Extraction of tabular data from annual reports with LLMs

Using in context learning with open source models and RAG

submitted by

Simon Schäfer

Matr.-Nr.: 944 521

Department VI – Informatics and Media
Berliner Hochschule für Technik Berlin

presented Master Thesis
to aquiere the academic degree

Master of Science (M.Sc.)

in the field of

Data Science

Date of submission September 1, 2025

Gutachter

Prof. Dr. Alexander Löser   Berliner Hochschule für Technik
Prof. Dr. Felix Gers   Berliner Hochschule für Technik

Abstract

Content of this thesis is a benchmark on information extraction from PDFs. The focus are annual reports of German companies. Special characteristic of the task is handling hierarchies in tables with financial data to prepare the data for import into a relational database.

The benchmark is composed of three sub tasks and the performance of different open source large language models is tested with different prompting approaches and compared to alternative methods.

This can be seen as a reimplementation study of "Extracting Financial Data from Unstructured Sources: Leveraging Large Language Models" - a paper published by Li et al. (2023). The key differences are the application on German documents using open source large language models.

Zusammenfassung

Gegenstand dieser Arbeit ist ein Benchmark zur Informationsextraktion aus PDF-Dateien. Dabei wird sich auf das Auslesen der Bilanzen und Gewinn- und Verlustrechnungen aus Jahresabschlüssen deutscher Unternehmen beschränkt. Ein besonderer Aspekt der Aufgabe ist die Berücksichtigung der Hierarchie innerhalb der Tabellen, um die Werte einem festen Schema zuzuordnen und so den Import in eine relationale Datenbank vorzubereiten.

Notes

- Qwen 2.5 hat zweiseitige GuV von IBB entdeckt und zur Anpassung der Ground Truth
- Google gemma war mit alter Klassifikation erfolgreich (anderer Prompt, mehr Seiten)

Contents

Chapter 1

Introduction

1.1 Motivation

- market: public administration, companies with data of special requirements for treating (secret and personal data (high risk data)) <– DSGVO, AI act
  - next market for hyper scalers might be public administration with local computing clusters
- whom is it helping
- why now: digital sovereignity, AI act; people want NLP AI products, frameworks get easier
- is the problem easier solvable then years ago? why?

missing law to access digital data and no law to choose the format of the data extensible Business Reporting Language as a standard changing from HGB to IFSR

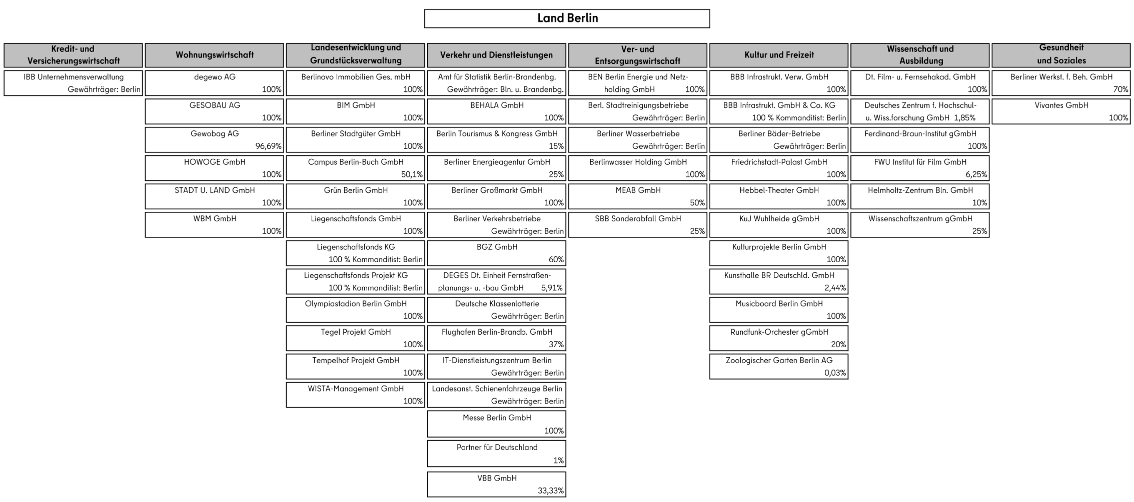
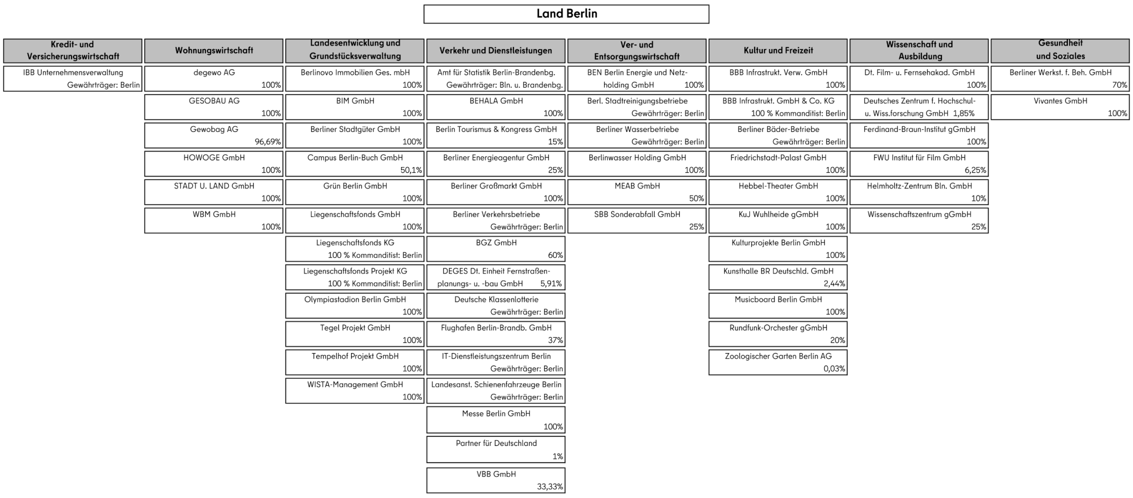


Figure 1.1: Companies Berlin has holds share at



1.2 Objectives

The sixth division at RHvB is auditing the companies Berlin is a stakeholder of. Basic information they have to process are the balance sheets and profit and loss accounting. Those information is provided via their annual reports in form of PDF files. The provided annual reports often differ from the publicly available ones in matter of information granularity and design and are treated as non public information. Automate the extraction of those information would be a good starting point for AI assisted information retrieval from PDFs for the RHvB overall.

It is important to get numeric values totally accurate; numeric values are difficult to handle for langauge models

- special part of big problem? central question
- two sentences: why this problem? new problem or just a part in the big task? hard to solve of straight forward? research or application? what was not done and why?
- building a system? what task to solve? core functionality? typical use cases?

1.3 Methodology (1 p)

- how to solve the problem?
- what foundations to have in mind?
- proceeding?

1.4 Thesis Outline (0.5 p)

1.5 To place in chapters abovehttp://127.0.0.1:29003/rmd_output/4/images/bete

This master thesis is motivated by a use case from practical work at the Berlin court of audit (Rechnungshof von Berlin; RHvB). The auditors often are faced with the problem that they need information that is provided as natural language or in tables inside of unstructured documents, i.e. in PDF files. The goal of this thesis is benchmarking methods for automated information extraction from specific tables from PDF files.

Ideally, the data extraction pipeline is able to autonomously * identify the pages with the tables of interest. * identify the tables of interest on these pages. * extract the information as provided into a structured table (e.g. as JSON, a csv file or HTML code). * transform the data into a given schema, stripping all aggregated values.

It should extract the values without errors. It would be nice if the computation time and energy consumption is as low as possible.

A more realistic approach, that is also beneficial to satisfy the AI Act (keine Entscheidung ohne menschliche Beteiligung), is an assistant system, that helps extracting information. Key features to get the human into the loop already at the step of information extraction for such an assistant might be:

- showing the results together with the systems confidence.
- showing the results next to the values of the source.
- allowing in place adjustments to the extracted data.

A sound decision making is only possible if the information the decision is based on is valid.

1.6 RHvB

- what does the RHvB do
- why is this important
- what does it not do yet (because data source is missing)

1.7 Datenverfügbarkeit

- keine Regelung, in welcher Form der Rechungshof die Daten, die er benötigt, bereitgestellt zu bekommen hat

Das Gesetz zur Förderung der elektronischen Verwaltung (EGovG) wurde erlassen, "um die Verwaltung effektiver, bürgerfreundlicher und effizienter zu gestalten." (BMI, Referat O2, 2013)

§ 12 EGovG

- Vorhaben zur Datenkatalogisierung innerhalb der Verwaltung angestoßen, aber noch nicht richtig gestartet
- Vornehmlich für Bürger*innen Zugang

1.8 Unstrukturierte Daten

- Beispielbilder

1.8.1 Portable Document Format

- print optimized
- Table structure information gets lost
- Bild und Textextract

Chapter 2

Literature review (less than 10 p)

(5 to 10 lines)

- overview of subchapters
- relevance for reader (Gutachter)
- link to previous chapter
- relevant basic tasks

2.1 Basic terms

2.2 Technological topic (related work)

- most important papers
- connection of papers (timeline)
- what used, what not?
- extending existing paper?

2.2.1 Extraction of numeric values

99.5 % or 96 % accuracy for extracting financial data from Annual Comprehensive Financial Reports (Li et al., 2023) In the untabulated test, GPT-4 achieved an average accuracy rate of 96.8%, and Claude 2 achieved 93.7%. Gemini had the lowest accuracy rate at 69%. (ebd.)

Too many hallucinated values when it was NA instead (Grandini et al., 2020)

2.3 optinal more topics like previous

2.4 Summary (0.5 p)

- lessons learned
- link to goal thesis
- link to next chapter

2.5 To place in chapters above

2.6 Table extraction tasks

2.6.1 Difficulties

- Beispielbilder

2.7 Document Extrtaction Process

2.7.1 Document Layout Analysis

An important step in the process of extracting information from documents is to recognize the layout of a document (Zhong et al., 2019).

Getting the order of texts correct align captions to tables and figure identify headings, tables and figures

One of the most popular datasets used for training and benchmarking is PubLayNet (see PubLayNet on paperswithcode.com). It contains over 360_000 document automatically annotated images from scientific articles publicly available on PubMed Central (Zhong et al., 2019, p. 1). This was possible, because the articles have been provided in PDF and XML format. For the annotations most text categories (e.g. text, caption, footnote) have been aggregated into one category. <– is this a problem for later approaches where a visual and textual model work hand in hand to identify e.g. table captions?

Manual annotated datasets often were limited to several hundred pages. Deep learning methods need a much larger training dataset. Previously optical character recognition (OCR) methods were used.

Identify potentially interesting pages with text / regex search. Check if there is a table present on this page.

Object detection

2.7.1.1 Vision Grid Transformer

2.7.2

2.8 Tools

2.8.1 TableFormer

SynthTabNet <– has it: - nested / hierarchical tables, where rows add up to another row? - identifying units and unit cols/rows

Chapter 3

Implementation (max 5p)

3.1 Speedup with vLLM and batching

3.2 Setup (Dockerfile and PV)

Chapter 4

Methods

4.1 Data

- companies Beteiligungsbericht
- number found Jahresberichte
- number used Jahresberichte first rows
- number used Jahresberichte Aktiva Tabellen

4.2 Page identification

The first task to solve, for a fully autonomous solution, is to identify the pages where the tables of interest are located. For benchmarking 74 annual reports from 7 companies have been used. For this benchmark we limit the tables of interest to those that show **Aktiva**, **Passiva** and **Gewinn- und Verlustrechnung**.

In those documents there are 252 pages of interest holding 265 relevant tables. On 13 pages there have been two tables (**Aktiva** and **Passiva**) on a single page. 21 tables are spread over two pages. In 8 documents there have been multiple tables per type of interest, distributed among the three types of tables as following:

type	count
Aktiva	7
GuV	8
Passiva	7

As a baseline a simple regex approach was used.

4.2.1 Baselines

4.2.1.1 Regex based

results potentially dependend on package used for text extraction (Auer et al., 2024, p. 2 f.)

- PyMuPDF
- pypdf
- docling-parse
- pypdfium
- pdfminer.six

pdfminer informs that some pdfs should not be extracted based on their authors will (meta data field)

results dependend on regex pattern

start with pypdf backend and simple regex developed more sophisticated regex based on missed pages

took wrong identified pages as base for a table detection benchmark and n-shot base for llm classification (contrasts)

some tables can't be found without previous ocr; some pages hold image of table and machine readable text

4.2.1.1.1 LLM based

4.2.1.2 Term frequency based

4.2.1.2.1 VLLM based was not implemented

4.3 Table detection

Can be used to narrow down set of possible pages

Can be used to focus only on the table content (measure if correct area was identified would be necessary)

Vision model as baseline

4.3.1 LLM

- table: yes/no
- akiva: yes/no
- multiclass

4.3.2 Vision Model

Yolo

4.3.3 Docling and Co

4.3.3.0.1 VLLM based was not implemented

4.4 Information extraction

4.4.1 Baselines

simple regex?

4.4.2 Simple pipeline

- extract text (if document can't be passed directly)
- query LLM directly

4.4.3 Sophisticated approaches

not implemented

- with pipelines
- Nougat
- maker
- Azure
- docling

Chapter 5

Results

5.1 Page identification

As described in A.2.1 open source libraries have been used to extract the text from the annual reports.

5.1.1 Baseline: Regex

Building a sound regular expression often is an iterative process. In a first approach a very simple one was implemented.

Comparing the differences in the metrics based on the different text extraction libraries it can be said that the extracted text is very similar but not identical. Since the resukts are not depending on the used text extraction library the *exhaustive regex restricted* has only been run with the fast text extraction library *pdfium*. The results of the regex based page identification are presented in the following tables.

- look into details where they differ and if it is because of a line break or whitespace ?

Due to the imbalanced distribution of the classes the accuracy is not a good metric to compare the performance of the different methods. The number of pages of interest is much smaller than the number of irrelevant pages. Therefore, precision, recall and F1 score are presented as well.

The regular expressions can be found in the appendix (see 5.1.1).

Table 5.1: Comparing page identification metrics for different regular expressions for classification task 'Aktiva'

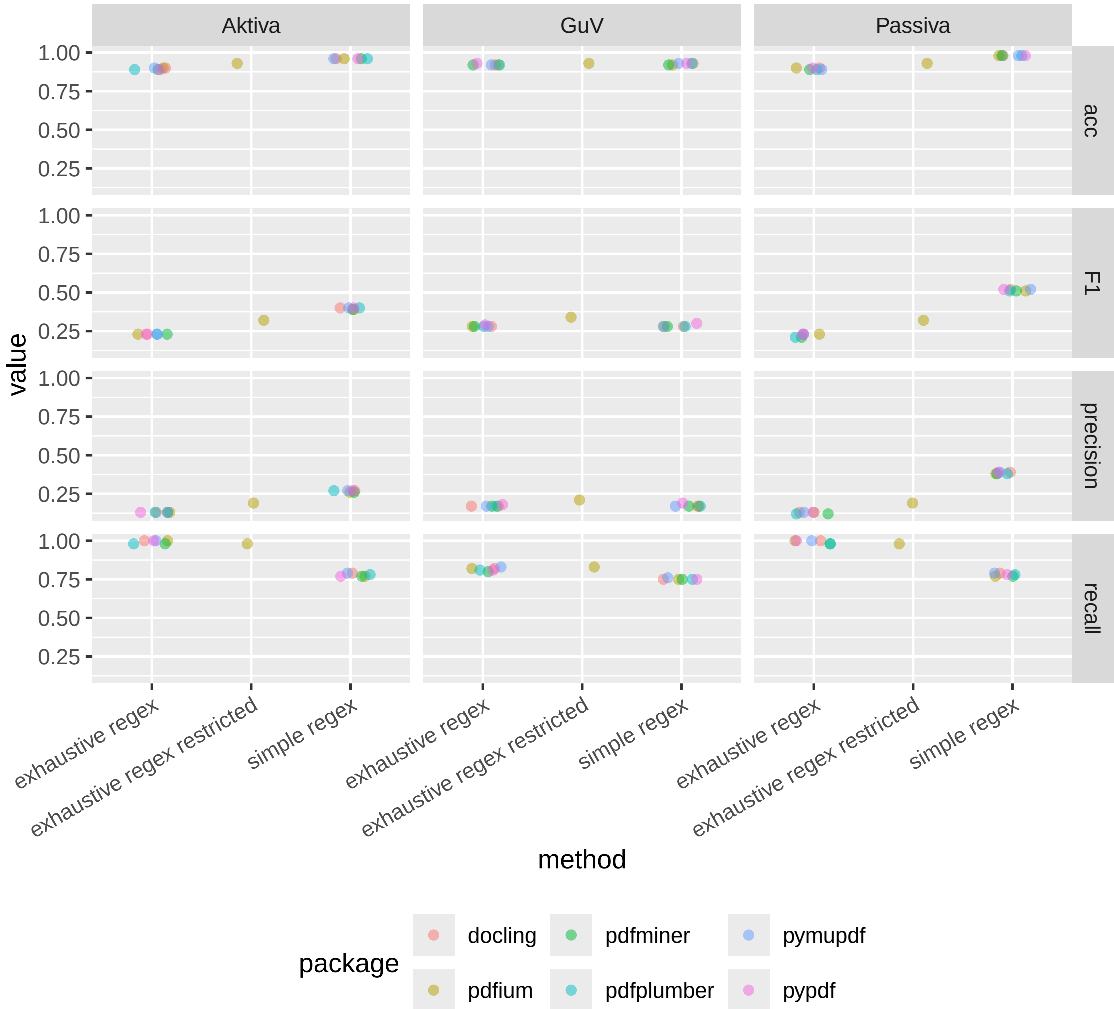
method	stat	precision	recall	F1
simple regex	mean	**0.267**	0.778	**0.397**
simple regex	sd	0.005	0.01	0.005
exhaustive regex restricted	mean	0.19	0.98	0.32
exhaustive regex restricted	sd	NA	NA	NA
exhaustive regex	mean	0.13	**0.993**	0.23
exhaustive regex	sd	0	0.01	0

Table 5.2: Comparing page identification metrics for different regular expressions for classification task 'Passiva'

method	stat	precision	recall	F1
simple regex	mean	**0.385**	0.78	**0.515**
simple regex	sd	0.005	0.009	0.005
exhaustive regex restricted	mean	0.19	0.98	0.32
exhaustive regex restricted	sd	NA	NA	NA
exhaustive regex	mean	0.127	**0.993**	0.223
exhaustive regex	sd	0.005	0.01	0.01

Table 5.3: Comparing page identification metrics for different regular expressions for classification task 'Gewinn- und Verlustrechnung'

method	stat	precision	recall	F1
simple regex	mean	0.173	0.752	0.283
simple regex	sd	0.008	0.004	0.008
exhaustive regex restricted	mean	**0.21**	**0.83**	**0.34**
exhaustive regex restricted	sd	NA	NA	NA
exhaustive regex	mean	0.172	0.815	0.282
exhaustive regex	sd	0.004	0.01	0.004



5.1.2 Advanced techniques

5.1.2.1 Table of Contents understanding

- top 1
- top k
- toc analysis
- cleaned measures

5.1.2.2 Classification with LLMs

5.1.2.2.1 Binary classification

- f1
- multiple models
- best model detail (different methods / settings)

5.1.2.2.2 Multi classification

- f1
- multiple models
- best model detail (different methods / settings)

5.1.3 Comparison

5.1.3.1 F1

5.1.3.2 Energy usage and runtime

5.2 Table extraction

Chapter 6

Discussion

6.1 Not covered

- OCR

Chapter 7

Conclusion

References

Auer, C., Lysak, M., Nassar, A., Dolfi, M., Livathinos, N., Vagenas, P., Ramis, C. B., Omenetti, M., Lindlbauer, F., Dinkla, K., Mishra, L., Kim, Y., Gupta, S., Lima, R. T. de, Weber, V., Morin, L., Meijer, I., Kuropiatnyk, V., & Staar, P. W. J. (2024). *Docling Technical Report.* arXiv. https://doi.org/10.48550/arXiv.2408.09869

BMI, Referat O2 (Ed.). (2013). *Minikommentar zum Gesetz zur Förderung der elektroni- schen Verwaltung sowie zur änderung weiterer Vor- schriften.*

Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: An Overview.* arXiv. https://doi.org/10.48550/arXiv.2008.05756

Li, H., Gao, H. (Harry)., Wu, C., & Vasarhelyi, M. A. (2023). *Extracting Financial Data from Unstructured Sources: Leveraging Large Language Models* [{SSRN} {Scholarly} {Paper}]. Social Science Research Network. https://doi.org/10.2139/ssrn.4567607

Zhong, X., Tang, J., & Yepes, A. J. (2019). *PubLayNet: Largest dataset ever for document layout analysis.* arXiv. https://doi.org/10.48550/arXiv.1908.07836

Chapter A

Appendix

A.1 Local machine

One can find the specifications of the local machine used to run the less computationally demanding tasks below. It is a lightweight laptop device. Its performance cores support hyperthreading and have a clock range between 2.1 and 4.7 GHz. However, due to the flat design, there is little active cooling. Thus, thermal throttling starts rather quickly. It is therefore a reasonable assumption that most locally benchmarked tasks are running at 2.1 GHz. Despite this handicap, it has a sufficiently large RAM of 32 GB and 3 GB of NVMe disk space.

System Details Report

Report details

- **Date generated:** 2025-07-19 13:56:16

Hardware Information:

- **Hardware Model:** LG Electronics 17ZB90Q-G.AD79G
- **Memory:** 32.0 GiB
- **Processor:** 12th Gen Intel® Core™ i7-1260P × 16
- **Graphics:** Intel® Graphics (ADL GT2)
- **Disk Capacity:** 3.0 TB

Software Information:

- **Firmware Version:** A2ZG0150 X64
- **OS Name:** Ubuntu 24.04.2 LTS
- **OS Build:** (null)
- **OS Type:** 64-bit
- **GNOME Version:** 46
- **Windowing System:** Wayland
- **Kernel Version:** Linux 6.11.0-29-generic

Table A.1: Comparing extraction time (in seconds) for different libraries

library	runtime in s
pdfium	**14**
pymupdf	22
pypdf	218
pdfplumber	675
pdfminer	752
docling-parse	1621

A.2 Benchmarks

A.2.1 Text extraction

A basic requirement for all succeeding tasks is, that the text gets extracted from the PDF files. As written in doclings technical report (Auer et al., 2024) the available open source libraries differ in their speed and restrictiveness of licensing. Since there are no benchmark results this report multiple libraries have been tested here.

The benchmark ran on the local machine described in section A.1. There have been 5256 pages to extract the text from.

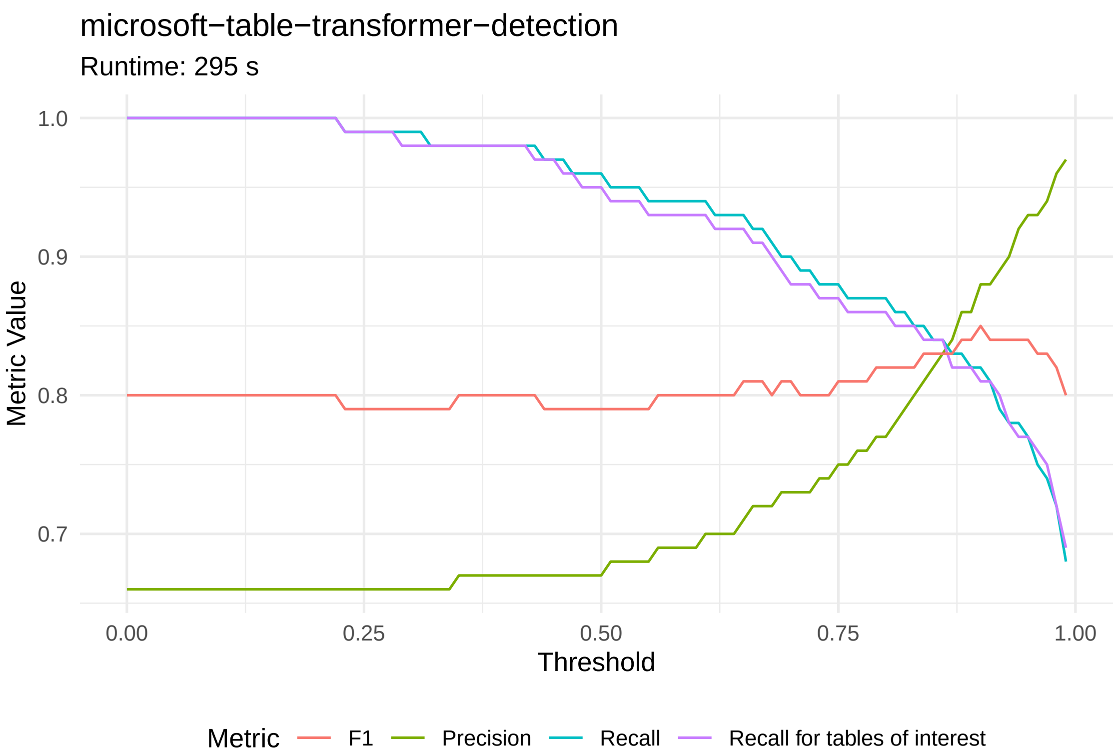
The result of docling-parse is not formated as markdown yet but also just plain text.

For implementation in a system where the text has to get extracted live or frequently the speed of the library might be paramount. But in special cases it can be important to invest more computational power into text extraction if this assures extraction according a more complicated document layout. E.g. some of the tables have been parsed by pdfium in such a manner that first all row descriptors have been extracted (first row) and thereafter all numeric columns (rowwise) ADD REFERENCE / EXAMPLE.

A.2.2 Table detection

- yolo benchmark and table transformer
- skip classification with llm

not so important anymore


microsoft−table−transformer−detection
Runtime: 295 s
1.0
0.9
0.8
0.7
Metric Value
0.00
0.25
0.50
0.75
1.00
Threshold
Metric
F1
Precision
Recall
Recall for tables of interest

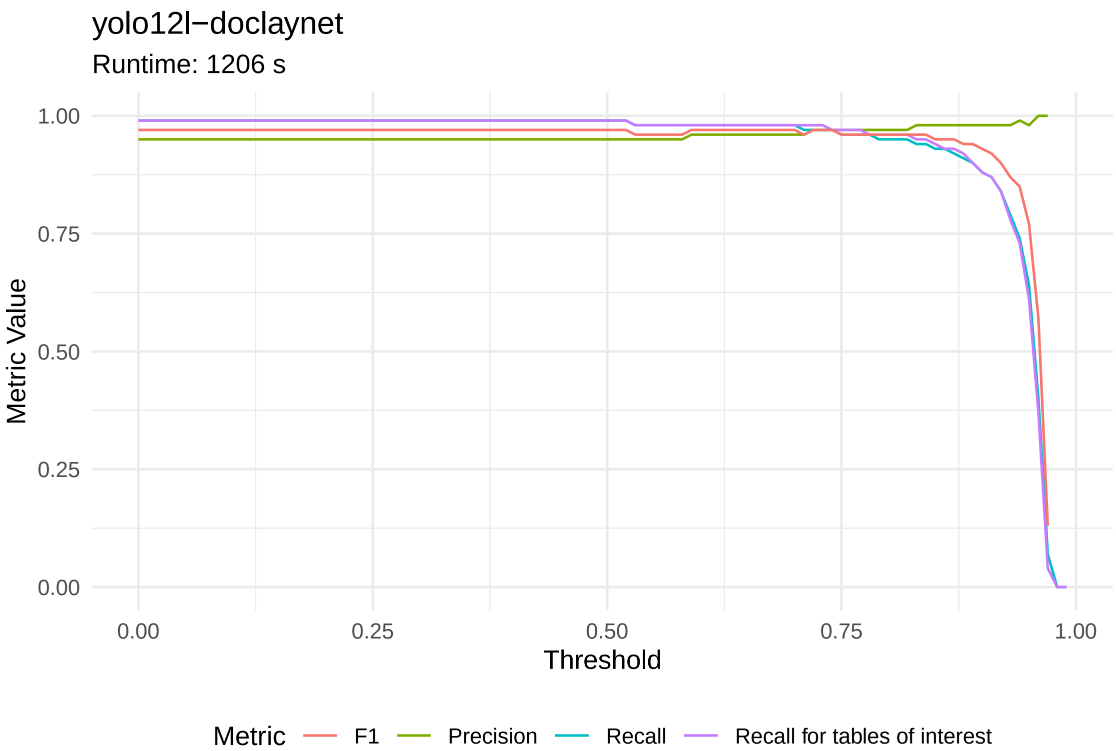
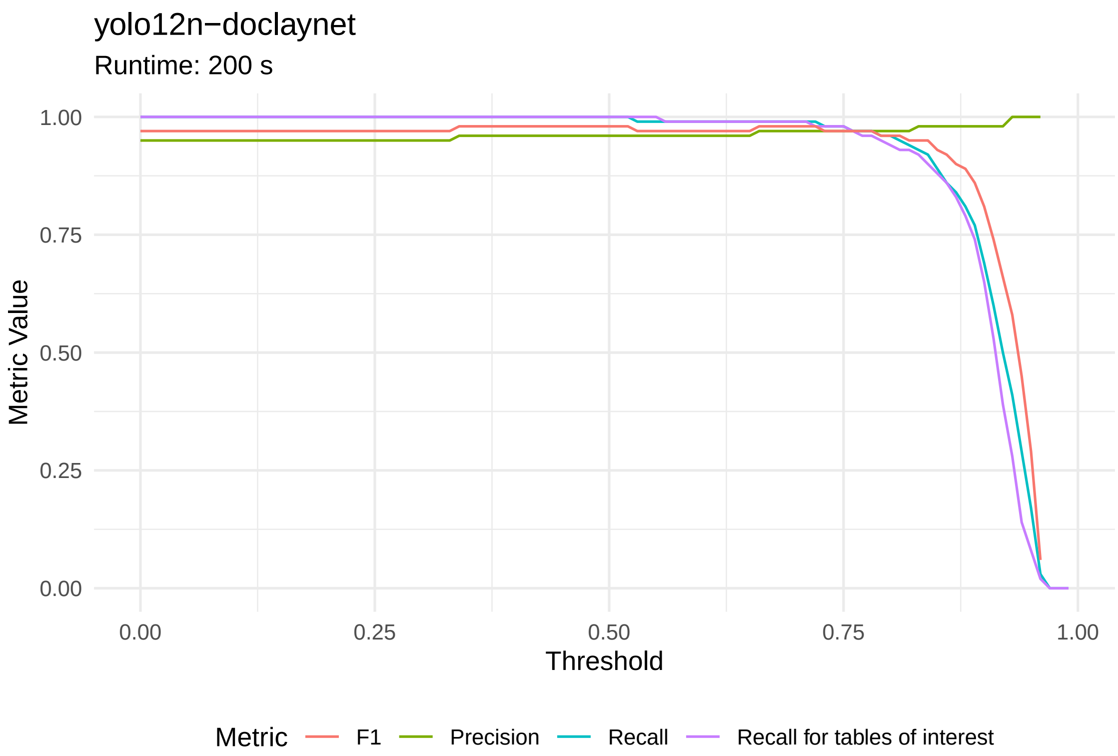

yolo12l−doclaynet
Runtime: 1206 s
1.00
0.75
0.50
0.25
0.00
Metric Value
0.00
0.25
0.50
0.75
1.00
Threshold
Metric
F1
Precision
Recall
Recall for tables of interest

Table A.2: Comparing time (in seconds) for processing ten asset tables using different libraries and approaches

Model parameters (in B)	Transformers	vLLM	vLLM batched
0.5	330	65	NA
3.0	628	130	20
7.0	940	217	30



A.2.3 Large language model process speed

In April 2025 there have been issues with running vllm within the Python framework. Thus the first experiments have been conducted using the transformers library. When the problems of building a working vllm based docker image for the experiments it was measured how long the same task takes with the transformers and the vllm library and how the batched processing competes versus a loop approach. The model family used was Qwen 2.5 Instruct. The task was to extract the assets table for ten real example pages.

Table A.2 shows that the experiments with vllm library run are around four to five times faster. Processing the messages in a batched mode again is six to seven times faster.

The change of the experimental setup from transformers loop-based to vllm batched mode made is possible run the benchmark on whole PDF documents giving a sound estimate of the false positive rate in the page identification task (see section 5.1.2.2). Previous experiments have only been using a subset of pages that have been selected with the baseline regex approach (see section 5.1.1). One can find the former results in section A.2.4.

A.2.4 Table identification with LLMs

A.3 Regular expressions

Here one can find the three regular expressions used for the benchmarks presented in section 5.1.1.

```
simple_regex_patterns = {
    "Aktiva": [
        r"aktiva",
        r"((20\d{2}).*(20\d{2}))"
    ],
    "Passiva": [
        r"passiva",
        r"((20\d{2}).*(20\d{2}))"
    ],
    "GuV": [
        r"gewinn",
        r"verlust",
        r"rechnung",
        r"((20\d{2}).*(20\d{2}))"
    ]
}
```

```
regex_patterns_5 = {
    "Aktiva": [
        r"a\s*k\s*t\s*i\s*v\s*a|a\s*k\s*t\s*i\s*v\s*s\s*e\s*i\s*t\s*e|anlageverm.{1,2}gen",
        r"((20\d{2}).*(20\d{2}))|((20\d{2}).*vorjahr)|vorjahr",
        r"Umlaufverm.{1,2}gen|Anlageverm.{1,2}gen|Rechnungsabgrenzungsposten|Forderungen",
        r"\s([a-zA-Z]|[0-9]{1,2}|[iI]+)[\.\)]\s"
    ],
    "Passiva": [
        r"p\s*a\s*s\s*s\s*i\s*v\s*a|p\s*a\s*s\s*s\s*i\s*v\s*s\s*e\s*i\s*t\s*e|eigenkapital",
        r"((20\d{2}).*(20\d{2}))|((20\d{2}).*vorjahr)|vorjahr",
        r"Eigenkapital|R.{1,2}ckstellungen|Verbindlichkeiten|Rechnungsabgrenzungsposten",
        r"\s([a-zA-Z]|[0-9]{1,2}|[iI]+)[\.\)]\s"
    ],
    "GuV": [
        r"gewinn|guv",
        r"verlust|guv",
        r"rechnung|guv",
        r"((20\d{2}).*(20\d{2}))|vorjahr"
        r"Umsatzerl.{1,2}se|Materialaufwand|Personalaufwand|Abschreibungen|Jahres.{1,2}berschuss|Jahres
        r"\s([a-zA-Z]|[0-9]{1,2}|[iI]+)[\.\)]\s"
    ]
}
```

```
regex_patterns_3 = {
    "Aktiva": [
        r"a\s*k\s*t\s*i\s*v\s*a|a\s*k\s*t\s*i\s*v\s*s\s*e\s*i\s*t\s*e|anlageverm.{1,2}gen",
        r"((20\d{2}).*(20\d{2}))|((20\d{2}).*vorjahr)|vorjahr"
    ],
    "Passiva": [
        r"p\s*a\s*s\s*s\s*i\s*v\s*a|p\s*a\s*s\s*s\s*i\s*v\s*s\s*e\s*i\s*t\s*e|eigenkapital",
        r"((20\d{2}).*(20\d{2}))|((20\d{2}).*vorjahr)|vorjahr"
    ],
    "GuV": [
        r"gewinn|guv",
```

```
        r"verlust|guv",
        r"rechnung|guv",
        r"((20\d{2}).*(20\d{2}))|vorjahr"
    ]
}
```