Advanced Methods of AI

Bank Statement Aggregation

Research Proposal

- Submitted By -

Leonard Haddad
leonard.haddad@uni-bremen.de

Arthur Belousov
belousov@uni-bremen.de

Zakaria Ouaddi
zak_oua@uni-bremen.de

Mohamed Amine Kina
mkina@uni-bremen.de

March 13, 2025

Contents

1 Cognitive Abilities **1**

2 Challenges **2**
2.1 Structural Variability, Quality and Layout Inconsistency within Bank-Statements 2
2.2 Complex Data Fields 3
2.3 Language and Character Support 4
2.4 Human Intervention and Error 4

3 Related Work **4**
3.1 Transformer Architectures and *“Attention Is All You Need”* 4
3.2 OCR Frameworks 5
3.3 An Efficient Method to Extract Data from Bank Statements Based on Image-Based Table Detection 6
3.4 Extracting Financial Data from Unstructured Sources: Leveraging Large Language Models 7

4 Prototype System Architecture **8**

5 Limitations **10**

Bibliography **13**

Abstract

Financial institutions frequently receive customer bank statements in paper or scanned PDF form. Manual data extraction remains time-consuming, prone to human error, and difficult to scale.
The need for a robust, automated solution that can extract dates, transaction amounts, and descriptive text from variously formatted statements is thus paramount.
With the rise of LLMs and the transition towards AI-based solutions, such tasks need no longer be done by humans. In this paper, we propose the cognitive architecture of an AI agent capable of performing the task of digitizing bank transfer documents. We discuss various approaches for implementing such an agent in practice, then demonstrate the credibility of our claims with our own prototype.

1 Cognitive Abilities

Before introducing the challenges faced for such an AI agent, we analyzed the core cognitive abilities the agent would need:

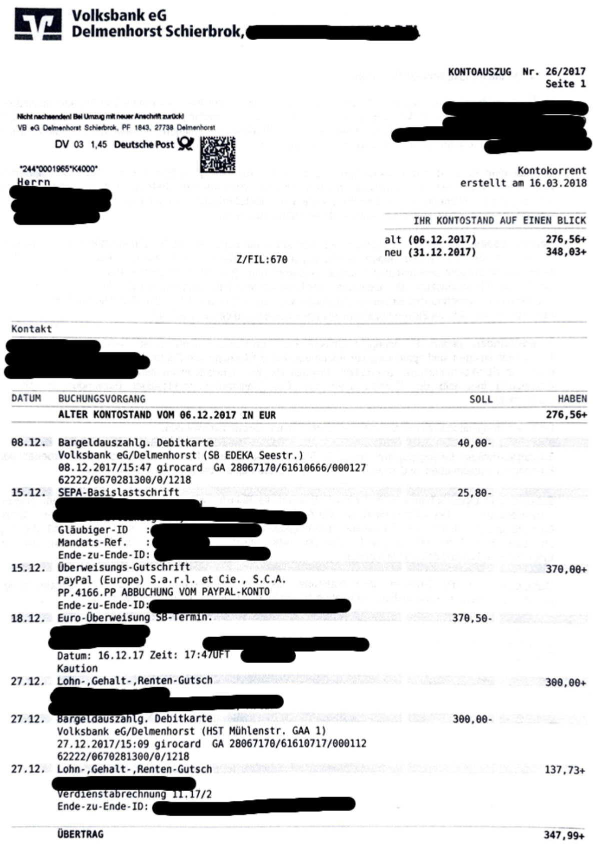
- Perception: the cognitive architecture requires a way of reading bank statement documents, thus the core ability of perceiving these using vision is paramount.
- Memory: To understand and reason with the data extracted from the bank statement documents, the architecture requires some form of short-term working memory for contextual understanding of the text. Furthermore, to reason correctly, the architecture needs some form of long term semantic memory based on which the reasoning can be performed. Without semantic language understanding, the agent would be unable to perform reliable reasoning.
- Reasoning: In addition to memory, the agent needs the ability to reason with the text it reads from the document. In the proposed architecture, we want to use the agent to extract formatted data from the text it reads, thus it needs to support logical inference reasoning to determine which parts of the text are relevant.

With these abilities, the agent should be able to read data from a given PDF file, reason with it and return formatted data in any desired format. This data could then be loaded into a database as required, and passed onto the consulting firms. These abilities do pose a challenge, however, due to the complex nature of the problem. In the next chapter, we glance over the challenges that face the agent and suggest ways of overcoming each.

2 Challenges

2.1 Structural Variability, Quality and Layout Inconsistency within Bank-Statements

Banks worldwide generate statements in a wide variety of layouts, differing in design, format, color, text size, and fonts. With technological advancements, particularly in the banking sector, bank statements are now typically available online for printing or sent by post as a collective statement. These statements often follow a standardized layout, with only marginal differences across different banks, as illustrated in Figure 1.



Volksbank eG
Delmenhorst Schierbrok, [redacted]

KONTOAUSZUG Nr. 26/2017
Seite 1

[redacted]

Nicht nachsenden! Bei Umzug mit neuer Anschrift zurück!
VB eG Delmenhorst Schierbrok, PF 1843, 27738 Delmenhorst
DV 03 1,45 Deutsche Post

*244*0001965*K4000*
Herrn
[redacted]

Kontokorrent
erstellt am 16.03.2018

IHR KONTOSTAND AUF EINEN BLICK

alt (06.12.2017)	276,56+
neu (31.12.2017)	348,03+

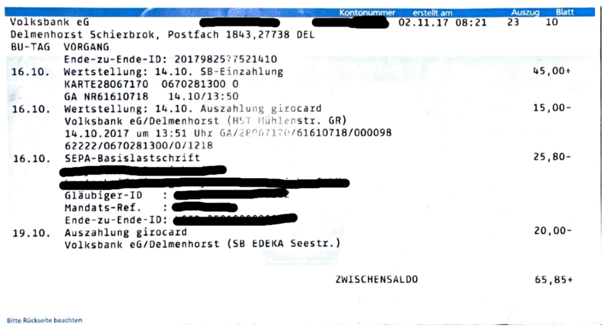
Z/FIL:670

Kontakt
[redacted]

DATUM	BUCHUNGSVORGANG	SOLL	HABEN
	ALTER KONTOSTAND VOM 06.12.2017 IN EUR		276,56+
08.12.	Bargeldauszahlg. Debitkarte Volksbank eG/Delmenhorst (SB EDEKA Seestr.) 08.12.2017/15:47 girocard GA 28067170/61610666/000127 62222/0670281300/0/1218	40,00-	
15.12.	SEPA-Basislastschrift [redacted] Gläubiger-ID : [redacted] Mandats-Ref. : [redacted] Ende-zu-Ende-ID: [redacted]	25,80-	
15.12.	Überweisungs-Gutschrift PayPal (Europe) S.a.r.l. et Cie., S.C.A. PP.4166.PP ABBUCHUNG VOM PAYPAL-KONTO Ende-zu-Ende-ID: [redacted]		370,00+
18.12.	Euro-Überweisung SB-Termin. [redacted] Datum: 16.12.17 Zeit: 17:47UFT [redacted] Kaution	370,50-	
27.12.	Lohn-,Gehalt-,Renten-Gutsch [redacted]		300,00+
27.12.	Bargeldauszahlg. Debitkarte Volksbank eG/Delmenhorst (HST Mühlenstr. GAA 1) 27.12.2017/15:09 girocard GA 28067170/61610717/000112 62222/0670281300/0/1218	300,00-	
27.12.	Lohn-,Gehalt-,Renten-Gutsch [redacted] Verdienstabrechnung 11.17/2 Ende-zu-Ende-ID: [redacted]		137,73+
	ÜBERTRAG		347,99+

Figure 1: Example of a collective bank statement

However, individual bank statements can still be obtained upon request from a bank statement machine at a local bank branch. These are typically limited to a specific number of transactions within a given period and often follow a different layout, such as form, coloring and font details, as from the collective or online versions. Such an exmaple can be seen in Figure 2.
The choice between different types of bank statements largely depends on personal preference and specific needs. While collective statements provide a comprehensive overview of transactions, individual statements may be more convenient for retrieving recent activities quickly. Despite variations in structure and format, both versions serve essential roles in financial record-keeping. Furthermore, beyond the examples presented here, it is evident that numerous other structural and formatting nuances exist across different banks, such as different date formats, currency notations or numerical representations.



```
                                               Kontonummer   erstellt am         Auszug  Blatt
Volksbank eG                     [redacted]    [redacted]    02.11.17 08:21      23      10
Delmenhorst Schierbrok, Postfach 1843,27738 DEL
BU-TAG  VORGANG
        Ende-zu-Ende-ID: 2017982577521410
16.10.  Wertstellung: 14.10. SB-Einzahlung                                          45,00+
        KARTE28067170  0670281300 0
        GA NR61610718   14.10/13:50
16.10.  Wertstellung: 14.10. Auszahlung girocard                                    15,00-
        Volksbank eG/Delmenhorst (HST Mühlenstr. GR)
        14.10.2017 um 13:51 Uhr GA/28067170/61610718/000098
        62222/0670281300/0/1218
16.10.  SEPA-Basislastschrift                                                       25,80-
        [redacted]
        Gläubiger-ID    : [redacted]
        Mandats-Ref.    : [redacted]
        Ende-zu-Ende-ID: [redacted]
19.10.  Auszahlung girocard                                                         20,00-
        Volksbank eG/Delmenhorst (SB EDEKA Seestr.)

                                                        ZWISCHENSALDO               65,85+

Bitte Rückseite beachten.
```

Figure 2: Example of a single bank statement

While first-hand text-based PDF files are typically easy and safe to process, the same cannot be said for physically generated „scans“. These are susceptible to various physical and environmental influences, such as the degradation of paper over time or material deterioration, where paper loses color due to aging. Additionally, the scanning process itself plays a crucial role in the final quality of a scanned image bank statement. Factors such as the use of a low-quality scanning device or unintended movement during scanning can introduce noise, blurriness, or distortions within the resulting PDF or image.

The presented formatting variations can pose a challenge for traditional OCR systems in accurately parsing content, as their performance heavily depends on the quality of images and files, as well as the preprocessing steps undertaken to optimize readability for the OCR engine. In particular, the study [1] on degraded images in the context of OCR has demonstrated that factors such as aging, physical degradation, photocopying artifacts, and embedded noise or distortions can significantly increase classification error rates. Consequently, maintaining sufficient accuracy with this approach may require considerable implementation and maintenance efforts.
Thus different forms of scans require a flexible model, that can handle these obstacles altogether.

2.2 Complex Data Fields

Transaction entries often extend across multiple lines, incorporating partial or interconnected textual descriptions, numerical values, addresses, or references. Extracting these fields reliably requires advanced language understanding to ensure accurate data association. The misalignment of extracted information, such as mistakenly attributing a value to the wrong transaction or misinterpreting textual descriptions, can lead to significant errors.
One of the critical challenges is preserving the correct association between a text line and its corresponding numerical value(s). A failure in this process could distort the final dataset, potentially resulting in severe consequences. One instance would be if a misinterpretation occurs during the extraction of a payment record within a bank statement, where a correctly recorded amount of 300 Euros might be mistakenly captured as 3 Euros (from a different numerical field within the same row) in the database. Such errors could lead to customer disputes, increased manual correction efforts, and operational inefficiencies.
Additionally, bank statements often contain both credit and charge statements, where the differentiation between debits and incoming payments must be correctly identified.

Thus, to mitigate this, a combination of visual and contextual understanding within the extraction is highly beneficial.

2.3 Language and Character Support

Bank statements may contain regional or special characters (e.g., German umlauts or other language-specific symbols). Many common OCR frameworks struggle with these characters, leading to misinterpretations or incomplete extractions, when they have not specifically be trained on them [2]. This issue becomes even more pronounced when processing statements from different countries, where varying languages and formatting conventions further complicate recognition (also see 2.1)
To ensure accurate extraction, broad multilingual support is essential, but also the ability to interpret the extracted text, when facing unfamiliar language structures.

2.4 Human Intervention and Error

In many domains, modern software is increasingly designed to operate autonomously with minimal human intervention. Studies, such as [3], have demonstrated that integrating AI into the workplace can help to significantly reduce human errors. Manual data entry is inherently prone to transcription errors and often leads to substantial labor costs. To ensure both efficiency and accuracy, automated systems could help in processing PDFs and images into structured digital datasets are essential, thereby minimizing reliance on human intervention.
This is particularly critical when handling sensitive data, where maintaining accuracy and operational efficiency requires a reliable and seamless solution. Consequently, the adoption of AI is pivotal in streamlining workflows, reducing errors, and alleviating the burden on human resources. However, selecting an appropriate AI system is crucial, as its effectiveness depends on factors such as contextual adaptability, interpretability, and reliability, as stated in previous sections (see 2.3 and 2.2)

3 Related Work

3.1 Transformer Architectures and *"Attention Is All You Need"*

The field of AI experienced a major breakthrough in 2017 with the release of the paper "Attention Is All You Need"[4], which introduced a new architecture called the Transformer. This model relies on self-attention, a mechanism that enables focusing on multiple parts of an input sequence when processing a word. Unlike previous models, the Transformer efficiently captures relationships between words, particularly long-range dependencies, by processing all tokens in parallel rather than sequentially. This is possible due to its parallel processing capability, which allows it to process all words in an input simultaneously. In contrast, models like RNNs and CNNs process words sequentially, making them less efficient and more computationally expensive, especially as input size increases.

A key improvement of self-attention is its ability to model relationships between words. This is achieved by computing attention scores, which determine how strongly words relate to each other. Each head in Multi-Head Attention specializes in different relationships, which could be instances such as long-range dependencies, subject-object relations, or general sentence structure. The higher the attention scores between words, the more strongly the model associates them in the given context. The formula can be seen in 1. Each attention score is computed using the dot product between Query (Q) and Key (K) vectors. The Query Matrix represents the words asking for attention, the Key Matrix represents words providing information, and the Value Matrix contains weighted representations of words that will be used in the final output [4].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where:

- Q is the Query matrix,
- K is the Key matrix,
- V is the Value matrix,
- d_k is the dimensionality of the key vectors,
- softmax ensures that attention scores sum to 1.

The calculated attention scores are normalized using softmax, ensuring they sum to 1 across all words in the sequence. A higher score means greater relative importance in the attention mechanism. All of it is done parallel for each Attention Head, allowing the model to learn multiple relationships at once [4]. Also multiple sentences can be processed in batches, which makes this Architecture highly efficient in use.

This blueprint forms the foundation of modern large language models (LLMs) such as GPT-3, GPT-4, and specialized models like LayoutLM and Donut for document comprehension. These systems excel in capturing long-range dependencies and contextual relationships in textual and visual data, which can also be beneficial for extracting information from banking statements. Notably, newer versions of GPT-4 support GPT Vision, an integrated capability that enables processing and analyzing images alongside text [5], effectively combining image extraction and interpretation. Since 2022, OpenAI has introduced an interactive chat-based variant of the Generative Pre-Trained Transformer, known as ChatGPT, which has gained widespread adoption across various domains, including education, healthcare, and software engineering, making it accessible to anyone with an internet connection [6].

3.2 OCR Frameworks

Optical Character Recognition (OCR) is a form of artificial intelligence that has been widely used in various domains of software development, including banking, healthcare, and robotics. Its primary function is to extract information, typically in the form of text, from handwritten documents, photocopies, and other types of documents. Technically, OCR is a computerized process that converts text from various sources into a digital format [7].

However, OCR algorithms usually require multiple pre-processing steps to extract text efficiently. These steps include file conversion, preprocessing (such as binarization, filtering, noise removal, and skew correction), text and character segmentation, feature extraction, and sometimes post-processing to correct errors made by the OCR engine [7].
Over time, several OCR engines have been developed. One of the most widely used is „Tesseract“, a lightweight OCR framework that supports multiple languages [7]. More advanced OCR frameworks, such as „PaddleOCR“, offer improved accuracy, particularly for complex images, but at the cost of increased computational requirements. This higher computational demand arises from its reliance on deep learning models and GPU acceleration, leading to longer processing times [8].
While frameworks like **Tesseract** and **PaddleOCR** are powerful and widely adopted, they often require extensive *pre- and post-processing* to handle multi-column or tabular layouts, such as those found in bank statements. Additionally, due to their dependency on image quality, structured text arrangements, and computational resources, OCR-based approaches may not always provide a fully reliable or efficient solution for complex text extraction tasks, especially when the image quality is low or contains high amount of noise[7].
Commercial Document Suites (e.g., ABBYY FineReader, Kofax) are sophisticated solutions with template-based extraction features. They can be costly and require specialized licensing.

3.3 An Efficient Method to Extract Data from Bank Statements Based on Image-Based Table Detection

The extraction of content from bank statements is not a novel concept, as research has demonstrated. Studies such as [8] have shown that extracting transaction data from bank statements is a feasible approach. They successfully extracted financial transaction data from bank statements using image-based table detection from scanned or digital PDFs and subsequently converted the extracted data into structured spreadsheets. This closely aligns with our project's goals and challenges (see 2).
In summary, the authors adopted a computer vision approach utilizing „OpenCV“, combined with the OCR framework „Tesseract“ for transaction data extraction. Additionally, they employed Python libraries such as „pdf2image“ for converting PDFs to images and „pandas“ for structuring the extracted data into a spreadsheet format.
Through a series of processing steps, including preprocessing to convert PDFs into images, binarization to enhance table visibility, table detection using OpenCV, and text extraction with Tesseract OCR, they successfully extracted dates, financial values, and corresponding descriptions from table cells. The final output was converted into an Excel format using Pandas.
Overall, their method achieved an accuracy of over 93%. However, the extraction of transaction descriptions had a lower accuracy of 83.5%, primarily due to the presence of special characters [8].
Despite successfully implementing the extraction using the aforementioned technologies, they also encountered several challenges. One significant issue was related to OCR accuracy, as text recognition for transaction descriptions proved to be less reliable, particularly when special characters were involved. Additionally, distorted or skewed tables required adjustments in row and column detection to ensure accurate data extraction.
In conclusion, the results presented in [8] are highly promising, demonstrating that such

an extraction process can be achieved with the appropriate tools and frameworks. However, considering the limitations of the employed technology and the encountered challenges, particularly the need for a high accuracy in data extraction, it is essential to explore further advancements that could mitigate the weaknesses of this approach. The authors also suggested the possibility of improving OCR performance, by adjusting configurations and pre-processing techniques, or by adopting alternative AI-based methods for more robust extraction.

3.4 Extracting Financial Data from Unstructured Sources: Leveraging Large Language Models

Another project with a similar approach was conducted utilizing the capabilities of Large Language Models (LLMs). The study [9] focused on automating financial data extraction from unstructured sources using LLMs. Unlike the previously presented work (see 3.3), this study built a framework specifically designed to process financial documents, particularly financial reports.
For this purpose, they used GPT-4, Claude 2, and Gemini as test LLMs for text extraction and analysis. Their objective was to understand document structures, recognize key financial terms, and extract relevant numerical data. The processing workflow involved converting PDFs into machine-readable text using tools such as PyPDF2, Tika, and pdfminer, as their approach was solely focused on text extraction from PDFs. To store the extracted data, they utilized a PostgreSQL database.
To improve the accuracy of extracted data, prompt engineering was applied to instruct the LLMs effectively. The authors followed a structured methodology that included designing precise instructions for the models to follow, incorporating examples where needed to enhance accuracy, and formulating detailed step-by-step prompts that facilitated logical reasoning in complex extraction tasks.
The study achieved an overall accuracy of 96.1 percent, and after refining the prompts, the accuracy for key financial indicators reached 100 percent. Large-scale tests were conducted on over 4,000 financial reports, yielding an overall accuracy rate of 96 percent. The results also demonstrated significant efficiency improvements, as the LLM framework performed extraction tasks nine to twenty-five times faster than manual processing. While manual extraction required approximately 200 minutes, the LLM-based method completed the task in just eight minutes [9]. Among the tested LLMs, GPT-4 exhibited the highest accuracy, while Gemini performed the worst, achieving an accuracy rate of only 69 percent. Based on these results, the study suggests that GPT-4 is a viable choice for such tasks, even for slightly different use cases.
Despite these promising results, the authors encountered several challenges. One issue involved the misidentification of numerical units, which was later mitigated by refining the prompt design. Another challenge related to processing inefficiencies in LLMs, particularly when handling large and unstructured PDFs, which required refining search strategies to reduce unnecessary computations. These difficulties highlight certain limitations of LLMs, particularly in cases where transformers are not inherently optimized for processing large volumes of data efficiently. Addressing these challenges is essential when considering the application of such a framework.
Privacy and security concerns were also highlighted as a limitation, especially when using LLMs on cloud-based platforms for processing sensitive financial data. The high cost of accessing advanced models such as GPT-4 API was another challenge, particularly for

large-scale operations. The study suggests that more affordable models, such as GPT-3.5, could serve as an alternative, offering a trade-off between cost and performance [9].

To summarize, the study provided valuable insights into the possible application of LLMs for financial data extraction. The findings suggesting that GPT-4 could also be considered for extracting data from bank statements, particularly if vision-based capabilities are ensured(see 3.1). However, the question of cost and security should be considered, given that bank statements contain highly sensitive financial data that must be protected from unauthorized access.

4 Prototype System Architecture

The code for the prototype, along with a demo video, can be found on GitHub [10]
For the prototype implementation, we initially attempted OCR using traditional OCR frameworks in Python, namely EasyOCR, PyTesseract, and PaddleOCR. While these frameworks could reliably extract some text, the results were often incorrectly formatted, missing information, or contained misclassified characters.
With the release of Azure OpenAI's GPT-4o with Vision capabilities, we opted to use the LLM for both OCR and data extraction tasks, achieving an average data correctness and completeness of over 90%.
Using the LLM also allowed us to eliminate the lengthy pre-processing steps. With traditional OCR, we first had to split pages into separate images, then check and correct their orientation (PaddleOCR, for instance, required documents to be upright to classify text correctly). Next, we performed binary masking and cropping to enhance text quality before sending the processed text to GPT to correct data inconsistencies.
Replacing traditional OCR with an LLM-based solution enabled us to discard the entire pre-processing module and consolidate OCR, text extraction, and data formatting into a single prompt.
Consequently, we engineered our prompting such that the LLM extracts the text, classifies the bank statements and returns these as an array of JSON objects, which can be parsed in Python and stored in a lightweight Sqlite3 database.
For old-school single-print bank statements (as mentioned in chapter 2 2) this approach worked flawlessly. For modern scanned documents, which are made up of multiple pdf pages, this did not work, as the LLM is incapable of performing OCR directly on PDF files (as these are uploaded as "printed" PDF files containing binary image data). To work around this, we used the Python library "Py2Image" to extract each of the PDF's pages as an image, then sent it to the LLM to perform OCR and data extraction.
Aggregating the data was done natively in Python, where the data from all pages of a PDF file is extracted as a JSON dictionary and saved in the database with a reference to the original PDF file.
To export the data in a format that the consulting company's workers can use, we implemented a CSV export for each PDF document.
Demos of the software execution (both on printed old-school and new documents) can be found in the GitHub repository.
To make the execution of the software as straightforward as possible, we opted for a docker-based environment, which both allows easy scalability, parallelism and prevents operating-system compatibility issues. Furthermore, it allows a simple "one-click-run"

approach for processing a batch of documents.

To allow the LLM to extract data from each page of a given PDF file, we used the following query:

~~~
You are an AI assistant assisting the german bankers in digitizing
scans and faxes of bank transactions. You are provided with the
following  image, which may contain multiple bank transactions.
Return a json response in the following format:

```json
{
    'transactions': [
        {
            'date': 'Transaction date, **always required**.',
            'amount': 'Transaction amount, **always required**.',
            'transaction_text': 'Transaction text, if available.'
        }
    ]
}

IF NOT TRANSACTIONS ARE AVAILABLE, RETURN THE ARRAY:

```json
{
    'transactions': []
}
~~~

# How to respond to this prompt: - response_format: JSON
# The JSON should be parseable using a single json.loads in python.
# RETURN NO FURTHER TEXT, JUST THE JSON.
```

The query is based on the best practices for prompt engineering by OpenAI [11]. The prompt asks the model to extract the data in the exact provided format, return the data **only** in JSON format and provides instructions for what to do if no data is found in the given page.

To prevent missing values in the extracted JSON response, an emphasis is made for each required value (followed by data validation in Python as a last resort).

When combining all the modules, our system has the following architecture:

The "Entrypoint" package (for lack of a better name) contains the core logic which reads in PDF files. The "PDF Processor" package is responsible for extracting PDF metadata, namely the number of pages within the documents, and generating images for each page. The "Azure OpenAI Handler" offers an interface for querying the Azure OpenAI API and retrieving data directly as JSON objects. The "Database Handler" stores data in the SQLite3 database, offers CRUD operations, and allows exporting data when needed. Lastly, the "CSV Handler" allows exporting formatted data for each PDF file.
```

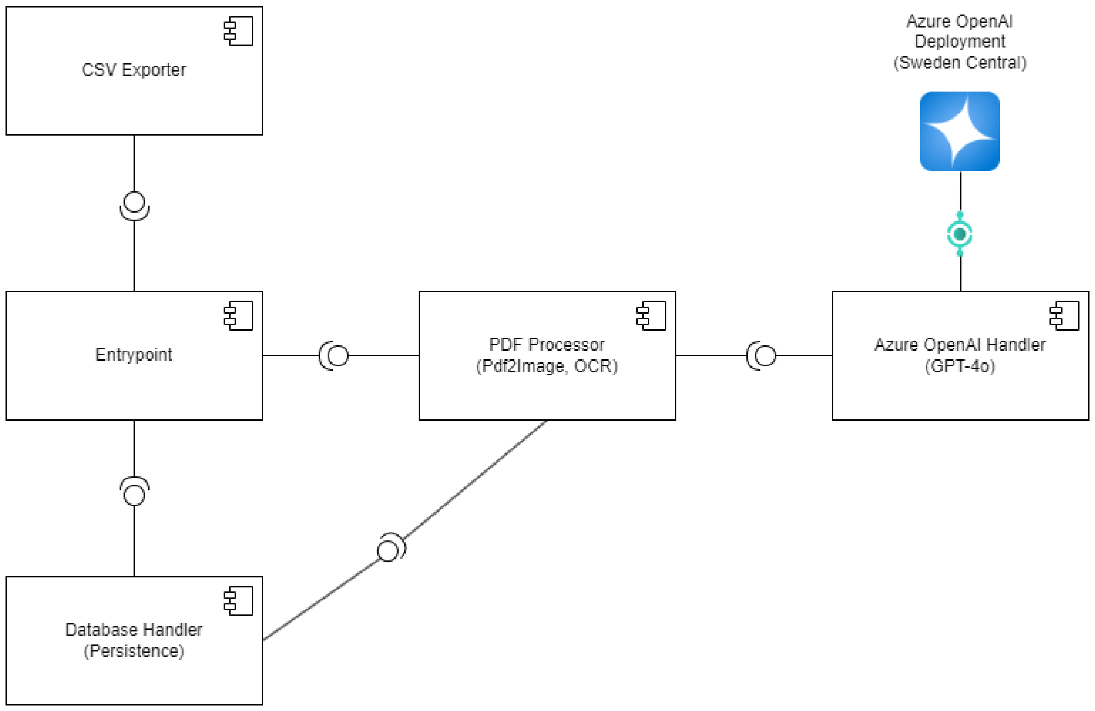

CSV Exporter
Azure OpenAI Deployment (Sweden Central)
Entrypoint
PDF Processor (Pdf2Image, OCR)
Azure OpenAI Handler (GPT-4o)
Database Handler (Persistence)

Figure 3: Bank Statement Aggregator Architecture

5 Limitations

Due to our architecture relying on an LLM to perform OCR and data extraction, the main limitations of the architecture are those of the LLM itself. Since the LLM uses reasoning to grasp the contextual relevance of the data it extracts, and OpenAI's Vision supporting complex tabular formats, the limitations boil down to:

- Limitations in text recognition.
- Limitations in data extraction.
- Limitations in data correctness.

In terms of *text recognition* limitations, the models offered by OpenAI perform OCR more accurately on Latin-based text due to the larger training volume in Latin languages. Languages with more complex alphabets (such as Arabic or Cyrillic-based language) would face challenges during text recognition and thus produce incoherent data, resulting in worse results or missing bank transactions altogether.
For correct *data extraction*, the LLM requires documents that are formatted in some coherent way. While German-based bank records are generally neatly formatted in tabular forms (with/without visible borders), bank statements from other countries are much more chaotically formatted. To correctly perceive, and thus extract, the transactions, the LLM has to recognize, extract and then process the statements as they appear. If the format is confusing, the LLM might get confused, causing data inconsistencies or wrong data extraction.
Lastly, due to the probabilistic nature LLMs, "all text generated by them can be considered a AI hallucination, that just happens to be true a lot of the time" [12]. Consequently,

data generated by the AI can vary in correctness and should be validated before use. Professional ways to mitigate this can include sending requests multiple times and comparing outputs, checking data against known client information or using human agents for validation (a process that can be lengthy and expensive).
Since the consulting companies work with sensitive client info, one might assume that data protection policies (the GDPR and similar) would prove to be the toughest legal limit. However, due to Microsoft's Data Protection Policy regarding GPT Data [13], these concerns are void and the proposed prototype adheres to GDPR policies.

References

[1] R. T. Hartley and K. Crumpton, "Quality of OCR for degraded text images," *CoRR*, vol. cs.DL/9902009, 1999. [Online]. Available: https://arxiv.org/abs/cs/9902009

[2] K. Hamad and M. Kaya, "A detailed analysis of optical character recognition technology," *International Journal of Applied Mathematics Electronics and Computers*, no. Special Issue-1, pp. 244–249, 2016.

[3] A. N. Abbas, C. W. Amazu, J. Mietkiewicz, H. Briwa, A. A. Perez, G. Baldissone, M. Demichela, G. G. Chasparis, J. D. Kelleher, and M. C. Leva, "Analyzing operator states and the impact of ai-enhanced decision support in control rooms: A human-in-the-loop specialized reinforcement learning framework for intervention strategies," 2024. [Online]. Available: https://arxiv.org/abs/2402.13219

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[5] W. Wu, H. Yao, M. Zhang, Y. Song, W. Ouyang, and J. Wang, "Gpt4vis: What can gpt-4 do for zero-shot visual recognition?" 2024. [Online]. Available: https://arxiv.org/abs/2311.15732

[6] S. S. Sohail, F. Farhat, Y. Himeur, M. Nadeem, D. Madsen, Y. Singh, S. Atalla, and W. Mansoor, "Decoding chatgpt: A taxonomy of existing research, current challenges, and possible future directions," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 8, p. 101675, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.jksuci.2023.101675

[7] R. Mittal and A. Garg, "Text extraction using ocr: A systematic review," in *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2020, pp. 357–362.

[8] J. Qian, Y. Ma, C. Lin, and L. Chen, "Accelerating ocr-based widget localization for test automation of gui applications," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3551349.3556966

[9] H. Li, H. H. Gao, C. Wu, and M. A. Vasarhelyi, "Extracting financial data from unstructured sources: Leveraging large language models," *Journal of Information Systems*, pp. 1–22, 08 2024. [Online]. Available: https://doi.org/10.2308/ISYS-2023-047

[10] "Bank Statement Digitizer Prototype on GitHub," last accessed: 15. Febuary, 2025. [Online]. Available: https://github.com/IsraTech-Software/advanced-methods-of-ai

[11] "Best practices for prompt engineering with the OpenAI API | OpenAI Help Center," last accessed: 15. Febuary, 2025. [Online]. Available: https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api

[12] R. Rahi - VP CSS OU Cloud Delivery | Oracle University, “Hallucinations | Fundamentals of Large Language Models | Oracle Cloud Infrastructure Generative AI Professional Certification | Oracle University,” last accessed: 15. Febuary, 2025. [Online]. Available: https://mylearn.oracle.com/ou/course/oracle-cloud-infrastructure-generative-ai-professional/136035/

[13] “Data, privacy, and security for Azure OpenAI Service - Azure AI services | Microsoft Learn,” last accessed: 15. Febuary, 2025. [Online]. Available: https://learn.microsoft.com/en-us/legal/cognitive-services/openai/data-privacy