Receipt details extraction using synthetically generated training data

Data Science Project 2 (DSPRO2) FS 2025
Hochschule Luzern - Informatik

Version 1.0

Jamian Rajakone
jamian.rajakone@stud.hslu.ch

Tamino Walter
tamino.walter@stud.hlsu.ch

Diego Gonzalez
diego.gonzalez@stud.hslu.ch

June 25, 2025

Contents

1 Introduction

1.1 Problem description

Imagine that your desk is full of receipts that you need to digitalize. You will need to manually copy every position and all the details of the receipts into, for example, an accounting software. The whole process needs hours of focus and concentration. It would be frustrating to write off everything by hand.

What if you could automate the input process by only taking a photograph of the receipt and nothing else. Extracting the data and copying it into the accounting software would happen automatically, this would save us so much time and effort. But not only would it save us lots of time, it would also save cost, because there is less human work needed to do the accounting inputs.

The obvious solution would be to try and use ChatGPT. But how can we ensure that the data privacy is guaranteed? On some receipts more or less personal information is shown and depending on the content you wouldn't want anyone to access this information. So ideally what you want would be a solution which can be run locally so that the data never has to leave your device. Alternatively a cloud solution would suffice as long as it is hosted by a trusted company in a country with high data protection laws.

1.2 Main goal

To solve this problem we aim to develop a new AI model which is optimised for the task of data extraction from receipts. Now training such a model requires a enormous amount of data. If we were to collect real receipts we would not only require a lot of time and money for the collection process, but we would stumble again into the problem we're trying to solve; manually extracting the data of hundreds of receipts to label them.

To avoid this problem we came up with the plan to synthetically generate receipts data and to fully rely on generated data during the training process. Setting this up would also be time consuming but once we are able to generate real looking receipts we can scale this process to generate not only a few hundreds but thousands of receipts.

So in this project we have two main goals: firstly we want to optimize a model to extract all relevant information from the image of a receipt, as the second goal we want to find out how well such a model performs after it has been fine tuned exclusively with the usage of synthetically generated data.

1.3 Related works

Document extraction represents a well-established research domain encompassing multiple methodological approaches that have evolved significantly over the past decades. The field has progressed from traditional rule-based systems and template matching techniques to sophisticated machine learning paradigms, including modern deep learning architectures. Early approaches relied heavily on optical character recognition (OCR) combined with handcrafted feature extraction methods, while contemporary solutions leverage convolutional neural networks (CNNs), transformer-based models, and multimodal architectures to achieve state-of-the-art performance. The landscape of document extraction methodologies can be broadly categorized into several distinct approaches: layout-based extraction utilizing spatial relationships and document structure analysis, content-based extraction focusing on textual patterns and semantic understanding, and hybrid approaches that combine multiple techniques for enhanced robustness. Recent advances in computer vision and natural language processing have enabled end-to-end trainable systems capable of simultaneously handling document layout analysis, text recognition, and information extraction tasks. This project builds upon established methodologies and theoretical foundations from the existing literature, integrating proven techniques with novel architectural innovations to address specific challenges in document extraction. Our approach synthesizes insights from seminal works in document understanding, leveraging transfer learning from pre-trained models while incorporating domain-specific adaptations to optimize performance for our target application domain.

1.3.1 Introduction to OCR pipelines

Traditional OCR pipelines for receipt and document processing typically employ a modular, multi-stage architecture. First, a text detection module (e.g., CTPN, TextSnake) identifies text regions; next, a recognition model (e.g., CRNN, TrOCR) reads the text; and finally, a parsing or IE module extracts structured information. While effective, each stage introduces potential errors and dependencies, and assembly often relies on extensive heuristics or post-processing. For survey-level context and benchmarks (e.g., the SROIE competition), tools like docTR combine detection and recognition, but still follow modular pipelines [3].

1.3.2 Advancements in localization-free and chunk-based models

Localization-free approaches, such as adaptations of TrOCR, bypass the text-detection stage. Zhang et al. fine-tune a pretrained TrOCR on progressively larger random “chunks” of receipt images, achieving an 87.8% F1 score and 4.98% CER on SROIE—without explicit bounding boxes [7]. By segmenting and gradually scaling to full-page inputs, the model directly transcribes full receipts end-to-end, significantly simplifying the OCR pipeline.

1.3.3 Vision-language and LLM-based approaches

Recent receipt extraction methods leverage multimodal strategies, combining visual layout understanding with large language models. For instance, models like LayoutLMv2/v3 and ViBERTgrid enrich text inputs with spatial and visual embeddings. More cutting-edge, generative approaches (e.g., Donut, Pix2Struct, mPLUG-DocOwl) produce structured output directly, often without OCR. Surveys covering modular versus end-to-end VLMs provide a comprehensive architectural overview [8].

1.3.4 Entity/key-value pair extraction pipelines

Rather than just recognizing text, entity/key-value pair extraction pipelines assign semantic labels to detected text regions. For example, Tan et al. (2023) apply YOLO-based region detection followed by BERT classification, extracting and linking keys and values [5]. Graph-based models like MatchVIE use entity relevancy to improve pairing [6]. Meanwhile, EATEN (Entity-aware Attention for Single-Shot Visual Text Extraction offers single-shot, entity-aware attention extraction directly from images [1]. New datasets like KVP10k further benchmark these approaches [4].

1.3.5 SmolVLM - Small and Efficient Vision-Language Model

Recent advances in vision-language modeling have produced powerful but computationally intensive architectures, often requiring large-scale hardware for training and deployment. To address these limitations, SmolVLM was introduced as a lightweight multimodal model designed to perform vision-language tasks efficiently in low-resource environments. The model integrates a compact visual encoder based on a scaled-down Vision Transformer (ViT-S) with a small decoder initialized from a lightweight language model. SmolVLM is trained using instruction-following datasets, enabling it to handle a variety of multimodal tasks such as image captioning, document understanding, and visual question answering. To reduce the number of trainable parameters during adaptation, the model supports parameter-efficient fine-tuning strategies like LoRA. These techniques allow SmolVLM to be fine-tuned on downstream tasks with significantly lower memory and compute requirements compared to larger VLMs. Evaluation on standard document understanding benchmarks, including CORD and DocVQA, demonstrates that SmolVLM achieves competitive results relative to much larger models. While it does not reach state-of-the-art performance, its efficiency and modularity make it a strong candidate for rapid experimentation and deployment on edge devices or within academic settings where resources are limited.

1.3.6 DONUT - Document Understanding Transformer

Traditional visual document understanding (VDU) approaches depend on optical character recognition (OCR) as an intermediate step, which introduces three key limitations: high computational overhead, reduced flexibility across languages and document types, and error propagation from OCR misrecognitions that degrade downstream performance. Kim et al. (2022) proposed

Donut, an end-to-end transformer architecture that eliminates OCR dependency by directly processing document images. The model combines a Swin Transformer Base visual encoder with a BART text decoder, enabling simultaneous visual pattern recognition and text comprehension without character-level supervision. The authors evaluated Donut on three core VDU tasks: document classification, information extraction, and document visual question answering. Performance was measured using field-level F1-score and Tree Edit Distance metrics. The training involved pre-training on synthetic documents followed by task-specific fine-tuning. Results demonstrated that Donut consistently outperformed OCR-based baselines across all tasks, showing particular strength in complex layouts and multilingual scenarios. This end-to-end approach established improved robustness to visual variations while eliminating OCR bottlenecks, representing a significant advancement in visual document understanding methodology.

1.3.7 Critical comparison and research gaps

Despite significant progress, challenges remain:

- **Layout dependence:** Modular pipelines struggle with novel receipt styles, while localization-free models may require heavy fine-tuning.
- **Language limitations:** Most methods target English/SROIE; multilingual robustness (e.g., for languages that aren't used by a lot of people) is under-explored.
- **Entity linking:** Precisely pairing keys and values can be tricky in irregular layouts. Graph-based or generative methods help but aren't flawless.
- **LLM hallucination:** Vision-language generators risk fabricating non-existent fields.
- **Data scarcity:** Many approaches rely on large labeled corpora—often proprietary. Benchmarks like KVP10k help, but broader coverage is needed.

Future work must explore hybrid models that combine localization-free, multimodal, and structured extraction techniques to handle multilingual and diverse receipt formats more reliably.

2 Methods

2.1 Data

Our experimental framework employs a mixed-dataset approach comprising both synthetic and real-world data to ensure robust model evaluation across diverse scenarios. The training and validation datasets consist of synthetically generated samples, while the test dataset is also composed of authentic real-world documents to assess model generalization capabilities under practical deployment conditions.

2.1.1 Synthetic data

We used a lot of synthetic data so that we don't have to get and label a huge amount of real-world receipts. The challenge was to make the synthetic data as authentic as possible. To generate receipts we used our data generation pipeline.

The design of the generated receipts is mostly inspired by Coop and Migros receipts. But we also took influence from the receipt design of other stores and brought in our own ideas to make the data generation as diverse as possible. We could extend the data generation even further but decided to keep it relatively simple for now to see how well it performs in the current state.

With said data generation pipeline we are able to generate 8'340 images per hour on local hardware (M3 MacBook Pro).

2.1.2 Real-world receipt data

In the duration of this project we collected authentic retail receipts from various commercial stores. Data acquisition involved standard smartphone photography of physical receipts under typical ambient lighting conditions, reflecting realistic scenarios encountered in practical deployment environments. This approach deliberately mirrors common user behavior and device capabilities to ensure ecological validity of the test dataset.

The images were captured using handheld devices, introducing natural variations in image quality, lighting, perspective, and potential motion blur that are characteristic of real-world document capture scenarios. This methodology ensures that model evaluation accurately reflects performance expectations under typical usage conditions rather than idealized laboratory settings.

Labels for these real receipts had to be mostly created manually. The annotation process for incorporated multi-stage quality control measures, including cross-validation by multiple annotators and systematic review of AI-generated labels to minimize annotation errors and ensure dataset reliability for model training and evaluation.

2.1.3 Dataset composition

Our train and validation dataset fully consists out of synthetically generated data. We solely relied on this type of data during training to check if it is possible to train a model on computer generated data and still get good results on real data.

Our test set was split into 4 different test sets with different compositions. This split in the test sets should show us how well our model generalizes on receipts from the same, similar and completely different domains.

Test set 1 consists of 100 synthetically generated data. This data set should ensure that we don't overfit on the validation set by verifying that new synthetic receipts are also processed correctly.

Test set 2 consists of 31 Coop and Migros receipts. This data set has the purpose to show the gap between the synthetic data, which was mainly inspired by the layout of Coop and Migros receipts, and the real receipts from these stores.

Test set 3 consists of 8 receipts from different grocery shops. Including Lidl and Aldi receipts among others. This data set should show us how well the model generalizes on different grocery store receipts.

Test set 4 consists of 12 receipts from completely different domains. This includes for example receipts from book stores, gas stations, restaurants and hardware stores.

In total this results in 151 records for testing.

2.1.4 Input data

The only input our model should get is the preprocessed image of the receipt. These images can vary in size because the size (primarily the height) of real receipts also greatly varies so the model must be able to handle this.

2.1.5 Output data (label)

The annotation schema employs JavaScript Object Notation (JSON) format, selected for its widespread adoption in structured data representation and its dual compatibility with both human readability and machine processing requirements. This standardized format facilitates seamless integration with downstream applications and enables efficient data interchange between system components.

The JSON schema was established during the initial project design phase to ensure consistency across all annotation tasks and to align with the target application's data processing pipeline. The structured format supports hierarchical information organization and enables precise field-level extraction validation.

Label Schema Structure:

```json
{
    "shop_name": "Migros",
    "date": "17.05.2025",
    "total": 10.15,
    "products": [
        {
            "name": "Brot 360g",
            "amount": 2,
            "price": 7.2
        },
        {
            "name": "Zwiebeln",
            "amount": 4,
            "price": 2.4
        },
        {
            "name": "Wasser 0.5l",
            "amount": 1,
            "price": 0.55
        }
    ]
}
```

Listing 1: JSON Label Example

shop_name	Shop or businessname `string`
date	"DD.MM.YYYY format `string`
total	Numeric total amount `string`
products.name	Item description `string`
products.amount	Numeric quantity of the item `string`
products.price	Numeric total for this item `string`

For simplicity we only selected the minimal amount of fields required while keeping enough fields to keep a usable result for real-life scenarios. When our Proof-of-Concept works the following fields could also be added:

- purchase time, additionally to the date
- unit price of products, currently we only extract the total amount of each position
- discounts
- location / branch of the shop
- payment methods, multiple if the amount was, for example, partially payed with a gift card

2.2 Data generation

The data generation pipeline is split up into two different steps. In the first step we generate the receipt content and render it to an image file. In the second step we use projection and quality reduction to make the result more realistic.

2.2.1 Template generation and rendering

To get sensible synthetic data we have to be able to generate a lot of receipts which look as realistic as possible with a stable and consistent method to avoid formatting errors in some scenarios. We figured out that using HTML and CSS to generate the layout of the receipts was our best option.

The first step we did was to use a receipt from a grocery chain and replicated it as accurately as possible using HTML and CSS. To get the best results we measured the receipt, font sizes and gaps of the physical receipt and compared them to the result shown on screen and adjusted the values accordingly to get an almost perfect match of the original receipt.

Implications & differences to the original

By using HTML and CSS for this we had one major difference in the layout compared to real receipts. In most real receipts every character has the same width independently of the font-size. Additionally the letters are placed on a fixed grid which fills the entire size of the receipt. These both factors couldn't be replicated using our approach.

To simulate the letter grid we used a monospaced font to make all characters have the same width. And to simulate these slim characters we rendered text in an area with a larger width than the width we would normally use and resized the rendered text to the width we needed. Due to this process the character got slenderer while maintaining their normal height which gave them the typical optic of a receipt font.

Figure 1: Comparison of a replicated Coop receipt (left) and the original (right)

To make the receipts as realistic as possible we had to do a lot of adjustments to make the visuals as authentic as possible. Listing all micro decisions we had to make for this would go beyond the scope of this section. But as an example, we had to replace all zeroes with big "O"s because in most monospaced fonts the zero has a dot or a diagonal line which is not present on real receipts.

Disassembly into sections

Next we split the manually recreated receipts into smaller sections. Most of the sections are present across different receipt types just with a slightly different design. To get a feeling of the scope of such a section the list below lists some of the ones we extracted.

- Shop name / logo
- Product table
- Total
- Tax
- Payment
- Dividing horizontal line
- Many more ...

The idea behind this is that we can interchange the sections. So when generating a new receipt we can use the layout of the "logo" section from "grocery shop 1" and of the "product table" section from "restaurant 3". This results in more variation in the generated receipts. Additionally we can rearrange the sections so that the order of the sections is not always the same and we can add or remove sections from some receipts.

Content generation

Now that we can generate an innumerable amount of different realistic-looking receipts layouts, we also need randomly generated content. To do this we used real shop names and product names which were selected randomly. We generated somewhat realistic prices which we rounded to .05 precision to resemble prices in Swiss francs. We also simulated discounted items even though the discounts will not be shown in the output of our model.

We also randomly generate content which is not relevant for the extraction task of this project. For example we generate different addresses, numbers which are displayed in the payment section, tax rates, and everything else which might be present on a receipt. We basically randomized every value present on a receipt which would differ from any two different receipts in reality.

In this step we only create one big data construct containing all these values. From this construct we late fill in the HTML file and create the label.

Bringing everything together

To generate the final HTML file we first randomly select which sections to include, generate the entire dynamic content, fill in the content into each respective section and concatenate all sections together.

For the CSS styles we use a base file in which we randomly select values for some parameters like: padding, font size, inner widths, content alignment.

And finally to render the resulting receipt we use a library which opens the file in a browser, takes a screenshot and returns us a picture of the result.

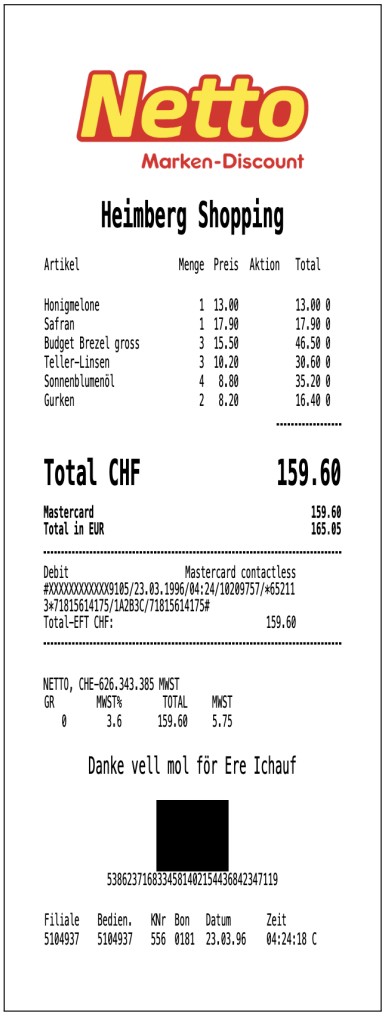
Netto
Marken-Discount
Heimberg Shopping
Artikel Menge Preis Aktion Total
Honigmelone 1 13.00 13.00 0
Safran 1 17.90 17.90 0
Budget Brezel gross 3 15.50 46.50 0
Teller-Linsen 3 10.20 30.60 0
Sonnenblumenöl 4 8.80 35.20 0
Gurken 2 8.20 16.40 0
Total CHF 159.60
Mastercard 159.60
Total in EUR 165.05
Debit Mastercard contactless
#XXXXXXXXXXXX9105/23.03.1996/04:24/10209757/*65211
3*71815614175/1A2B3C/71815614175#
Total-EFT CHF: 159.60
NETTO, CHE-626.343.385 MWST
GR MWST% TOTAL MWST
0 3.6 159.60 5.75
Danke vell mol för Ere Ichauf
538623716833458140215443684234711 9
Filiale Bedien. KNr Bon Datum Zeit
5104937 5104937 556 0181 23.03.96 04:24:18 C

Figure 2: A randomly generated receipt in its raw form without any further processing

2.2.2 Projection and quality reduction

With the previous steps we have an image of a generated receipt. But this receipts looks way to perfect. It has no errors, no noise, no bad quality and just looks to clean. So to make it look realistic we have to make it look worse.

The first step to make the receipt look more realistic is to add printing errors to the text. Without this step the text would be to crisp but in reality receipts often have quite a lot of printing errors. To simulate printing errors we generate a Gaussian noise layer. This noise layer is made translucent by converting the color channel to the alpha channel. The color channel is replaced with the color white which is the background color. Then we apply the noise to the original receipt image. This has the result that the text will have holes and errors in the color of the background but the background is not affected by the noise.

Next we add the most prominent printing error which is present on most receipts. Often receipts have vertical lines across them in which nothing is printed. So all text is interrupted by a small but noticeable line. To simulate this we select a random amount of columns in the image and overwrite all pixel in this column to be fully white.

Figure 3: A randomly generated receipt in its raw form without any further processing

At this point the colors of the image are still not quite realistic. When photographing a receipt the white of the background will never be fully white with RGB(255, 255, 255) and the color of the text never fully black with RGB(0, 0, 0). To improve this we scale the color values to be between 75-200 on each channel. So we convert them to gray values which look almost black and almost white.

The next problem is that the generated image is a perfect cut out of the receipt. When handling real data we have to cut out the receipt from the image through our preprocessing pipeline. To be able to treat the generated data in the same way, we have simulate the receipt to lay on a table, so it has to run through the same preprocessing pipeline as the read data has to. We cannot just add a background around the image because this wouldn't look realistic and would make the cut out process too easy. To properly simulate this we project the receipt including the background into a 3 dimensional space and then simulate taking a picture of the projected receipt. This allows us to add random tilt / rotation it in all 3 dimensions as well as using different zoom values.

At this stage the receipts looks quite realistic already, but it still looks too much like a computer generated image. So in a last step we add another noise layer but this time we use the normal Gaussian noise with different colors. This will ensure that the preprocessing is less perfect and also that some noise is added on the white background of the receipt.

Figure 4: The generated receipt projected onto a wooden table

2.3 Data preprocessing and quality control

2.3.1 Preprocessing pipeline

The first step of the preprocessing pipeline is to identify the receipt in the image and to cut it out.

To do this we start off by applying the Canny algorithm for edge detection on the greyscale image as well as on each RGB color channel. Then we concatenate all images with detected edges bitwise. We need to also apply the edge detection because on some backgrounds the receipt and the background share a very similar gray tone even though the color value is vastly different. To still be able to identify all edges properly we apply the edge detection multiple times on different channels.

Next, we apply morphological closing on the image including the identified edges. After that, we extract all contours. Contours are defined as neighboring pixels, using an 8-neighbor relationship of a pixel with the same intensity. Because sometimes the contour has gaps we cannot simply select the largest one to find the contour of the receipt. Instead we need to do further processing.

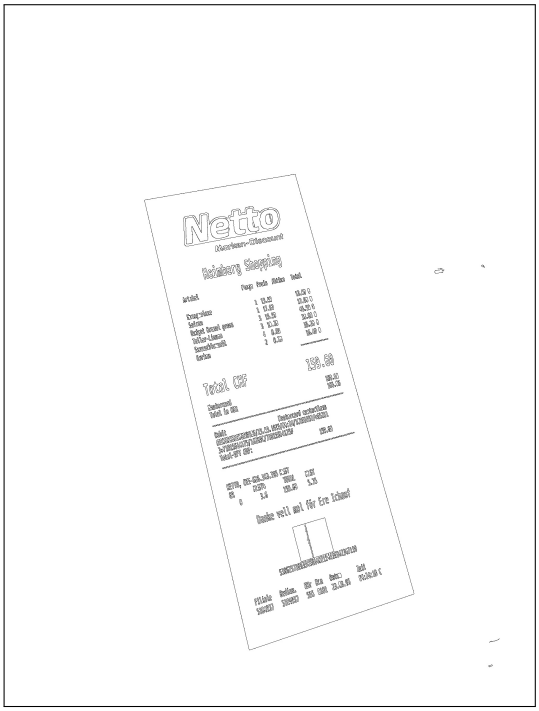

Figure 5: The identified edges of the image containing the receipt

The first step of this is to filter out all contours which are too small and considered irrelevant. This is evaluated using different thresholds for the length and diameter of the contour. The length threshold is ⅓ of the height of the image and the diameter threshold is ¼ of the height of the image. Only contours which are over these thresholds are kept while the others are removed.

Now that we only have the relevant contours we can identify the receipt. To do this we calculate the convex hull over all remaining contours. This hull should now be a mask which perfectly matches the receipt.

Figure 6: The extracted receipt

In the next step we simplify the hull to only keep 4 corners. These corners are the most prominent ones with the largest angle respectively the ones closest to a right angle. Using these corners we apply a wrap transform to convert the quadrilateral of the hull to a rectangle. The width of this rectangle is fixed to 800 pixels. This value was chosen because real receipts have a width of 8 cm. The height of the resulting rectangle containing the receipt is calculated with an approximated ratio between the width and height of the original quadrilateral of the hull. The result of this is the cut out of the receipt with the background removed and the original ratio reconstructed as good as possible.

The second step of the preprocessing pipeline is now to improve the readability of the receipt. We need to do this because different lighting conditions may lead to bad readability. Additionally we want to remove shadows that fall onto the receipt and ensure that the color shade is consistent so that processing is easier for our model.

To achieve this we first convert the image to only use one gray color channel then we apply a local contrast filter across the entire image. This filter works by calculating the mean of the neighboring pixels in a 15-pixel radius. Then we push the pixel more into its extreme. If the pixel value is close to the mean no large change is done to the pixel but the farther it is away from the mean the more it is adjusted. If the value is much darker than the mean it gets even darker and vice versa.

After applying this local contrast filter we cap all values to be between 0-255 again. The result of this is that the text of the receipt is fully black and the text has outlines which are fully white. The background further away from any text remains unchanged. The idea of this process is that the text in which we are interested is fully black and the background in which we are not interested consist of any shade of gray. The next step is to simply take all pixels which are above the threshold of 50 and set them to be white. This leaves us with the receipt only containing black text on a white background.

Figure 7: The receipt after preprocessing

2.3.2 Data cleaning

For all data that we generate synthetically the data cleaning procedures can be omitted. During the generation process, we can precisely specify the desired data output characteristics, ensuring consistent quality and format across all generated samples.

Real-life data cleaning

For the test dataset consisting of real-world receipt images, extensive data cleaning and preprocessing were necessary. We identified and removed severely degraded samples, including receipts that were:

- Physically damaged or corrupted beyond recognition
- Captured under inadequate lighting conditions
- Positioned such that critical text regions were illegible even to human annotators

Following this quality filtering process, we retained only high-quality images for evaluation. The selected images then underwent the standardized preprocessing pipeline.

2.4 Fine-Tuning

2.4.1 General Process

For model fine-tuning, we utilized a custom dataset organized in a standardized format compatible with the HuggingFace ecosystem. The HuggingFace library provides the `load_dataset` function, which enables loading datasets from either the HuggingFace Hub repository or local directories. During the synthetic data generation phase, we structured our data to ensure seamless integration with this loading mechanism, allowing direct access from the local directory structure.

The dataset organization follows the standard machine learning convention with separate training and validation splits. The directory structure is organized as follows:

```
dataset_root
├── train
│   ├── 001.jpg
│   ├── 002.jpg
│   ├── 003.jpg
│   ├── 004.jpg
│   ├── ...
│   └── metadata.jsonl
└── validation
    ├── 001.jpg
    ├── 002.jpg
    ├── 003.jpg
    ├── 004.jpg
    ├── ...
    └── metadata.jsonl
```

The `metadata.jsonl` files contain the ground truth annotations in JSON Lines format, where each line corresponds to a single training example with its associated image filename and target output structure. This format ensures efficient data loading and processing during training.

Rather than implementing custom training loops, we leveraged high-level APIs from PyTorch Lightning [**pytorch_lightning**]. This approach provides several advantages including automatic mixed precision training, distributed training support, and comprehensive logging capabilities, while reducing the likelihood of implementation errors.

2.4.2 Donut Model Fine-Tuning

The Donut (Document Understanding Transformer) model [**naver-clova-ix/donut-base**] employs a BART-based language decoder that was originally pre-trained on Markdown (HTML-like) formatted text. Since our target output format is JSON, we implemented a format conversion strategy to bridge this gap. Specifically, we converted JSON structures to XML-like representations

and incorporated special tokens into the decoder vocabulary, necessitating an expansion of the vocabulary size to accommodate these domain-specific tokens.

The fine-tuning process initially presented significant challenges, as the model failed to converge during early training attempts. After extensive debugging and community research, we identified version compatibility issues as the root cause. Multiple GitHub users reported similar training failures, with successful resolution achieved through downgrading specific package versions. This suggests that the Donut model, originally developed in 2022, has dependencies that are incompatible with newer package versions due to breaking changes in the underlying frameworks.

The successful fine-tuning configuration required the following specific package versions:

```
transformers==4.25.1
pytorch-lightning==1.8.5
timm==0.5.4
```

Listing 2: Required Package Versions for Donut Fine-Tuning

The transformers library downgrade proved particularly challenging due to cascading dependency conflicts, requiring careful resolution of transitive dependencies to maintain a stable environment. We recommend using a dedicated virtual environment or containerized setup to avoid conflicts with other project dependencies.

Fortunately, the original Donut authors provided comprehensive training and evaluation scripts under the MIT license [2], implemented using the PyTorch Lightning framework. These scripts include built-in accuracy calculation mechanisms and standardized evaluation protocols. We adapted the provided codebase with minimal modifications to the data processing pipeline to accommodate our specific dataset format and output requirements.

The base Donut model supports three primary task categories:

- **CORD (Consolidated Receipt Dataset)**: Structured document parsing for receipt information extraction
- **RVL-CDIP (Ryerson Vision Lab Complex Document Information Processing)**: Document classification across 16 categories
- **DocVQA (Document Visual Question Answering)**: Visual question answering on document images

The CORD task configuration is most relevant to our use case, as it specializes in extracting structured information from receipts and outputting results in JSON format. However, initial evaluation of the pre-trained CORD model on our dataset revealed significant performance limitations.

The baseline model performance was inadequate for our specific requirements due to several

```
{
  "company": "WALMART",
  "date": "2021-12-15",
  "address": "123 Main St",
  "total": {
    "total_price": "45.67",
    "cash_price": "50.00",
    "change": "4.33"
  },
  "menu": [
    {"nm": "Bread", "price": "2.99"},
    {"nm": "Milk", "price": "3.49"}
  ]
}
```

Listing 3: Example Output from Base Donut CORD Model

factors:

1. **Dataset Complexity Gap**: The base model was trained on simpler, more standardized receipt formats, whereas our dataset contains more diverse and complex document structures.
2. **Image Quality Issues**: Many receipt sections in our dataset exhibit blur, low resolution, or poor contrast, challenging the model's OCR capabilities.
3. **Schema Mismatch**: The JSON key structures produced by the base model do not align with our target schema requirements.
4. **Domain Adaptation**: The base model lacks exposure to the specific visual and textual patterns present in our target domain.

Quantitative evaluation revealed a baseline accuracy of approximately 0% on our validation set, measured using exact match accuracy for the complete JSON structure. This low performance motivated the need for comprehensive fine-tuning on our domain-specific dataset.

Detailed accuracy metrics and evaluation methodologies are discussed in Model Validation Section.

2.5 Model validation

The Donut trainer framework incorporates comprehensive validation accuracy evaluation mechanisms through the implementation of two distinct but complementary accuracy measurement functions. These functions employ similar underlying principles while serving different evaluation purposes within the model assessment pipeline.

2.5.1 Validation Accuracy Evaluation

For validation accuracy assessment, we employed the Normalized Edit Distance (NED) metric, which is computed using the Levenshtein distance algorithm implemented through the Natural Language Toolkit (NLTK) library. The Levenshtein distance, also known as edit distance, quantifies the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another.

The fundamental principle underlying the Levenshtein metric is the quantification of transformation costs associated with different edit operations. Each operation type—insertion, deletion, and substitution—incurs a specific cost penalty. Consequently, a higher Levenshtein distance score indicates that more edit operations are required to achieve textual equivalence between the predicted and ground truth outputs. In this context, elevated scores represent degraded model performance, while lower scores indicate superior accuracy. The metric can be formally expressed as:

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

2.5.2 Test Accuracy Evaluation

For test accuracy evaluation, we utilized the JSONParseEvaluator (implementation reference), which extends the Levenshtein distance methodology for structured JSON data comparison. This evaluator maintains the core Levenshtein distance computation while providing enhanced functionality for direct JSON object comparison, eliminating the need for preliminary string conversion processes.

The key distinction of the JSONParseEvaluator lies in its ability to process JSON structures natively, enabling more sophisticated comparison of hierarchical data representations. Following distance calculation, the raw scores undergo tree normalization to account for structural differences in the JSON hierarchy. The final accuracy metric is computed using the normalized Tree Edit Distance (nTED) formula:

$$\text{Accuracy} = \max(1 - \text{nTED}, 0) \tag{1}$$

This formulation ensures that the output metric follows an intuitive interpretation paradigm where higher scores correspond to superior model performance. The subtraction from unity (1 - nTED) transforms the distance-based penalty into an accuracy-based reward, while the maximum operation with zero prevents negative accuracy values in cases of extreme prediction errors.

The normalization process accounts for variations in document structure complexity, ensuring fair comparison across diverse input samples with varying JSON tree depths and branching factors. This approach provides a robust evaluation framework particularly suitable for document understanding tasks where both textual accuracy and structural fidelity are critical performance indicators.

2.5.3 Validation Accuracy during Training

Epoch	Edit Distance metric	Training Loss
0	0.0293	0.1222
1	0.0023	0.0122
2	0.0013	0.0003

Table 1: Training Metrics

Training Performance Analysis

The training metrics presented in Table 1 demonstrate rapid model convergence during the initial training phase. The validation accuracy exhibits a substantial improvement from epoch 0 to epoch 1, with the accuracy score decreasing from 0.0293 to 0.0023, representing an approximate 92% improvement in model performance. This significant enhancement indicates that the model rapidly learns the underlying patterns in the validation dataset during the first epoch.

It is important to note that since the validation accuracy is computed using the Normalized Edit Distance metric (as described in the previous section), lower numerical values correspond to superior model performance. The observed decrease in validation accuracy scores therefore represents improved predictive capability rather than performance degradation.

The training loss similarly demonstrates convergent behavior, decreasing from 0.1222 in the initial epoch to 0.0122 after the first epoch, indicating effective gradient-based optimization. The continued reduction to 0.0003 by epoch 2 suggests that the model is successfully minimizing the training objective function, though careful monitoring for potential overfitting is recommended as training progresses.

The rapid convergence observed in both metrics suggests that the pre-trained Donut model architecture effectively leverages transfer learning principles, enabling quick adaptation to the target domain with minimal fine-tuning epochs. This behavior is characteristic of well-initialized transformer-based models that have been pre-trained on large-scale datasets with similar structural properties to the fine-tuning task.

2.6 Model Output Analysis

The validation example presented in Listing 4 demonstrates the Donut model's capability to accurately parse and structure information from German grocery receipts. The model successfully extracts all essential receipt components including merchant identification (`s_shop_name`) transaction date (`s_date`), itemized product information (`s_products`), and total amount (`s_total`).

```
Prediction:
<s_shop_name>Penny</s_shop_name>
<s_date>07/05/24</s_date>
<s_products>
    <s_name>Rindsschnitzel</s_name><s_amount>1</s_amount>
    <s_price>7,4</s_price>
    <sep/>
    <s_name>Langkornreis</s_name><s_amount>1</s_amount>
    <s_price>3,7</s_price>
    <sep/>
    <s_name>Rosmarin</s_name><s_amount>1</s_amount>
    <s_price>9,0</s_price>
</s_products>
<s_total>20,7</s_total>

Answer:
<s_shop_name>Penny</s_shop_name>
<s_date>07/05/24</s_date>
<s_products>
    <s_name>Rindsschnitzel</s_name><s_amount>1</s_amount>
    <s_price>7,4</s_price>
    <sep/>
    <s_name>Langkornreis</s_name><s_amount>1</s_amount>
    <s_price>3,7</s_price>
    <sep/>
    <s_name>Rosmarin</s_name><s_amount>1</s_amount>
    <s_price>9,0</s_price>
</s_products>
<s_total>20,1</s_total>

Normed ED: 0.002994011976047904
```

Listing 4: Log output of validation

2.6.1 Structured Information Extraction Performance

The model exhibits excellent performance in extracting structured data from the input receipt image. Key achievements include:

- **Perfect merchant and date recognition**: The shop name "Penny" and date "07/05/24" are extracted with 100% accuracy.
- **Comprehensive product detail extraction**: All three products (Rindsschnitzel, Langkornreis,

Rosmarin) are correctly identified with their respective quantities and individual prices.

- **Proper hierarchical structuring**: The XML output maintains the correct nested structure for product listings using appropriate separator tokens (`<sep/>`).

2.6.2 Error Analysis

The sole discrepancy occurs in the total amount calculation, where the model predicts €20.7 while the ground truth indicates €20.1. This represents a €0.6 difference, which could result from:

1. Optical character recognition ambiguity in decimal digit interpretation
2. Potential rounding differences in multi-currency or tax calculations
3. Ground truth annotation inconsistencies

The Normalized Edit Distance of 0.00299 confirms that this minor discrepancy has minimal impact on overall extraction quality. The corresponding accuracy score of 99.7% demonstrates near-perfect performance, indicating that the model successfully generalizes to real-world German receipt formats with high fidelity.

This validation example exemplifies the model's robust capability to handle multilingual content (German product names) and complex structured document layouts typical of retail receipt formats.

3 Results

Test Dataset	Accuracy (1 - nTED)	F1-Score
Test Set 1	0.99	0.95
Test Set 2	0.80	0.70
Test Set 3	0.31	0.28
Test Set 4	0.08	0.03

Table 2: Testset Evaluation

3.1 Performance Analysis

The evaluation results presented in Table 2 demonstrate a systematic degradation in model performance as test conditions progressively deviate from the training distribution. The results reveal clear patterns of domain adaptation challenges across four distinct complexity tiers.

Synthetic-to-Real Domain Transfer Analysis

Optimal Synthetic Performance (Test Set 1): The model achieves near-perfect performance on synthetic data with 99% accuracy and 95% F1-score. This exceptional performance confirms that the model has successfully learned the underlying patterns present in the training distribution and can accurately process data with similar characteristics and controlled variations.

Initial Real-World Adaptation (Test Set 2): Performance degrades moderately when transitioning to real-world receipts from similar shops, with accuracy dropping to 80% (19% decrease) and F1-score to 70% (25% decrease). This degradation reflects the inherent complexity introduced by real-world factors such as print quality variations, paper conditions, and scanning artifacts that are not fully captured in synthetic data generation.

3.1.1 Layout Generalization Capabilities

Cross-Merchant Generalization (Test Set 3): When evaluating on receipts from completely different shops while maintaining similar layouts, accuracy drops significantly to 31% (50% decrease from Test Set 2). However, the F1-score of 28% indicates that the model retains reasonable field-level extraction capabilities despite struggling with overall sequence accuracy. This divergence suggests that layout familiarity is crucial for optimal performance, while individual text recognition remains partially robust.

Complete Layout Divergence (Test Set 4): The model exhibits critical failure when confronted with both different merchants and novel layouts, achieving only 8% accuracy and 3% F1-score. This catastrophic performance drop indicates that the model has not developed sufficient invariance to handle fundamental layout variations combined with image quality degradation.

3.1.2 Robustness and Generalization Insights

The progressive performance degradation reveals several key insights about model robustness:

1. **Synthetic-Real Gap**: The 19% accuracy drop from synthetic to real-world data highlights the importance of incorporating realistic image artifacts and variations during training data generation.
2. **Layout Dependency**: The dramatic performance reduction when layout changes (Test Set 3 vs. Test Set 2) indicates strong model dependency on spatial feature patterns learned during training.
3. **Compound Challenge Effects**: Test Set 4 demonstrates that combining multiple challenging factors (layout changes + image distortions) results in multiplicative rather than additive performance degradation.
4. **Metric Behavior Patterns**: The divergence between accuracy and F1-score in Test Set 3 suggests that sequence-level accuracy is more sensitive to domain shift than individual field extraction capabilities.

3.1.3 Implications for Model Development

These results highlight critical areas for improvement in model architecture and training methodology:

- **Training Data Diversification**: The performance gap between synthetic and real-world data emphasizes the need for more diverse and realistic training samples.
- **Layout Invariance**: The sensitivity to layout changes suggests potential benefits from incorporating spatial attention mechanisms or layout-agnostic architectures.
- **Robustness Enhancement**: The complete failure under challenging conditions indicates the need for adversarial training or data augmentation strategies to improve model resilience.

3.2 Model comparison & Baseline model

3.2.1 Model Performance Comparison and Alternative Approaches

Fine-tuned Donut vs. Base Model Performance

To evaluate the effectiveness of our fine-tuning approach, we conducted a comprehensive comparison between our domain-adapted Donut model and the original pre-trained Donut-base model. The evaluation was performed on our validation dataset using standardized metrics including exact match accuracy and F1 score for structured output matching.

The base model's complete failure (0% accuracy and F1 score) on our dataset confirms the critical importance of domain-specific fine-tuning. This zero performance can be attributed

to several factors:

- **Schema Incompatibility**: The base model's output format does not align with our target JSON schema
- **Domain Gap**: Significant differences between the training data (CORD dataset) and our specific document types
- **Visual Pattern Mismatch**: The model fails to recognize layout patterns and textual elements specific to our dataset
- **Vocabulary Limitations**: Missing domain-specific terminology and formatting conventions

Alternative Model Exploration: SmolVLM

As part of our comprehensive model evaluation strategy, we investigated SmolVLM, a compact Vision-Language Model (VLM) with 2 billion parameters. SmolVLM represents a different architectural approach compared to Donut, employing a more general-purpose vision-language framework that combines visual understanding with natural language generation capabilities.

The motivation for exploring SmolVLM included:

1. **Architectural Diversity**: Testing alternative approaches beyond the encoder-decoder paradigm of Donut
2. **Parameter Efficiency**: Evaluating whether a smaller model could achieve competitive performance
3. **Generalization Capability**: Assessing the model's ability to handle diverse document formats without extensive fine-tuning

SmolVLM Baseline Performance Initial evaluation of the pre-trained SmolVLM model on our dataset yielded the following baseline metrics:

Evaluation Metric	Score
Tree Edit Distance	0.287
F1 Accuracy	0.262

Table 3: SmolVLM Baseline Performance Metrics

SmolVLM Fine-tuning Challenges and Discontinuation

Despite the promising baseline performance, our attempts to fine-tune SmolVLM encountered significant technical and performance-related obstacles:

- **Training Instability**: The fine-tuning process exhibited convergence issues, with loss values fluctuating erratically and failing to achieve stable reduction over training iterations
- **Performance Degradation**: Contrary to expectations, fine-tuned models demonstrated

worse performance than the baseline, suggesting potential overfitting or inappropriate hyperparameter configurations

- **Resource Requirements**: The model's memory footprint and computational demands exceeded our available resources for extensive hyperparameter exploration
- **Documentation Limitations**: Limited fine-tuning documentation and community support made troubleshooting difficult compared to the well-established Donut ecosystem

Given these substantial challenges and the superior baseline performance of fine-tuned Donut models, we decided to discontinue the SmolVLM approach and focus our efforts on optimizing the Donut fine-tuning pipeline. This decision was further supported by:

1. **Proven Track Record**: Donut's established success in document understanding tasks
2. **Community Support**: Extensive documentation and troubleshooting resources available
3. **Reproducible Results**: Consistent fine-tuning outcomes reported in literature
4. **Resource Efficiency**: Better performance-to-resource ratio for our specific use case

This comparative analysis reinforced our confidence in the Donut-based approach while providing valuable insights into alternative model architectures for future research directions.

3.3 Computational Infrastructure and Cloud Setup

The computational requirements for deep learning model training necessitated access to GPU-accelerated hardware. After evaluating multiple cloud service providers, including Google Cloud Platform and Microsoft Azure, we selected GPU Hub, which is provided as part of the institutional resources available to students at Hochschule Luzern (HSLU). This decision was primarily driven by cost-effectiveness considerations, as commercial cloud GPU instances would have incurred substantial expenses that exceeded the project's budget constraints.

The institutional GPU Hub platform provided adequate computational resources for our research objectives while eliminating financial barriers that could have limited experimental scope and model iteration cycles.

3.3.1 Hardware Specifications

The computational experiments were conducted on GPU Hub infrastructure featuring the following specifications:

- **GPU Model**: NVIDIA A16 Tensor Core GPU
- **Video Memory**: 16 GB GDDR6 VRAM
- **Architecture**: Ampere-based compute capability
- **Memory Bandwidth**: High-speed memory interface optimized for deep learning workloads

The A16 GPU architecture provides sufficient computational throughput and memory capacity for training and evaluating document extraction models on our dataset scale. The 16 GB VRAM allocation enables processing of high-resolution document images and supports batch training with reasonable batch sizes for efficient gradient computation.

3.3.2 Model Scale and Parameter Limitations

The selection and configuration of language models for document extraction were constrained by available computational resources, particularly GPU memory limitations. Modern transformer-based architectures for document understanding, such as LayoutLM, DocFormer, and similar models, typically require substantial memory allocation due to their large parameter counts, often ranging from hundreds of millions to several billion parameters.

The 16 GB VRAM constraint of the available A16 GPU infrastructure limited the maximum model size that could be effectively trained and deployed. This hardware limitation influenced architectural decisions, requiring careful balance between model complexity and memory efficiency to achieve optimal performance within the available computational budget.

3.4 Machine Learning Operations (Deployment)

Once our model has been trained and evaluated, the next step would be to integrate it into a real-world system where it can be used reliably and at scale. In our case, the trained receipt extraction model can be deployed as part of a local or cloud-based application that takes an image of a receipt as input and returns a structured JSON object containing all relevant fields. One of the key advantages of our system is that it does not rely on any external OCR service or cloud-based inference engine by default. This allows for full offline deployment, which is especially important for use cases where data privacy is critical. For example the model can run entirely on a local server, edge device or even integrated into a desktop or mobile application - ensuring that receipt images never leave the user's environment.

Deployment can be done using existing tools such as Docker or Python-based APIs (e.g., FastAPI or Flask), allowing developers to integrate the model into their own software with minimal overhead. Thanks to the modularity of the pipeline - including preprocessing, model inference, and postprocessing - each step can be adapted or extended independently depending on the use case.

3.4.1 Practical Use Case

The receipt extraction system we developed has several practical applications. We see the largest value in the following industries:

- **Accounting and Bookkeeping:** Small businesses and accounting firms can integrate the system into their accounting software. By running locally or on a secure private server, it supports compliance with data privacy laws while reducing the need for human data entry.
- **Automated Expense Reporting:** The tool could be integrated into HR-Software or expense management platform, reducing manual entry time and minimizing errors.
- **POS Integration and Receipt Digitization:** For retail or hospitality sectors, the system could serve as a post-processing tool to digitize and archive printed receipts, especially in cases were POS systems do not generate structured digital outputs.
- **Regulatory and Tax Compliance:** The system can help both businesses and individuals keep proper digital records of receipts for things like tax returns, reimbursements, or audits. It ensures that receipt data is stored in a structured format, making it easier to find, share, or submit when needed.

In General, our model can provide value anywhere receipts need to be digitized or manually entered into a system. By automating the extraction of relevant information, it reduces the need for repetitive manual work, minimizes input errors and speeds up processing. Future enhancements like confidence scoring or human-in-the-loop validation could further improve its reliability in production settings.

4 Discussion

4.0.1 Interpretation of results

The evaluation shows that our fine-tuned Donut model achieves significantly higher accuracy scores across all testsets. These results demonstrate that the use of synthetic training data can lead to substantial performance gains, which confirms our hypothesis. The evaluation results from our different test categories also reveals that the model is also generalizing well, although it performs better on accustomed data.

4.0.2 Comparison with existing studies

In contrast to traditional OCR-based pipelines that rely on separate text detection and recognition stages, our project adopts an end-to-end approach using a visual-language model (Donut) trained directly on document images. While many state-of-the-art models are trained on large, diverse real-world datasets, our approach relies exclusively on synthetic data. Despite this, the fine-tuned Donut model surpassed the base models by a significant margin, showing that task-specific training via synthetic data can compete with more resource-heavy alternatives when applied correctly.

Compared to other models explored in the literature, such as LayoutLMv3 or TrOCR-based pipelines, our method avoids the complexity of bounding box annotations and multi-stage pipelines. Furthermore, while models like SmolVLM are designed for lightweight applications, our evaluation indicates that Donut's architecture currently offers better performance for document parsing tasks when fine-tuned.

Suggestions for Future Work

While our current approach showed strong results, especially considering that the model was trained entirely on synthetic data, there are several directions worth exploring to further improve performance and especially flexibility.

- **Expanding the Synthetic Generator:** Even though the training data was already diverse, expanding the generator to support more edge cases (e.g., multilingual receipts, handwritten annotations, and very long receipts) would make a real-world application using this model attractive for even more people.
- **Confidence Estimation and Error Detection** In a real-world setting, users need to trust the extracted output. Adding mechanisms to estimate prediction confidence or flag likely extraction errors would make the system more robust and reliable.
- **Postprocessing with Rule-based Validation:** Integrating simple logic (e.g., total must equal sum of items, tax must match percentage ranges) could help catch or correct model errors without needing human review for every result.

- **Real-world Deployment Testing:** A next logical step would be to test the system as part of a larger pipeline, for instance importing receipts directly into accounting- or tax software to evaluate real-world usability.

4.1 Conclusions

The goal of this project was to explore whether an AI model could be fine-tuned to extract structured data from receipts using only synthetically generated training data. To address this challenge we designed a custom data generation pipeline capable of producing realistic-looking receipt images, along with corresponding structured labels.

Our findings confirm that a fine-tuned model significantly outperforms baseline models when applied to real-world receipt images in inference. This validates the effectiveness of our synthetic data approach and demonstrates that models like this can be successfully trained without access to large annotated datasets. Given that the synthetic generation process can be expanded and diversified with minimal cost, we believe this method has strong potential for practical applications in environments where data privacy, scalability, or labeling resources are a concern.

4.2 Reflexion

Looking back at this project, we're proud of how much we were able to achieve. One of the most valuable aspects was learning how to design a complete machine learning pipeline from scratch. From generating synthetic receipts to fine-tuning a vision-language model, we gained hands-on experience across the entire AI workflow.

Throughout this project we encountered and overcame several practical challenges, including model fine-tuning issues, mismatches in expected input formats, and unexpected behaviors during training. Solving these required a more in-depth understanding of the tools we were using, which significantly improved our technical proficiency and and problem-solving skills.

Overall, this project challenged us in the right way and gave us the confidence to tackle similar applied AI problems outside of the school environment.

5 Appendix

5.1 References

[1] He Guo, Xiameng Qin, Jiaming Liu, et al. “EATEN: Entity-aware Attention for Single-Shot Visual Text Extraction”. In: *arXiv preprint arXiv:1909.09380* (2019).
[2] Geewook Kim et al. “OCR-Free Document Understanding Transformer”. In: (2022). URL: https://github.com/clovaai/donut.
[3] Mindee. “docTR: End-to-End Document Text Recognition”. In: *GitHub* (2022).
[4] Oshri Naparstek et al. “KVP10k: A Comprehensive Dataset for Key-Value Pair Extraction in Business Documents”. In: *arXiv preprint arXiv:2405.00505* (2024).
[5] QiuXing Michelle Tan et al. “Information Extraction System for Invoices and Receipts”. In: *University of Glasgow technical report*. 2023.
[6] Guozhi Tang, Lele Xie, Lianwen Jin, et al. “MatchVIE: Exploiting Match Relevancy between Entities for Visual Information Extraction”. In: *arXiv preprint arXiv:2106.12940* (2021).
[7] Hongkuan Zhang, Edward Whittaker, and Ikuo Kitagishi. “Extending TrOCR for Text Localization-Free OCR of Full-Page Scanned Receipt Images”. In: *arXiv preprint arXiv:2212.05525* (2022).
[8] Qintong Zhang, Victor Shea-Jay Huang, et al. “Document Parsing Unveiled: Techniques, Challenges, and Prospects for Structured Information Extraction”. In: *arXiv preprint arXiv:2410.21169* (2024).

5.2 List of figures

5.3 Github

GitHub Project URL DSPRO2 - HSLU AI/ML Swiss Receipt Extractor

Jamian Rajakone | GitHub

Tamino Walter | GitHub

Diego Gonzalez | GitHub

5.4 Weights & Biases

Results | Weights & Biases view