that there is no mixing of languages in the model's response, as was the case when comparing the target formats (see Section 3.2). The output of the model is a text with the information transferred from the document in JSON format.

3. Post-processing: The model output is then processed through a post-processing pipeline, which validates the generated JSON string and converts it into a valid JSON format. Initially, the response is subjected to a process of refinement to eliminate superfluous character strings and line breaks. It is imperative to ensure the use of double quotes for the keys and values in the JSON string and to verify that no additional instances of double quotes occur elsewhere.

Following the completion of the post-processing of the model output, the presence of a valid JSON format in the cleaned response needs to be verified. For the response to be successfully integrated into the database, it is essential to ascertain that the JSON string can be converted into

a JSON object and that the JSON output does not contain any invalid keys that are not present in the data model. In the event that the model fails to generate a valid JSON format, the system logs the incorrect output and initiates a new request to the LLM. Conversely, if the model generates a new response, the post-processing pipeline is executed once more. If there are two incorrect outputs, the document is marked as failed. Further correction loops are omitted due to the longer runtimes and the sharply declining success rates in generating the model response.

4. PostgreSQL Database: Finally, the documents are integrated into the relational database. Once the model output has successfully passed through the post-processing pipeline, the valid JSON object is automatically imported into a *PostgreSQL* database. To ensure that the JSON objects correspond to the data model that was used as the JSON template, missing columns are supplemented with empty values. This is only important for the initial data set that creates the table, so that it corresponds to the data model. In addition, the JSON object is given information about the source folder and the file path of the transferred document. This metadata can be relevant when the database is used later to facilitate the retrieval of the documents.

To facilitate the transparent management of AI-generated content, the source text extracted from the document is also saved in the JSON object, enabling retrospective verification of its accuracy and the retrieval of any missing information. The JSON object is initially converted into a *Pandas DataFrame*, a type of table, as the Pandas library offers a function for converting DataFrames into SQL statements. These SQL statements can then be imported into PostgreSQL using the Python package *SQLAlchemy*. The importation of the initial data record initiates the automatic creation of a new table. In the event that the table has been previously created, the new data records are imported into the existing database.

4.2. Results

The performance results of the LLM-based ETL process are summarized in Table 1. For the complete results and comparison to the pre-tests, please refer to [7]. A total of 5,443 out of 5,670 documents were successfully imported into the database. Therefore, the success rate of the automated data structuring process is approximately 96%. The correction loop was used for around 5% of the documents, achieving a success rate of approximately 64% (184 out of 288 correction runs). The response time per document is 43 seconds and is lower than in the pre-test due to the shorter average response length. The LLM-based process thus required a period of around three full days (71 hours, around 75 minutes per 100 documents) for the total data volume of 5,670 documents.

Table 1. Performance of the LLM-based ETL process for the different source folders

Source Folder	Number of Documents	Succesful Imported	Correction Runs	Answer Length	Answer Time	Runtime Total
A	842	829 (98%)	55 (7%)	698 Token	38 s	approx. 10 h
B	1,729	1,642 (95%)	79 (5%)	651 Token	36 s	approx. 18 h
C	1,199	1,124 (94%)	69 (6%)	786 Token	48 s	approx. 17 h
D	1,900	1,848 (97%)	85 (5%)	817 Token	49 s	approx. 27 h
Total	5,670	5,443 (96%)	288 (5%)	742 Token	43 s	approx. 71 h

The structuring process was executed with partial data sets from four distinct source folders, all of which exhibited a comparable success rate ranging from 94% to 98%. A notable distinction emerged in the response time of the first two folders, which exhibited a reduction of approximately ten seconds per document compared to the latter two folders. This can be attributed to a substantially shorter response length of approximately 100 tokens per response. The correction loops exhibited a consistent level, ranging from 5% to 7%.

To obtain a comprehensive understanding of the outcomes, it is imperative to consider the unsuccessful attempts at data structuring. Incorporating the correction loops, approximately 9% of the 5,958 (5,670 documents + 288 correction runs) were unsuccessful. The investigation into the root causes of these errors revealed that around 24% of unsuccessful attempts could not be attributed to the output of the LLM. This category includes 113 textless documents (e.g., image-only documentation) and 10 documents that exceed the 8,000-token maximum. However, these are irrelevant as they do not contain any inventory information. The remaining 392 unsuccessful attempts were due to erroneous responses from the LLM. Of the 515 failed attempts in total, 34% were related to the JSON format not corresponding to valid JSON syntax, primarily due to incorrectly placed double quotes. A further 42% of the failed outputs contained invalid column names that did not correspond to the specified data model, either due to hallucinations or because a similar column name from the document was used instead of the template.

Until now, it has not been possible to verify which information was extracted from the documents and to what extent. An SQL query was formulated to analyze the completeness of the imported data records. The evaluation of the data completeness in the 29 columns from the database indicates that certain attributes were covered very consistently by the LLM, while others contain only a few values. Ten of the 29 columns demonstrated data completeness of at least 90%, with the object name (99%), the description (97%), and the location (94%) exhibiting particularly reliable transfer from the documents.

It is important to note that the location information provided in the documentation is essential for the geolocalization of the objects. While the location code is absent in 69% of the records, the location name is only missing in 6%, so the location codes can be added later by a mapping script. Conversely, three columns exhibited data completeness of less than 10%: quantity (8%), polychromy (7%), and loan agreements (1%). It should be noted that the incomplete data precludes the determination of whether the missing details are included in the documents. The absence of a standardized structure to measure the expected data per column, due to the differing documentation of the art objects, precludes the ability to make any definitive statements about the completeness of the data transferred from the documents.

5. Discussion

5.1. Evaluation of the Results

The LLM-based ETL process developed as part of the case study successfully imported approximately 96% of the 5,670 documents into the database. The average response length of 742 tokens confirms the successful transfer of information from the documents to the database.

For the qualitative evaluation, the representative sample of 30 documents was selected (see Section 3.2), and the data records were evaluated for incomplete, missing, and incorrect information in the 29 data fields. Overall, the amount of missing information in the sample was very small, typically due to poor data quality or a lack of structure that makes it difficult to accurately match data fields, for example, when information is only contained in the document header. Incorrect information is uncommon, as data is generally transferred directly from the document text without modification.

Several entries, such as restorations, were not explicitly recorded in the document under the name but were extracted from the description of the art objects and assigned to the correct data field. This is a positive side effect. However, the incomplete information in one data field is conspicuous. The description of the art objects, which can extend over several pages of the document, was often cut off after the first paragraph. This phenomenon is particularly prevalent in PDF and DOC files, which lack a coherent structure due to their line-by-line text extraction process. This makes it difficult for the LLM to recognize paragraphs in their entirety, resulting in incomplete descriptions.

Despite these limitations, the descriptions of 99% of the data records were successfully transferred, albeit partially. However, the information is not entirely lost, as the extracted source text remains accessible via the database, enabling a transparent review of the information and facilitating the recovery of any missing data.

Besides the object description, the geoinformation of the recorded art objects is particularly relevant for further processing of the data. This is due to the necessity of integrating the art objects into the map application at their respective locations. The transfer of 94% of the place names and 99% of the descriptions was reliable, so the database contains the basic information necessary for effective use of the data. The 259 records lacking geoinformation require manual review to determine their relevance, with possible removal from the database if they do not contain information relevant to documentation.

In total, 227 out of 5,670 documents could not be successfully imported into the database. The 123 documents that could not be processed by the LLM can be discarded. While 113 documents contained no text, the remaining ten could not be processed due to the maximum context length because they contained too much text. The 104 documents that could not be imported due to invalid model outputs can be integrated into the database by further import attempts or manually, after further failed attempts, if they contain relevant object descriptions.

5.2. Recommendations for a General Approach

Based on our results, we derive the following aspects that should be considered for a general approach to the use of LLMs for data structuring and integration (RQ4). **Ensure Data Quality:** A standardized file format and the preparation and cleaning of the extracted texts can improve the quality of the results. **Parameter Fine-tuning:** The temperature parameter can be used to adjust the creativity of the models during response generation and thus influence the quality of the results. **Prompt Engineering:** Using proven prompting methods, such as one-shot learning, is recommended. As there is no established method for evaluating prompts [3], various prompting methods should be tried out. The results can be optimized according to the trial-and-error principle. **Template for Output:** In one-shot learning, the model is given an example for generating the response. Similarly, a template of the data model in JSON format helps the model generate a valid JSON output. **Post-processing:** Post-processing is essential for an effective and efficient structuring process. It can significantly increase the success rate of generating valid JSON formats. **Correction Loop:** Additional requests to the LLM can be used to check and correct output or generate a new output. **Model Selection:** The selection of a suitable model should represent the best relationship between performance and quality of results. The length of the response is an indicator of a reasonable quantity, but not of the correctness of the output. A manual check of the output is always necessary. **Model Size:** Depending on the hardware, the largest possible model should be used that can be loaded into the VRAM. If the model is executed in the RAM, the performance decreases considerably. **Optimize Runtime:** The prerequisite for local operation of an open-source model is a graphics processing unit (GPU) with sufficiently large dedicated memory, as the graphics memory is specially optimized for fast parallel processing. The technical equipment determines the speed of the model during token generation.

5.3. Limitations

The Llama model was selected based on a comparison of several known open-source models. A more comprehensive evaluation of other models could potentially identify more suitable open-source LLMs. In addition, the optimization measures (e.g., prompt engineering) were not pursued further in other models. Fine-tuning these to the specific domain of the documents, e.g., by training with existing documents, could have further improved the quality of the results. The hardware specifications prevented the use of larger models that could have potentially improved

the accuracy and quality of the extractions. Due to the large data volume of over 5,000 documents, it is not possible to perform a comprehensive review of all the data transferred. Therefore, 30 representative documents were selected to thoroughly assess accuracy and completeness. It is possible that potential errors or problems may have remained undetected in this sample.

The generalizability of our results is limited, in particular by the pre-processing requirements of the documents and the given data model, which is represented as a one-dimensional table without any relationships and constraints. Due to the small number of pages per document, the context limit was set to less than 8,000 tokens, which was sufficient for most documents, but could lead to difficulties with longer texts. For large documents, the implementation of RAG could be considered to allow more efficient processing of extensive textual data. The optimizations developed as part of this work, particularly in the area of prompt engineering and data pre-processing, are therefore not fully transferable to other use cases.

6. Conclusion and Outlook

This work proposes a procedure for an LLM-based ETL process, which was successfully implemented to investigate the potential of LLMs in structuring heterogeneous data. The results of the case study demonstrate that open-source LLMs can efficiently transfer unstructured documents into a structured target format, achieving a success rate of 96% when transferring over 5,000 documents into a relational database (RQ1). A brief representative comparison with test documents reveals that *GPT-4o mini* is about five times faster in token generation and, consequently, in total runtime [7]. The optimizations carried out, particularly in the areas of prompt engineering and post-processing, have made a significant contribution to increasing the quality of the results (RQ3). Our case study shows that successful LLM-supported data structuring and integration is based on (i) high data quality through standardized formats, cleaning and post-processing; (ii) methodical control of the output via parameter fine-tuning, output templates, and automated correction loops; and (iii) a balanced ratio of result quality and runtime through careful model and hardware selection (RQ4). Although the results are promising, challenges such as processing longer documents and adapting to more complex data models remain.

The outlook for future work indicates numerous approaches to extend the results of this work, including the use of fine-tuning and larger models, which could further improve the quality of the results. Alternative open-source models (e.g., Mistral) could provide new insights and help answer RQ2 in addition to our comparisons. Similarly, applying the developed LLM-based ETL process to more complex use cases and data models could provide valuable insights into the generalizability. The application of RAG to the processing of extensive text documents emerges as a promising approach that merits further exploration. In summary, the LLM-based ETL process outlined in this work successfully demonstrates the potential of LLMs in structuring and integrating heterogeneous data. Despite certain limitations, such as model selection and generalizability to disparate use cases, the case study demonstrates the capacity of LLMs to reliably convert unstructured documents into a structured format. The approaches developed and insights gained in this work are transferable to other domains and use cases, providing a valuable basis for future developments and research in the field of LLM-based data structuring.

References

[1] Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., et al.: Phi-3 technical report: A highly capable language model locally on your phone. In: *arXiv preprint arXiv:2404.14219* (2024).

[2] Ågerfalk, P. J., Conboy, K., Crowston, K., Eriksson Lundström, J., Jarvenpaa, S. L., Ram, S., and Mikalef, P.: Artificial intelligence in information systems: State of the art and research roadmap. In: Association for Information Systems. 2022.

[3] Ajith, A., Pan, C., Xia, M., Deshpande, A., and Narasimhan, K.: InstructEval: Systematic Evaluation of Instruction Selection Methods. In: *Findings of the Association for Computational Linguistics: NAACL 2024*. Mexico City, Mexico, 2024, pp. 4336–4350.

[4] Anu Mohan, M., Ashok, K., Shaikh, M., Dhanush, Y., and Vishwakarma, Y.: Data Integration and Transformation using Large Language Models. In: *Grenze International Journal of Engineering and Technology* (2023), pp. 1644–1649.

[5] Anuradha, I., Mitkov, R., Nahar, V., et al.: Evaluating of Large Language Models in Relationship Extraction from Unstructured Data: Empirical Study from Holocaust Testimonies. In: *14th International Conference on Recent Advances in Natural Language Processing*. 2023, pp. 117–123.

[6] Arora, S., Yang, B., Eyuboglu, S., Narayan, A., Hojel, A., Trummer, I., and Ré, C.: Language Models Enable Simple Systems for Generating Structured Views of Heterogeneous Data Lakes. In: *Proc. VLDB Endow.* 17.2 (Oct. 2023), pp. 92–105.

[7] Bongertmann, H., Nast, B., Griesch, L., Rotzoll, H., and Sandkuhl, K.: *Dataset for "Large Language Models for Structuring and Integration of Heterogeneous Data"*. Zenodo, 2025. URL: https://doi.org/10.5281/zenodo.14779109.

[8] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1877–1901.

[9] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. In: *arXiv preprint arXiv:2407.21783* (2024).

[10] Dwivedi, Y. K., Hughes, L., Ismagilova, E., Aarts, G., Coombs, C., Crick, T., Duan, Y., Dwivedi, R., Edwards, J., Eirug, A., et al.: Artificial Intelligence (AI): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. In: *International Journal of Information Management* 57 (2021), p. 101994.

[11] Floridi, L. and Chiriatti, M.: GPT-3: Its Nature, Scope, Limits, and Consequences. In: *Minds and Machines* 30.4 (2020), pp. 681–694.

[12] Haslhofer, B. and Klas, W.: A survey of techniques for achieving metadata interoperability. In: *ACM Computing Surveys (CSUR)* 42.2 (2010), pp. 1–37.

[13] Huang, J., Yang, D. M., Rong, R., Nezafati, K., Treager, C., Chi, Z., Wang, S., Cheng, X., Guo, Y., Klesse, L. J., et al.: A critical assessment of using ChatGPT for extracting structured data from clinical notes. In: *npj Digital Medicine* 7.1 (2024), p. 106.

[14] Huang, W., Abbeel, P., Pathak, D., and Mordatch, I.: Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 9118–9147.

[15] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[16] Maragno, G., Tangi, L., Gastaldi, L., and Benedetti, M.: Exploring the factors, affordances and constraints outlining the implementation of Artificial Intelligence in public sector organizations. In: *International Journal of Information Management* 73 (2023), p. 102686.

[17] Omar, M. A.: Measurement of ChatGPT Performance in Mapping Natural Language Speficaction into an Entity Relationship Diagram. In: *2023 IEEE 11th International Conference on Systems and Control (ICSC)*. IEEE. 2023, pp. 530–535.

[18] Remadi, A., El Hage, K., Hobeika, Y., and Bugiotti, F.: To prompt or not to prompt: Navigating the use of large language models for integrating and modeling heterogeneous data. In: *Data & Knowledge Engineering* 152 (2024), p. 102313.

[19] Rimban, E. L.: Challenges and limitations of ChatGPT and other large language models. In: *International Journal of Arts and Humanities* 4.1 (2023), pp. 147–152.

[20] Rochan, A., Sowmya, P., and Daniel, D. A. J.: Large Language Model Based Document Query Solution Using Vector Databases. In: *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*. IEEE. 2024, pp. 1–5.

[21] Schaefer, C., Lemmer, K., Samy Kret, K., Ylinen, M., Mikalef, P., and Niehaves, B.: Truth or dare?–how can we influence the adoption of artificial intelligence in municipalities? In: *54th Hawaii International Conference on System Sciences* (2021).

[22] Schultenkämper, S. and Bäumer, F. S.: Structured Knowledge Extraction for Digital Twins: Leveraging LLMs to Analyze Tweets. In: *International Conference on Innovations for Community Services*. Springer. 2024, pp. 150–165.

[23] Sharma, A., Li, X., Guan, H., Sun, G., Zhang, L., Wang, L., Wu, K., Cao, L., Zhu, E., Sim, A., et al.: Automatic data transformation using large language model-an experimental study on building energy data. In: *2023 IEEE International Conference on Big Data (BigData)*. IEEE. 2023, pp. 1824–1834.

[24] Siqueira De Cerqueira, J. A., Dos Santos Althoff, L., Santos De Almeida, P., and Dias Canedo, E.: Ethical perspectives in ai: A two-folded exploratory study from literature and active development projects. In: *54th Hawaii International Conference on System Sciences* (2021).

[25] Vijayan, A.: A prompt engineering approach for structured data extraction from unstructured text using conversational LLMs. In: *Proceedings of the 2023 6th International Conference on Algorithms, Computing and Artificial Intelligence*. 2023, pp. 183–189.

[26] Wang, C., Teo, T. S., and Janssen, M.: Public and private value creation using artificial intelligence: An empirical study of AI voice robot users in Chinese public sector. In: *International Journal of Information Management* 61 (2021), p. 102401.

[27] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24824–24837.

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

[29] Wirtz, B. W., Weyerer, J. C., and Geyer, C.: Artificial intelligence and the public sector—applications and challenges. In: *International Journal of Public Administration* 42.7 (2019), pp. 596–615.

[30] Wretblad, N. and Gordh Riseby, F.: *Bridging language & data: Optimizing text-to-sql generation in large language models (Linköpig Universtity, Master's Thesis)*. 2024.

[31] Zhao, W., Chen, Q., and You, J.: LlmRe: A zero-shot entity relation extraction method based on the large language model. In: *7th International Conference on Electronic Information Technology and Computer Engineering*. 2023, pp. 475–480.