

Ist mein KI-Use-Case machbar?

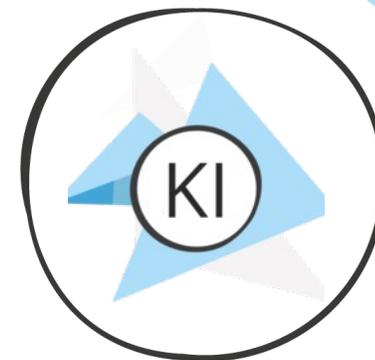
Ein Beispiel aus der Praxis

10.07.2024

Mannheim

Steffen Heberle
steffen.heberle@andrena.de

Steve Haupt
steve.haupt@andrena.de



Ausgangssituation



Stellt Use-Case vor:

Belege

Informationen

```
{  
  id: "anld",  
  kategorie: "Spendenbescheinigung",  
  typ: "Steuerbeg. Zweck",  
  betrag: "10.00",  
  datum: "2024-03-23",  
  land: "Inland",  
  empfänger: "Bester Verein der Welt",  
  ort:  
  ...  
}
```

Idee für KI Use-Case

Kunde lädt zum Brainstorming ein

Ausgangssituation



Idee für KI Use-Case

Stellt Use-Case vor:

Belege



Informationen

- Belege liegen als Bilder vor und können sehr divers sein
- Aktuell werden alle Informationen manuell extrahiert
- Fehler können rechtliche Konsequenzen haben

Kunde hat viele Fragen:

- Ist Use-Case heute machbar?
- Wie viel müssen wir investieren?
- Wie viel Mehrwert können wir erwarten?

Kunde lädt zum Brainstorming ein

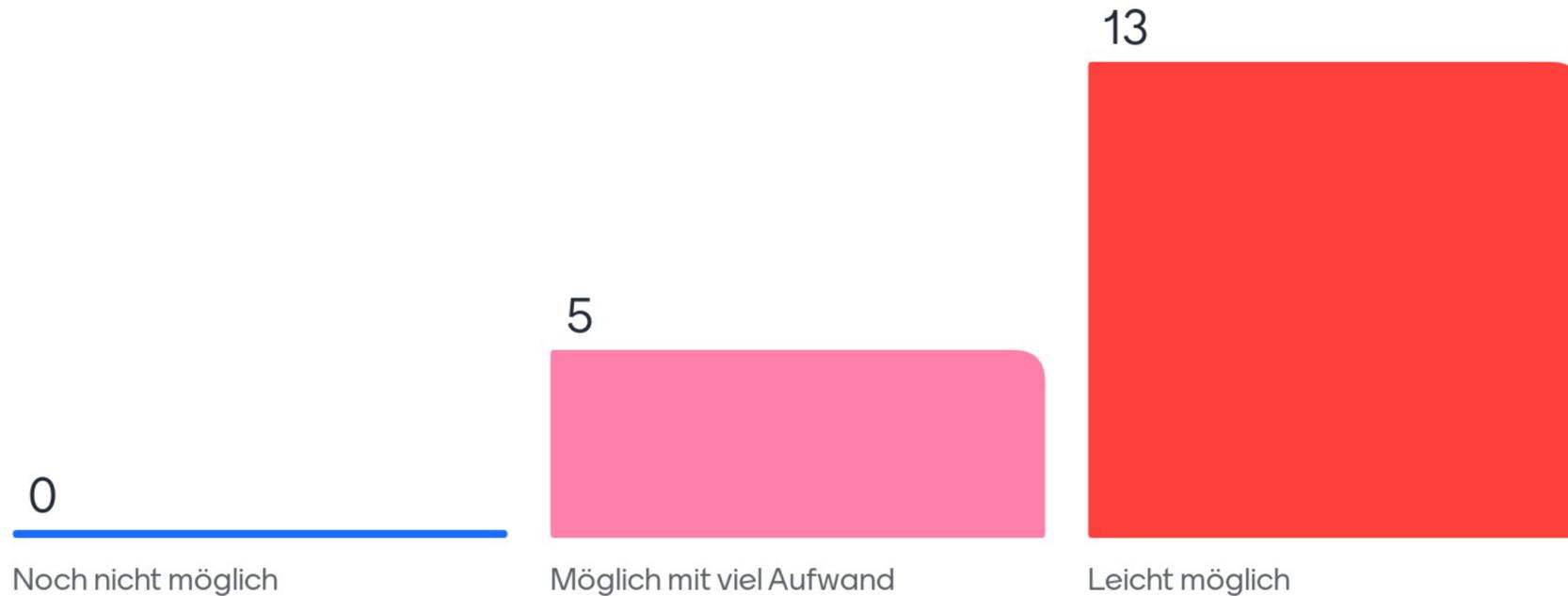
Ausgangssituation

Unser Aussage:

?

Ausgangssituation

Ist der Use-Case machbar?



Ausgangssituation

Unser Aussage:

"Ihr müsst heute loslegen, KI kann in eurem Anwendungsfall hohen Mehrwert bringen."

Unser Vorschlag:

Wir verproben den Use-Case an Beispielen "händisch", um eine Basline zu ermitteln.

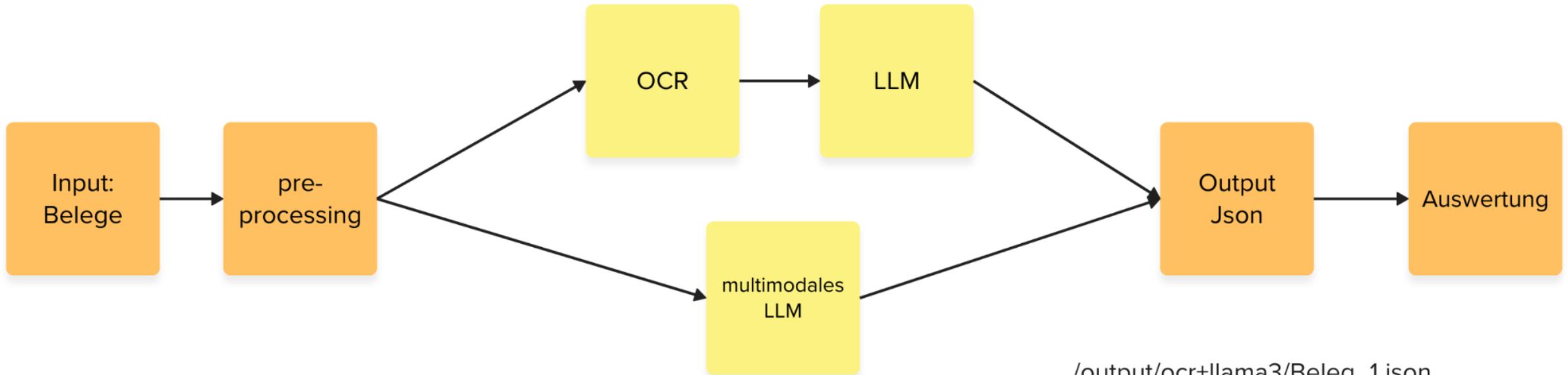
Ziel:

Fundierte Entscheidung treffen: **Investieren, ja** oder **nein?**

Hausaufgaben:

- Für den Kunden: > 100 gelabelte Belege vorbereiten
- Für uns: Skripte für Extraktion und Auswertung anlegen

Unsere Hausaufgaben

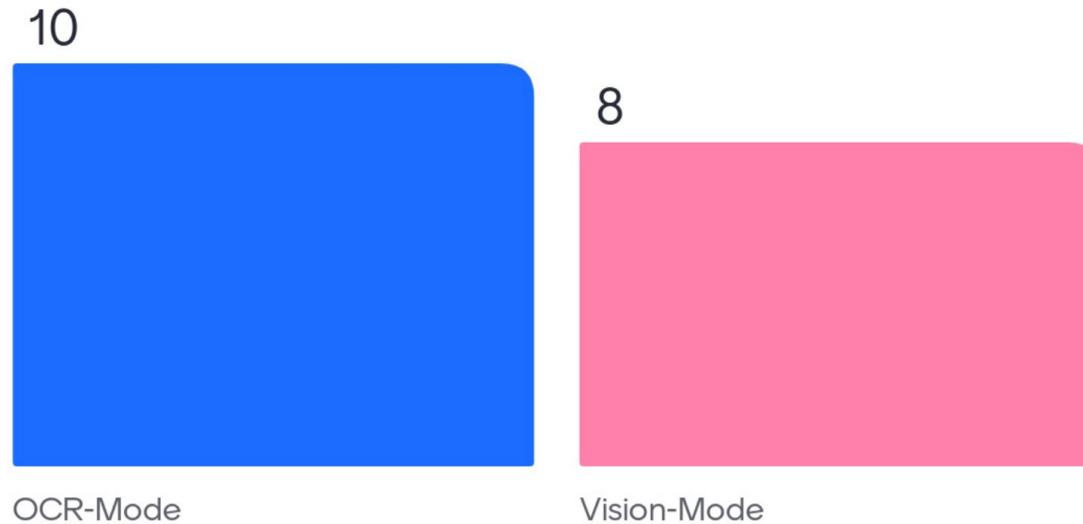


/input/Belege.xlsx # ground truth
/input/Beleg_1.pdf
/input/Beleg_2.pdf

./output/ocr+llama3/Beleg_1.json
./output/gpt4o/Beleg_1.json

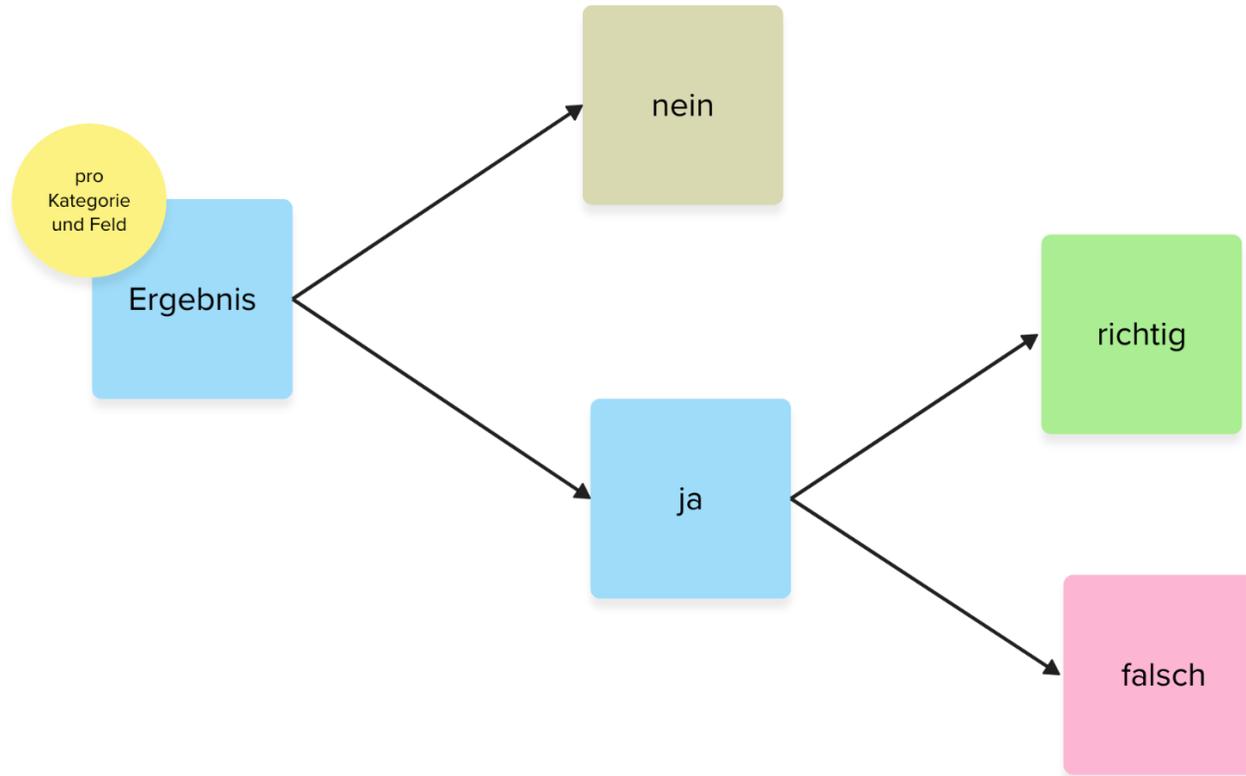
Unsere Hausaufgaben

Welche Variante liefert bessere Ergebnisse?



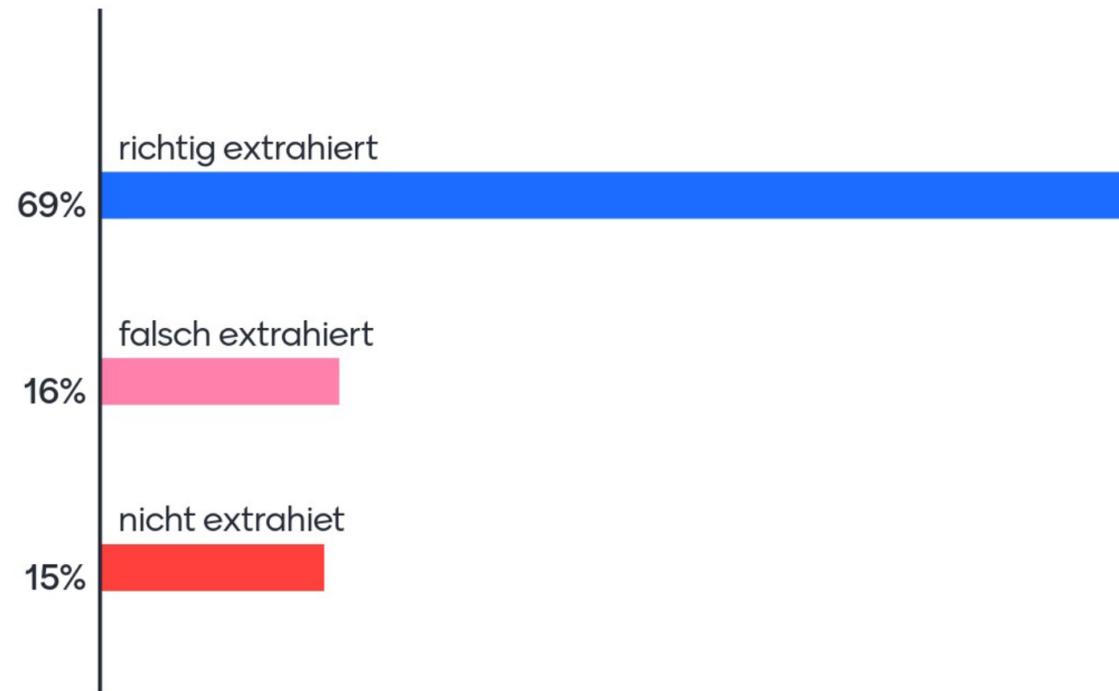
Unsere Hausaufgaben

Auswertung:



Unsere Hausaufgaben

Welche Baseline werden wir schaffen?



Unsere Hausaufgaben

```
from langchain.chat_models import init_chat_model

chat_model = init_chat_model(
    model=model_config['name'],
    model_provider=model_config['provider'],
    temperature=model_config['temperature'],
    model_kwargs=model_config['kwargs']
)
```

```
"model": {
  "provider": "groq",
  "type": "text",
  "name": "llama3-70b-8192",
  "temperature": 0.5,
  "kwargs": {}
},
```

```
"model": {
  "provider": "openai",
  "name": "gpt-4-turbo",
  "type": "text",
  "temperature": 1,
  "kwargs": {
    "logprobs": true,
    "response_format": {
      "type": "json_object"
    }
  }
},
```

```
"model": {
  "provider": "anthropic",
  "name": "claude-3-5-sonnet-20240620",
  "type": "text",
  "temperature": 0.5,
  "kwargs": {}
},
```

Unsere Hausaufgaben

```
@staticmethod
def create_message_content(provider: str, prompt: str, image_data: str) -> List[Dict[str, Any]]:
    if provider == "openai":
        return [
            {"type": "text", "text": prompt},
            {
                "type": "image_url",
                "image_url": {"url": f"data:image/jpeg;base64,{image_data}"}
            }
        ]
```

```
response = self._get_model_response(message_content, print_prompts)
extracted_document = self._parse_response(response, ground_truth_document)
```

Unsere Hausaufgaben

Wieso klappt das object parsing?

Json Mode

```
"model": {  
  "provider": "openai",  
  "name": "gpt-4-turbo",  
  "type": "text",  
  "temperature": 1,  
  "kwargs": {  
    "logprobs": true,  
    "response_format": {  
      "type": "json_object"  
    }  
  }  
},
```

Prompt

Hier ist eine JSON-Struktur die du gleich befüllen sollst:

```
...  
{  
  "Vorname": "Max",  
  "Name": "Mustermann",  
  "Straße/Hausnummer": "Hauptstr. 11",  
  "PLZ": "12345",  
  "Ort": "Mittelstadt",  
  "Betrag": "120,00",  
  "Empfänger": "Malteser Kreisverband",  
  "Datum": "2017-05-22",  
  "Land": "Inland",  
  "Typ": "Steuerbeg. Zweck"  
}  
...
```

Trage bitte die relevanten Informationen aus dem Bild in das JSON ein, ignoriere zusätzliche Informationen aus dem Bild und setze Felder, für die die Information nicht vorliegt, auf `null`.

Gib bitte nur das reine JSON aus ohne markdown oder sonstige Formatierung:

Unsere Hausaufgaben

Wie gehts im OCR Mode?

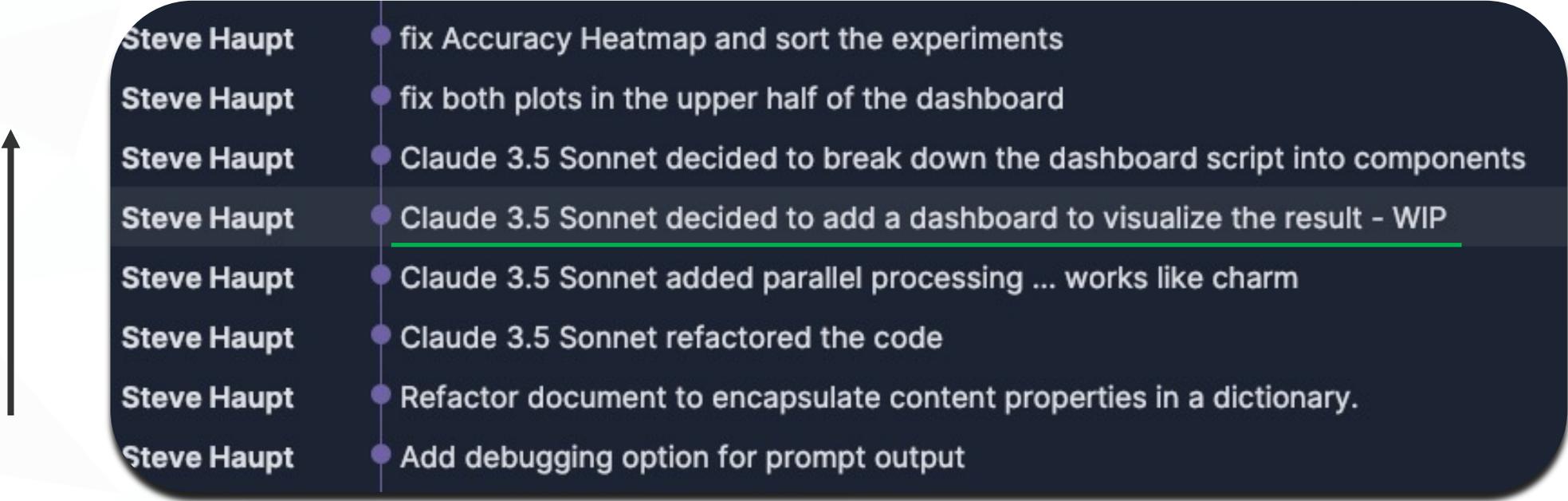
```
text = pytesseract.image_to_string(image, lang='deu')
output_filename = f"{base_name}.txt"
output_path = os.path.join(output_folder, output_filename)
with open(output_path, 'w', encoding='utf-8') as f:
    f.write(text)
```

```
with open(txt_path, 'r', encoding='utf-8') as file:
    text_content = file.read()

message_content = [
    {"type": "text", "text": f"{self.prompt}\n\nDocument content:\n{text_content}"}
]
```

Unsere Hausaufgaben

Unsere Arbeitsweise:

- 
- Steve Haupt • fix Accuracy Heatmap and sort the experiments
 - Steve Haupt • fix both plots in the upper half of the dashboard
 - Steve Haupt • Claude 3.5 Sonnet decided to break down the dashboard script into components
 - Steve Haupt • Claude 3.5 Sonnet decided to add a dashboard to visualize the result - WIP
 - Steve Haupt • Claude 3.5 Sonnet added parallel processing ... works like charm
 - Steve Haupt • Claude 3.5 Sonnet refactored the code
 - Steve Haupt • Refactor document to encapsulate content properties in a dictionary.
 - Steve Haupt • Add debugging option for prompt output

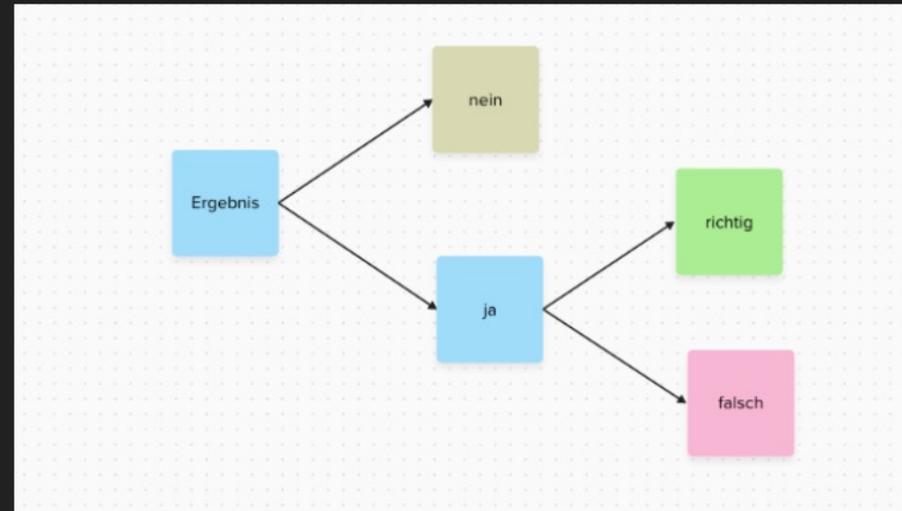
work smart not hard

Unsere Hausaufgaben

This parallelization should significantly speed up your script, especially for large numbers of documents. However, please note that the exact speedup will depend on factors like your network connection, API rate limits, and system resources.



steve



this is the data i want to analyze/visualize to get best insights

```
"number_of_documents": 2,  
  "end_time": "2024-06-29_00:14:22",  
  "duration": 6.606994,
```

Unsere Hausaufgaben

maybe you have more or better ideas to visualize the result of my experiment runs

come up with a plan for analyzing and plotting the data



AI Claude

Based on the data you've provided and the questions you've raised, here's a plan to analyze and visualize the results of your experiment runs:

To implement this plan:

1. Collect data from multiple experiment runs.
2. Use a data analysis library like pandas to process the data.
3. Use a visualization library like matplotlib or seaborn to create the plots.
4. Consider creating an interactive dashboard using tools like Plotly or Dash for easier exploration of results.

This comprehensive analysis will give you deep insights into the performance, efficiency, and areas for improvement in your document processing system.

Unsere Hausaufgaben

write code for an interactive dashboard with all the ideas you listed before



A) Claude

Certainly! I'll create an interactive dashboard using Plotly and Dash, which will allow you to visualize and analyze the data from multiple experiment runs. This dashboard will incorporate the ideas I mentioned earlier.

First, let's set up the necessary imports and data loading:

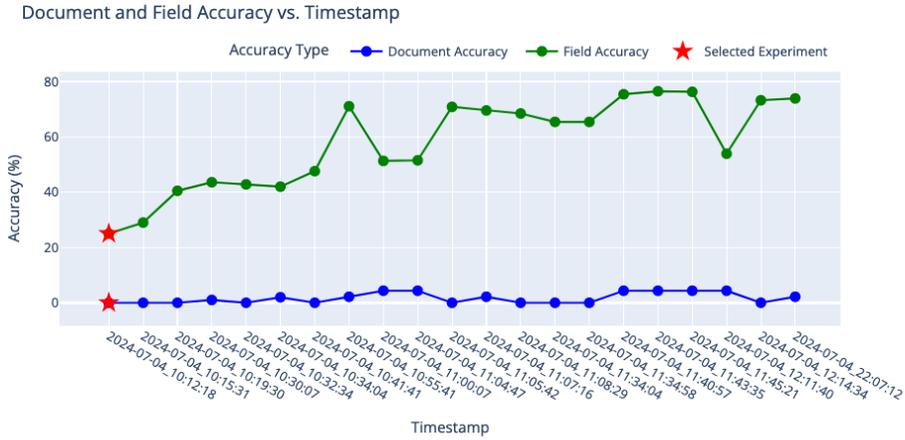
```
python Copy code  
  
import os  
import json  
import pandas as pd  
import plotly.express as px  
import plotly.graph_objs as go  
from dash import Dash, dcc, html, Input, Output  
from dash.dependencies import Input, Output  
  
# Function to load data from experiment folders  
def load_experiment_data(root_folder='output'):  
    data = []  
    for folder in os.listdir(root_folder):  
        if folder.startswith('experiment_'):  
            json_file = os.path.join(root_folder, folder, 'experiment_metadata
```

Experiments

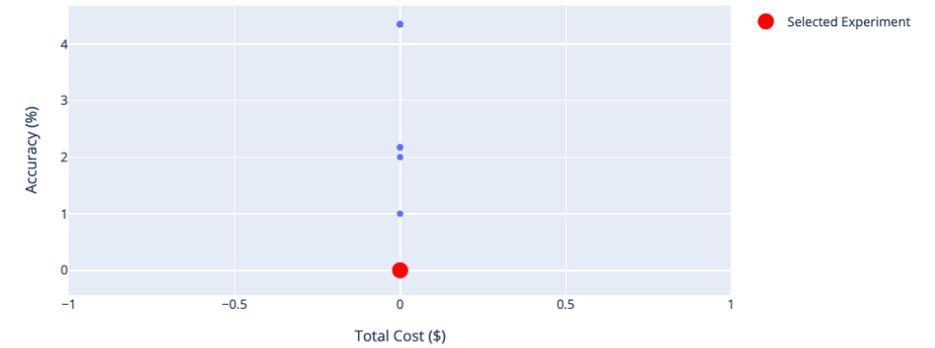
- [2024-07-04 10:12:18 text_gpt35 temp_0_5](#)
- [2024-07-04 10:15:31 text_gpt35 temp_0_5](#)
- [2024-07-04 10:19:30 text_gpt4i temp_0_5](#)
- [2024-07-04 10:30:07 text_gpt4i temp_0_5](#)
- [2024-07-04 10:32:34 text_gpt4o temp_0_5](#)
- [2024-07-04 10:34:04 text_claude35sonnet temp_0_5](#)
- [2024-07-04 10:41:41 vision_gpt4o temp_0_5](#)
- [2024-07-04 10:55:41 vision_gpt4o temp_0_5](#)
- [2024-07-04 11:00:07 text_gpt4i temp_0_5](#)
- [2024-07-04 11:04:47 text_gpt4i temp_0_5](#)
- [2024-07-04 11:05:42 vision_gpt4o temp_0_5](#)
- [2024-07-04 11:07:16 vision_gpt4o temp_0_5](#)
- [2024-07-04 11:08:29 vision_gpt4o temp_0_5](#)
- [2024-07-04 11:34:04 vision_gpt4o temp_0_5](#)
- [2024-07-04 11:34:58 vision_gpt4o temp_0_5](#)

Document Processing Experiment Analysis Dashboard

All Experiments Overview



Document Accuracy vs. Cost



Selected Experiment Info

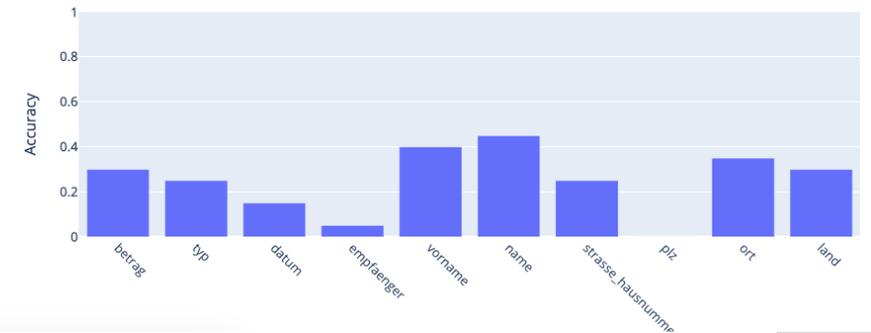
Start Time: 2024-07-04_10:12:18
 Experiment ID: text_gpt35_temp_0_5
 Model Name: gpt-3.5-turbo-0125
 Number of Documents: 20
 End Time: 2024-07-04_10:12:26
 Duration: 8.41 seconds
 Total Cost: \$0.0000

Selected Experiment Details

Document Processing Flow

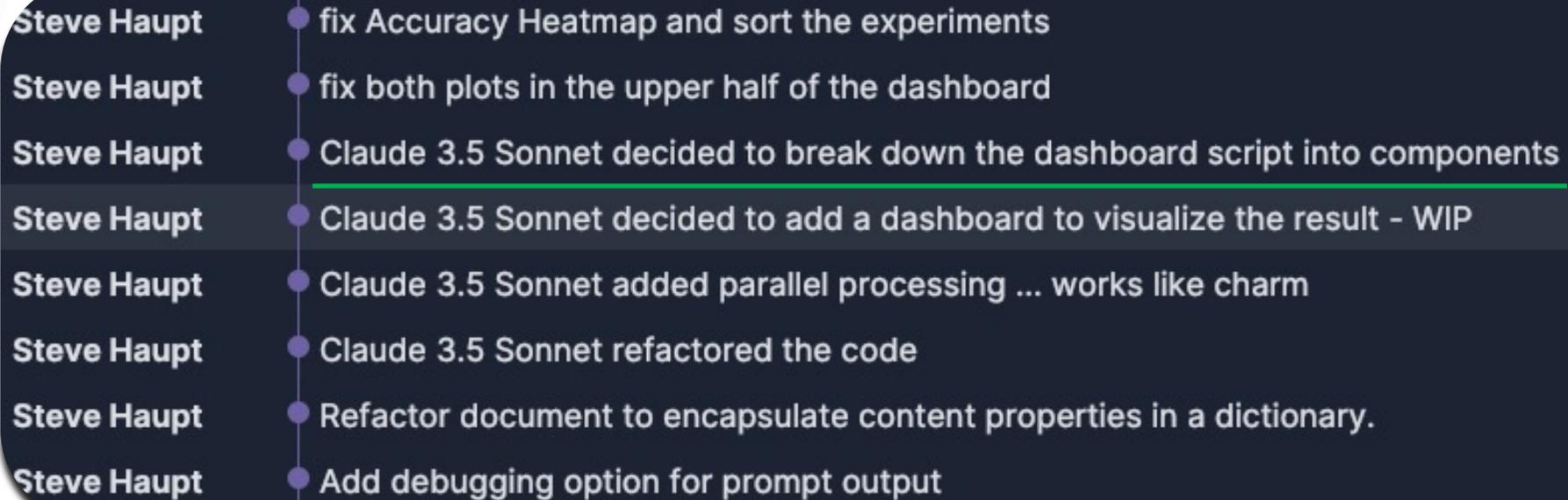


Field-wise Accuracy



Unsere Hausaufgaben

Unsere Arbeitsweise:

- 
- Steve Haupt • fix Accuracy Heatmap and sort the experiments
 - Steve Haupt • fix both plots in the upper half of the dashboard
 - Steve Haupt • Claude 3.5 Sonnet decided to break down the dashboard script into components
 - Steve Haupt • Claude 3.5 Sonnet decided to add a dashboard to visualize the result - WIP
 - Steve Haupt • Claude 3.5 Sonnet added parallel processing ... works like charm
 - Steve Haupt • Claude 3.5 Sonnet refactored the code
 - Steve Haupt • Refactor document to encapsulate content properties in a dictionary.
 - Steve Haupt • Add debugging option for prompt output

Unsere Hausaufgaben

Ein paar Tage vor der Baseline Messung:

Letzter Abstimmungs-Termin mit Kunden

- "... wir haben in einem PDF viele Belege ..."
- "... manchmal sind die Belege rotiert..."

```
def detect_text_orientation(pil_image):  
    # Convert PIL image to OpenCV format  
    open_cv_image = cv2.cvtColor(np.array(pil_image), cv2.COLOR_RGB2BGR)  
  
    # Convert to grayscale  
    gray = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2GRAY)  
  
    # Use pytesseract to detect text orientation  
    osd = pytesseract.image_to_osd(gray, output_type=Output.DICT)  
  
    rotation = osd.get('rotate')  
    return rotation  
  
> def rotate_image(image, rotation):  
  
def save_image(image, output_path):
```

Baseline Messung (3h)

9:00 Uhr: Agenda und Vorgehen besprechen

9:15 Uhr: 200 Belege und Excel File von USB Stick laden

- Excel Inhalt hat "falsches" Format
- Excel anpassen, Skripte anpassen

9:35 Uhr: Starte preprocessing (Rotation & OCR)

9:45 Uhr: Starte erste Experimente mit lokal laufenden Modellen

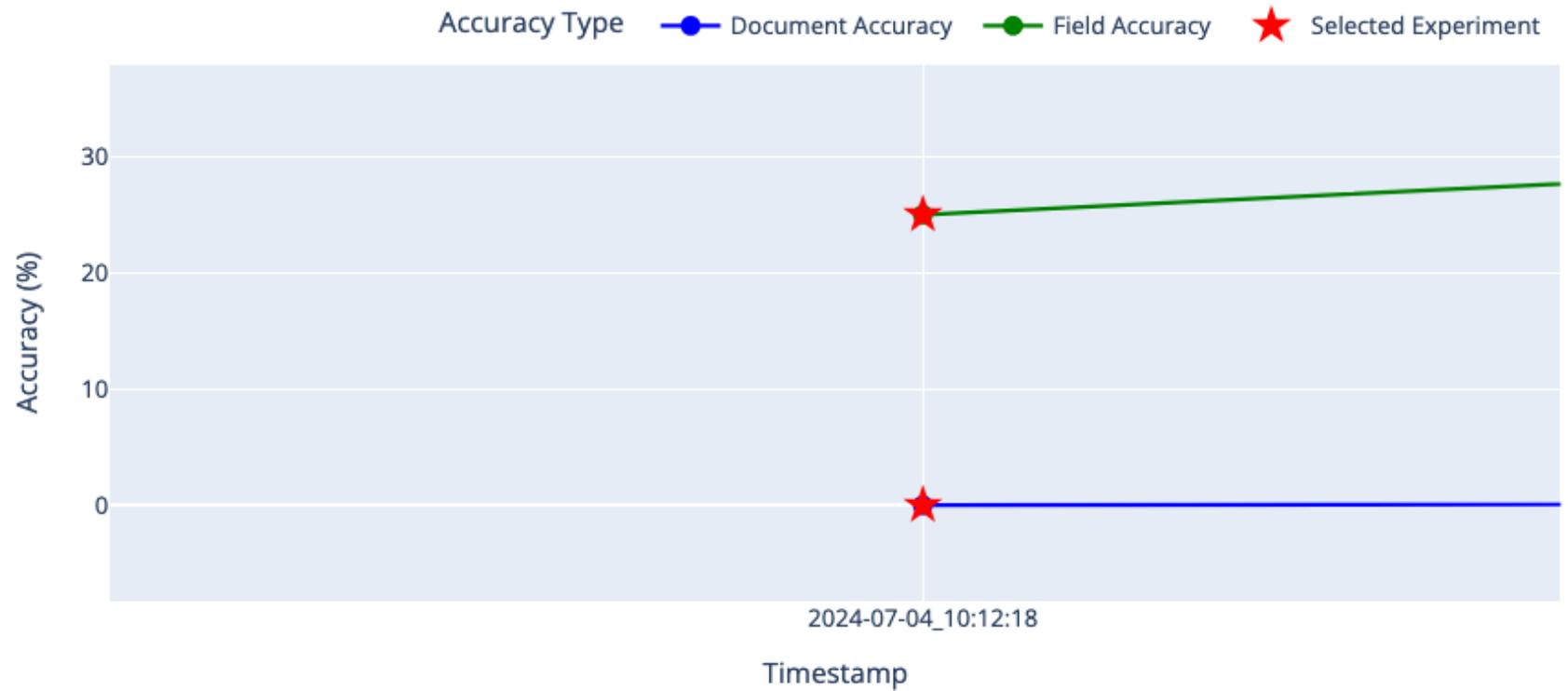
- Llama 3 8B
- Llama 3 70B
- katastrophales Ergebnis

10:10 Uhr: Starte erste Experimente mit closed source Modellen

Baseline Messung (3h)

- **OCR Mode**
GPT-3.5-turbo
20 Belege

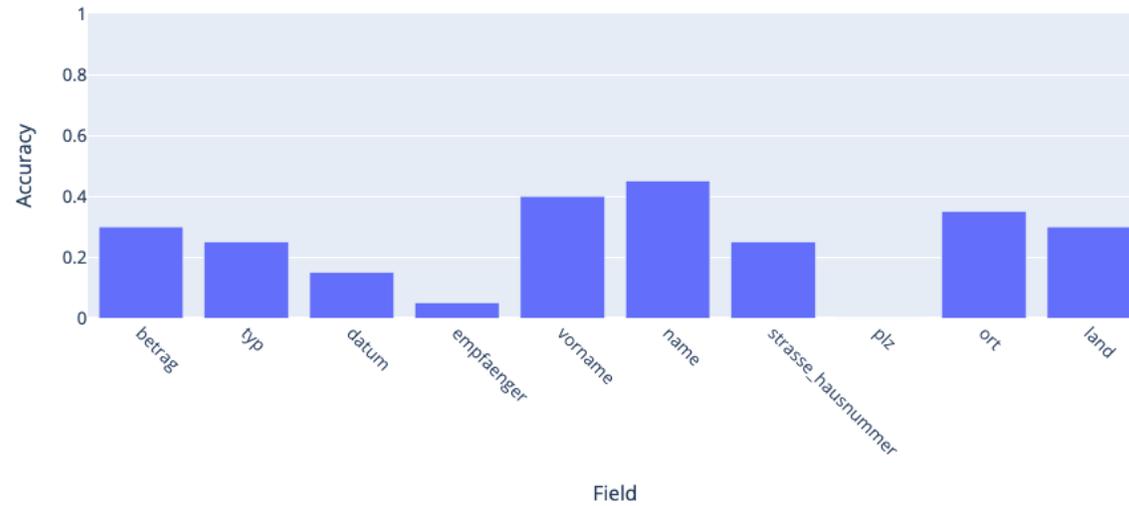
Document and Field Accuracy vs. Timestamp



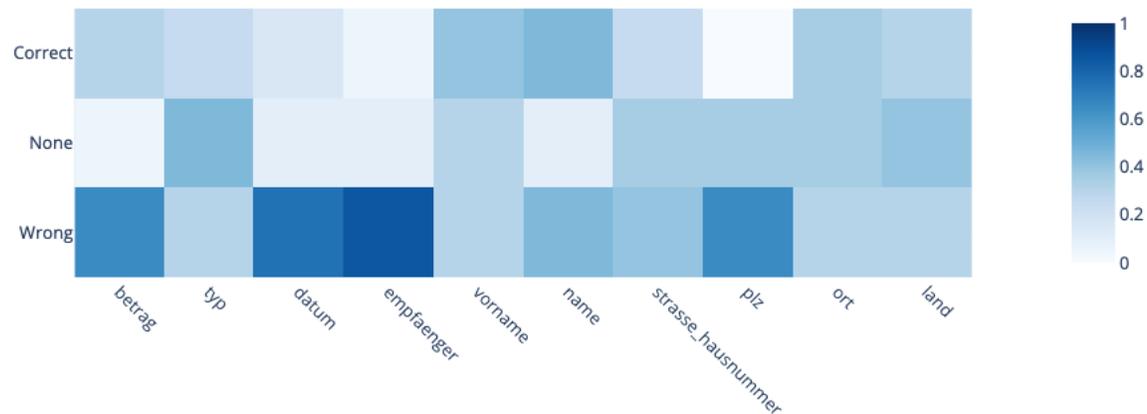
Baseline Messung (3h)

- **OCR Mode**
GPT-3.5-turbo
20 Belege

Field-wise Accuracy



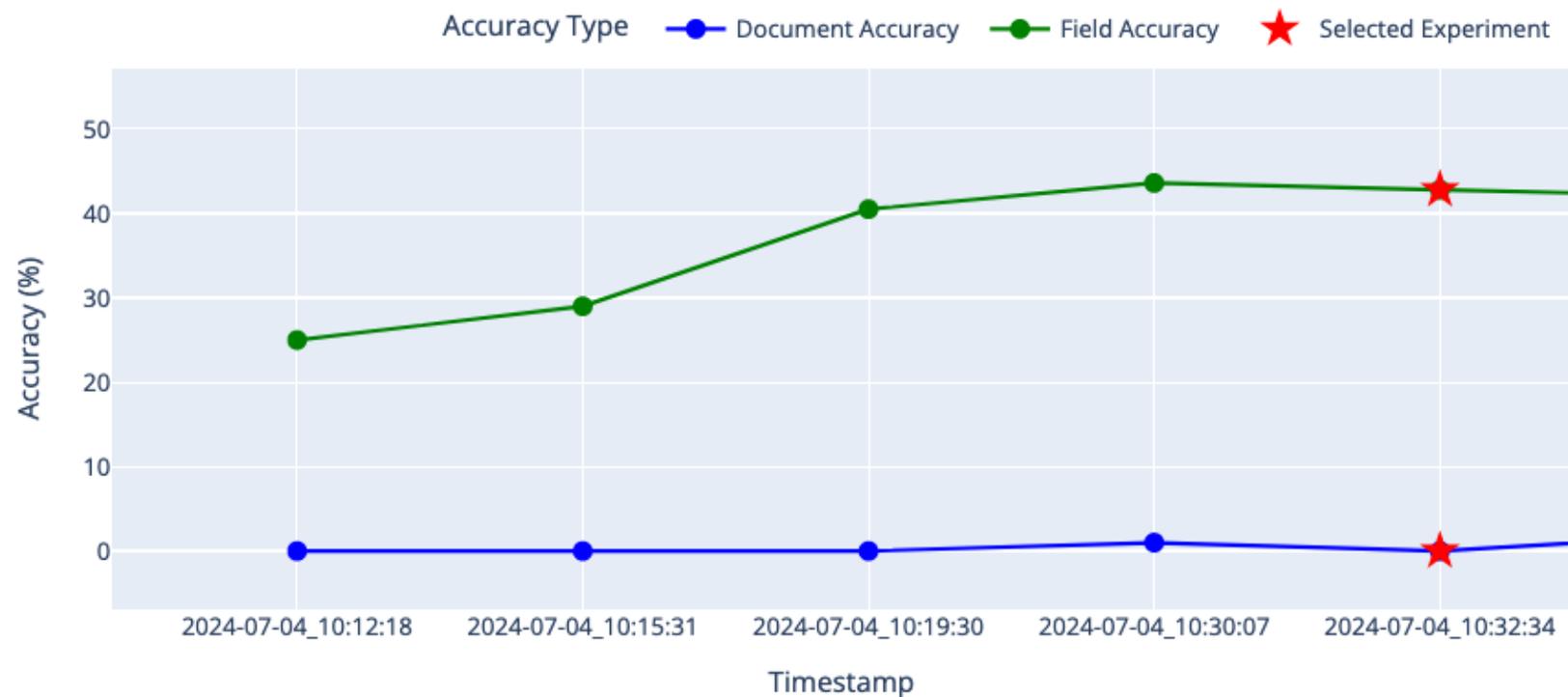
Accuracy Heatmap for 2024-07-04_10:12:18_text_gpt35_temp_0_5



Baseline Messung (3h)

- **OCR Mode**
GPT-4 (4, 4t, 4o)
20 Belege

Document and Field Accuracy vs. Timestamp



Baseline Messung (3h)

- **OCR Mode**
GPT-4 (4, 4t, 4o)
20 Belege

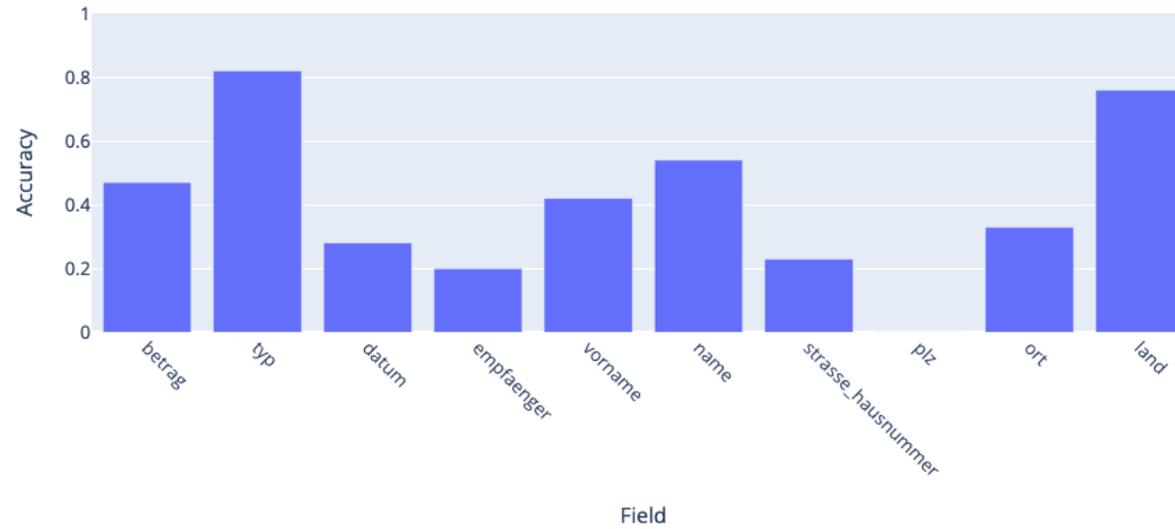
Load Differences

Click the button to load differences.

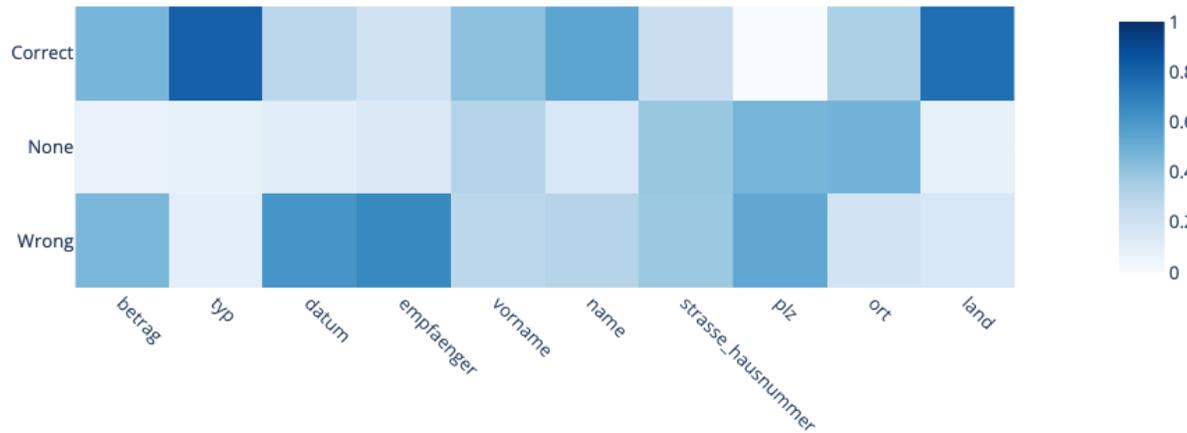
Document: 46505080_Spende_C1257D

plz:
Extracted: 15299
Correct: 15299.0

Field-wise Accuracy



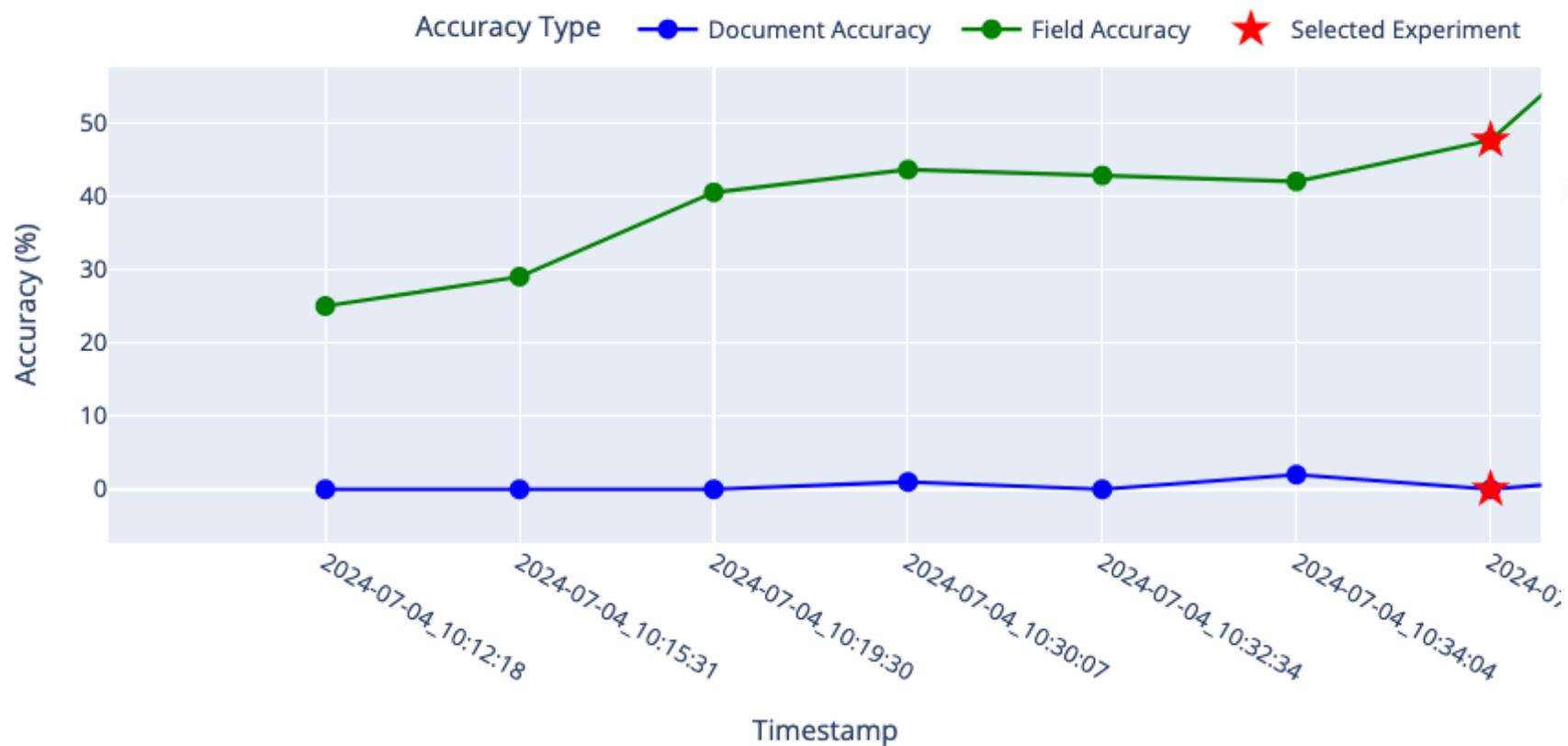
Accuracy Heatmap for 2024-07-04_10:19:30_text_gpt4t_temp_0_5



Baseline Messung (3h)

- **Vision Mode**
GPT-4o
100 Belege

Document and Field Accuracy vs. Timestamp



Baseline Messung (3h)

- **Vision Mode**
GPT-4o
100 Belege

Load Differences

Click the button to load differences.

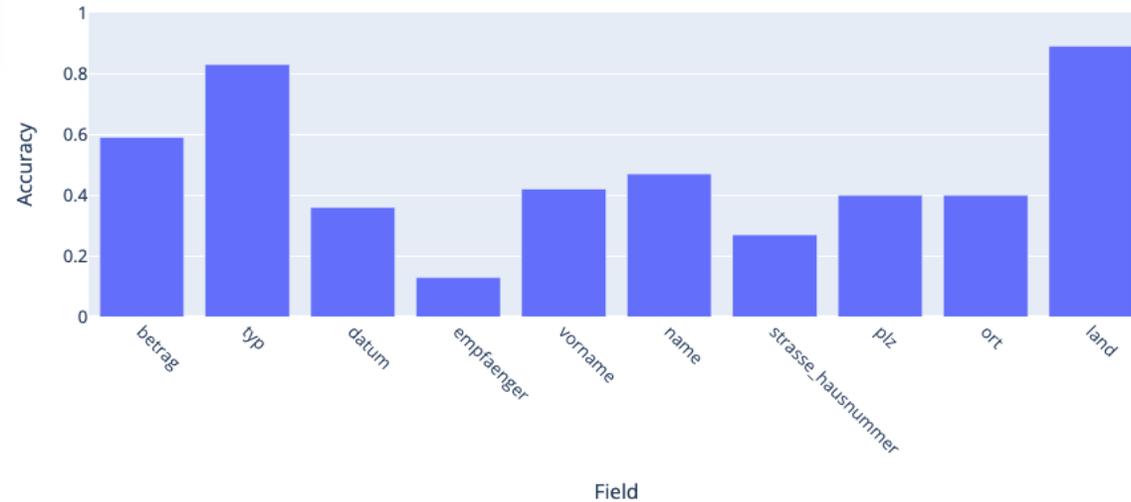
Document: 14407438_Spende_C1257C

plz:
Extracted: 04279
Correct: 4279.

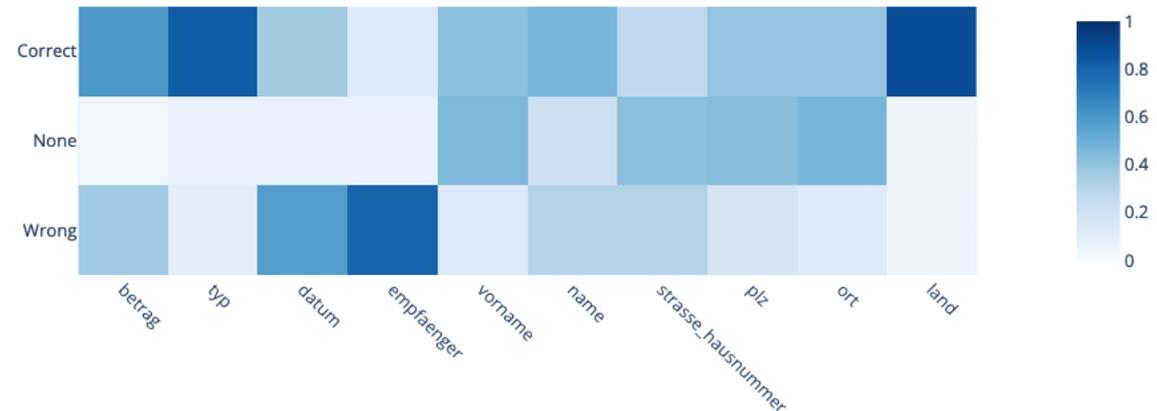
betrag:
Extracted: 500.0
Correct: 18.41

empfaenger:
Extracted: atmosfair gGmbH
Correct: Greenpeace e. V.

Field-wise Accuracy



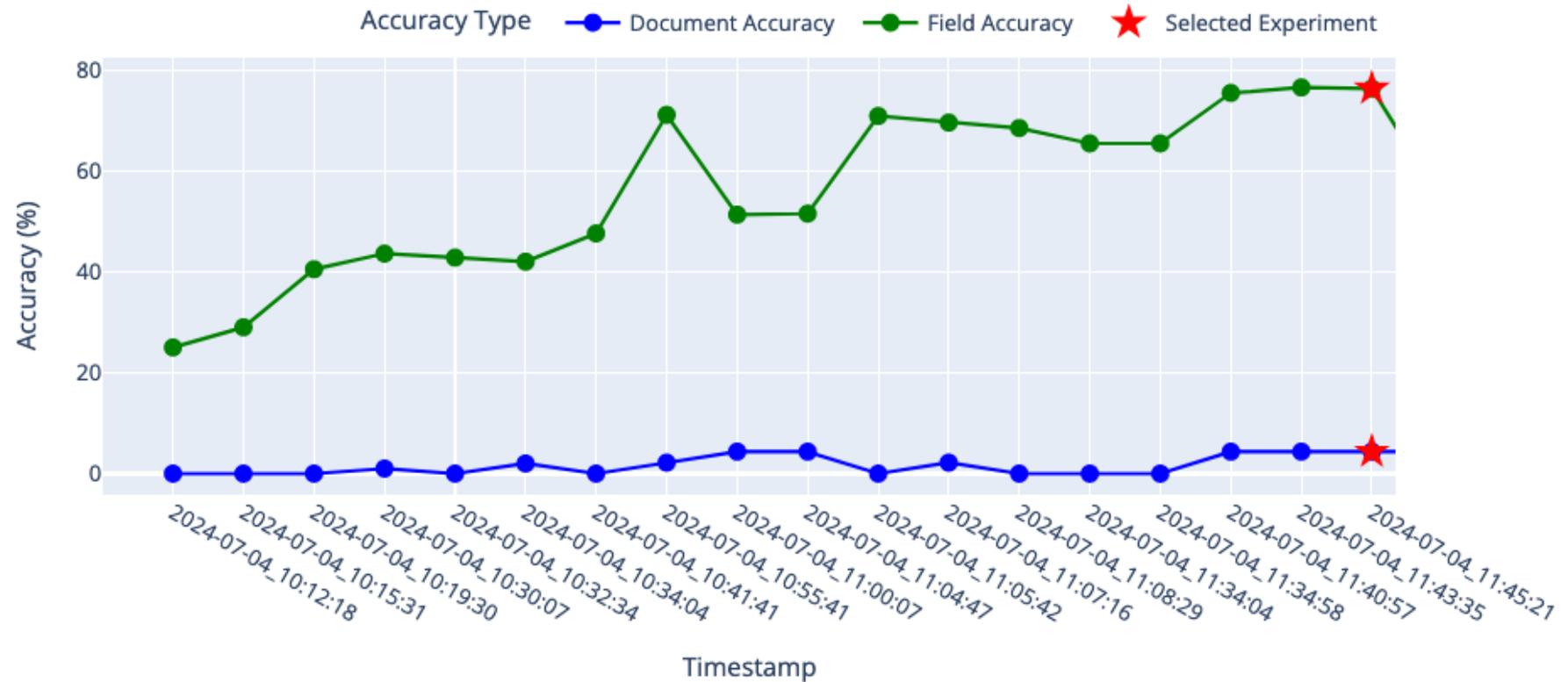
Accuracy Heatmap for 2024-07-04_10:41:41_vision_gpt4o_temp_0_5



Baseline Messung (3h)

- **Vision Mode**
GPT-4o
46 (bereinigte) Belege

Document and Field Accuracy vs. Timestamp



Baseline Messung (3h)

- **Vision Mode**
GPT-4o
46 (bereinigte) Belege

Load Differences

Click the button to load differences.

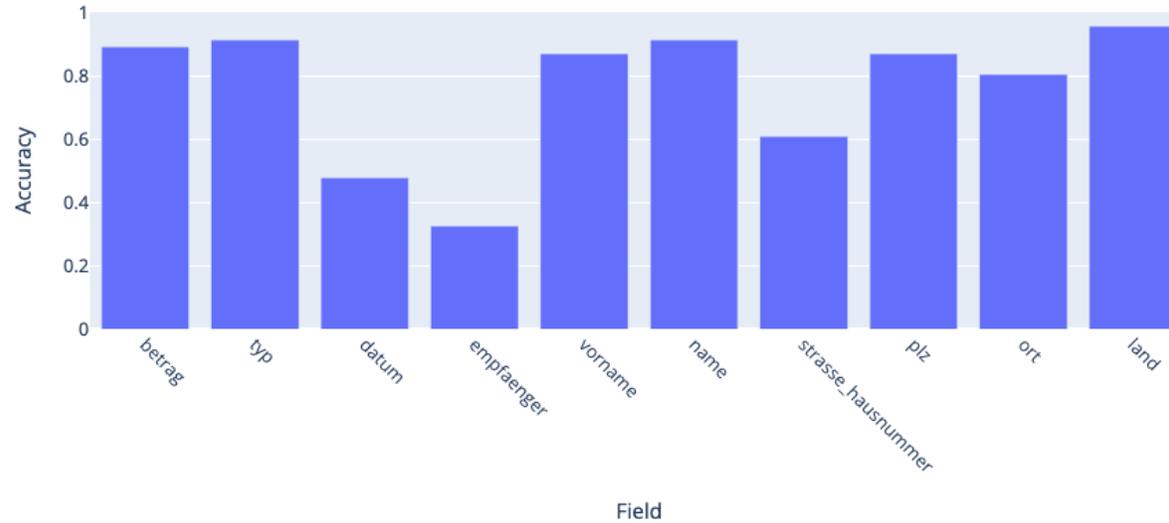
empfaenger:
Extracted: Malteser Hilfsdienst e.V.
Correct: Malteser Hilfsdienst e. V.

empfaenger:
Extracted: BUND (Bund für Umwelt und Naturschutz Deutschland e.V.)
Correct: Bund für Umwelt und Naturschutz Deutschland e.V.

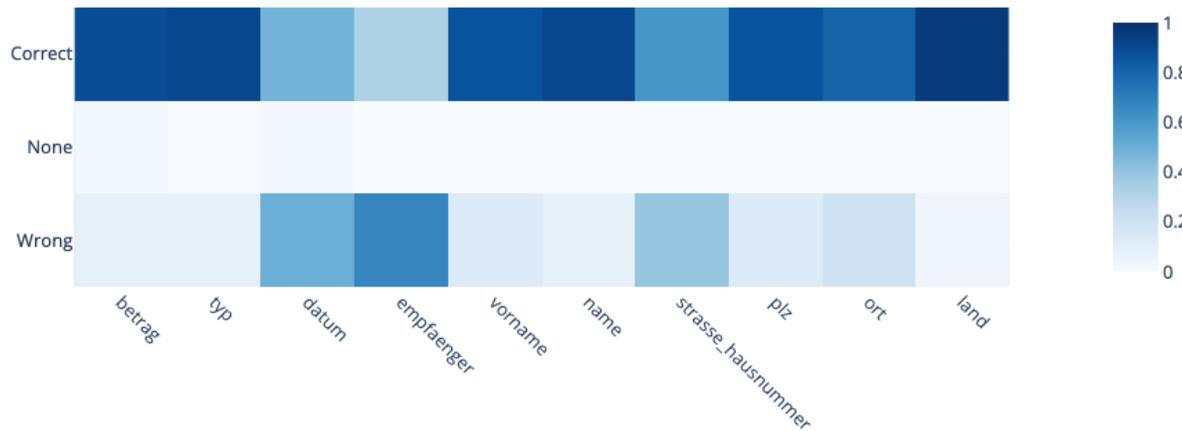
datum:
Extracted: 2022-12-31
Correct: 2022-01-01

betrag:
Extracted: -12.0
Correct: 12.0

Field-wise Accuracy



Accuracy Heatmap for 2024-07-04_11:45:21_vision_gpt4o_temp_0_5



Nachbetrachtung

- Vision Mode > OCR Mode
- Kosten ca. 1 Cent pro Beleg
- Baseline: ca. 90% Genauigkeit bei der Felderkennung
- Im Gespräch vor Ort festgestellt hohe Accuracy ist nicht wichtig, sondern kleine Fehlerrate

Nachbetrachtung

Strategien um die Fehlerrate so klein wie möglich zu bekommen:

- Zerlegung in Teilaufgaben
- Konfidenzbewertung durch das LLM
- Nutzung von Logprobs-Statistiken
- Majority Voting
- Einstimmigkeit
- Clustering und Analyse verbleibender Fehler
- Human-in-the-Loop Ansatz

Nachbetrachtung

Nächste Schritte:

- Aufbau des Evaluierungs-Frameworks
- Evaluierung der menschlichen Fehlerrate
- Detaillierte Analyse der Fehlercluster
- Implementierung und Test der vorgeschlagenen Verbesserungsstrategien.
- Balance zwischen Fehlerrate, Genauigkeit, Verarbeitungszeit und Kosten ermitteln
- Vorbereitung für einen möglichen Human-in-the-Loop Ansatz.



Viele Experimente, viele Entscheidungen: Agile Arbeitsweise noch wichtiger!

Kontakt



steve.haupt@andrena.de



www.linkedin.com/in/stevehaupt



www.youtube.com/@developers-club

Fragen

