Internship B-robots

Realization Document

Bachelor's degree in the Applied Computer Science

Ehran Lenaerts

R0661627

Academic year 2022-2023

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

Table of Contents

Introduction

B-robots, founded in 2017, has emerged as a prominent player in the field of Robotic Process Automation (RPA). With their deep expertise and innovative approach, they have swiftly established a strong foothold in this dynamic and competitive market. B-robots excels in providing comprehensive solutions across various domains, empowering businesses to streamline their processes and achieve operational efficiency.

Their main service, Robotic Process Automation (RPA), enables the automation of repetitive tasks using software robots. This empowers organizations to reduce manual effort, improve accuracy, and enhance productivity.

In addition to RPA, B-robots branched out into AI automation, which combines the power of Artificial Intelligence and automation to create intelligent workflows. Leveraging cutting-edge algorithms and machine learning techniques, they enable businesses to automate complex decision-making processes, thereby accelerating digital transformation.

Figure 1 B-robots logo

During my internship at B-Robots, I had the privilege of working on diverse projects. Throughout this experience, my primary focus revolved around two significant tasks: data extraction from documents, with a focus on invoices, and working on a computer vision model to integrate within an automated testing framework. For the latter, I collaborated with another intern from a company called Brightest. My internship at B-Robots offered me an invaluable opportunity to improve my own technical skills and knowledge in the field. In this realization document, I aim to outline and showcase the work I accomplished during my time at B-Robots, shed light on the achievements, lessons learned, and personal growth I attained throughout this journey.

1. Internship Objectives

1.1 Addressing Invoice Fraud and Data Extraction

One of the primary objectives of my internship was to research and address the issue of Invoice Fraud. Initially, the project was conceived as a Proof of Concept aimed at exploring the feasibility of creating a tool to detect Invoice Fraud on digitalized invoices so that B-robots could potentially offer this service to future customers.

To begin, I conducted an in-depth analysis and investigation into various fraudulent practices associated with invoice processing. This involved studying different types of fraudulent activities, understanding the common patterns and indicators, and exploring existing fraud prevention techniques and technologies.

However, as the project progressed, the focus shifted towards becoming a broader Data Extraction initiative for various types of documents. This expanded version of the project would incorporate an additional feature that allowed for the evaluation of the reliability and validity of important information extracted from documents, so that the original idea of combating problems such as Invoice Fraud remained at the forefront.

Thus, I worked on automating the data extraction process from invoices using a combination of techniques, including OCR (Optical Character Recognition) and NLP (Natural Language Processing). Leveraging Python, spaCy, and Google's Tesseract OCR Engine, I developed a pipeline capable of extracting key information, such as invoice numbers, dates, and amounts, from invoice documents.

1.2 Collaboration and Computer Vision Model Development

In collaboration with another intern from Brightest, we embarked on an exciting project to develop a Computer Vision model for a testing framework. Our goal was to use the power of Computer Vision to detect and locate web elements, like buttons and links, within web applications, primarily websites, using pure visual methods. These detected clickable web elements would serve as the basis for simulating real-world user interactions, mimicking their actions by accurately emulating clicks on the identified elements.

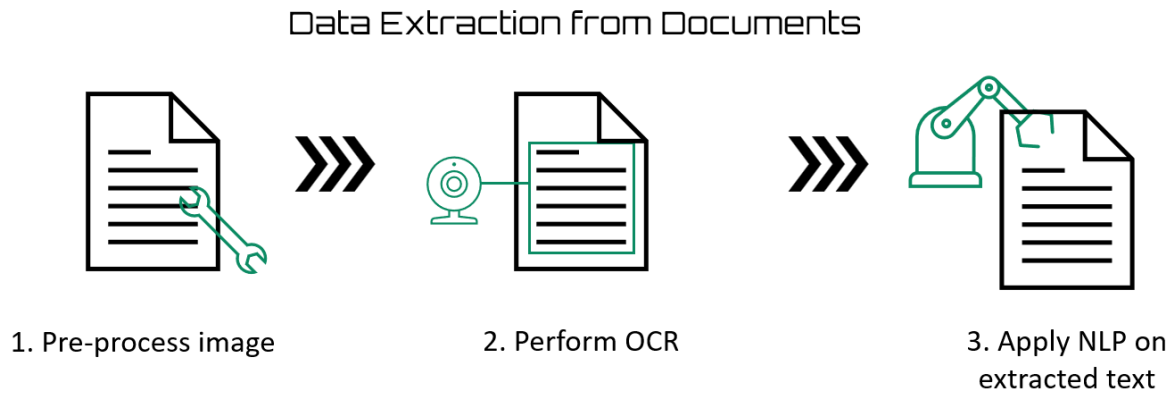
We chose for a purely visual approach to replicate real-world usage in its most authentic form, as relying solely on internal paths to find these elements could potentially lead to clicking on non-visible elements which normal users wouldn't use. By focusing on the visibility of elements, our approach would yield more realistic results during the testing phase, ensuring that the interactions were accurately performed on the web elements it identified. Later, with more data and if this PoC proved successful, the project could be expanded to other applications.

2. Projects and Tasks

In the following sections, I will provide a general technical overview of the different projects, followed by a more detailed explanation that delves deeper into the intricacies of the implementation and the different components of each task.

2.1 Document Data Extraction

Businesses and organizations rely on various types of documents, such as invoices, receipts, contracts, and other forms of paperwork, to manage their operations and finances. However, manually processing the information on these documents can be a tedious and error-prone task. To automate this process, I devised a pipeline that incorporated the Google Tesseract OCR (Optical Character Recognition) engine along with a custom-built NLP model based on the spaCy transformer template.



An annotated dataset of invoices was created by running scanned invoices through Google Tesseract. Google Tesseract is an open-source OCR (Optical Character Recognition) service meaning its accuracy will be slightly reduced. The resulting text was then manually annotated using an open-source annotation tool, which allowed me to highlight and label specific data on an invoice and export this data in a JSON format.

Accurate annotations play a pivotal role in instructing machine learning models about the specific information they need to learn. Precise and meticulous annotations provide the necessary guidance for the machine learning model, allowing it to recognize and comprehend the targeted entities or patterns effectively. By carefully and correctly annotating the data, I lay the foundation for training a robust and reliable machine learning model that can make accurate predictions and perform effectively in production scenarios.

The NLP model was created using Python and spaCy. It includes several components which are placed within one ‘spaCy pipeline’, such as text tokenization, a Transformer Listener layer, Named Entity Recognition, and a Span Categorizer, which work together to perform a complete analysis of the given invoice.

To enhance user convenience, a user interface was developed utilizing Streamlit. This interface empowers users to upload their invoices and conveniently visualize the extracted information in a user-friendly format.

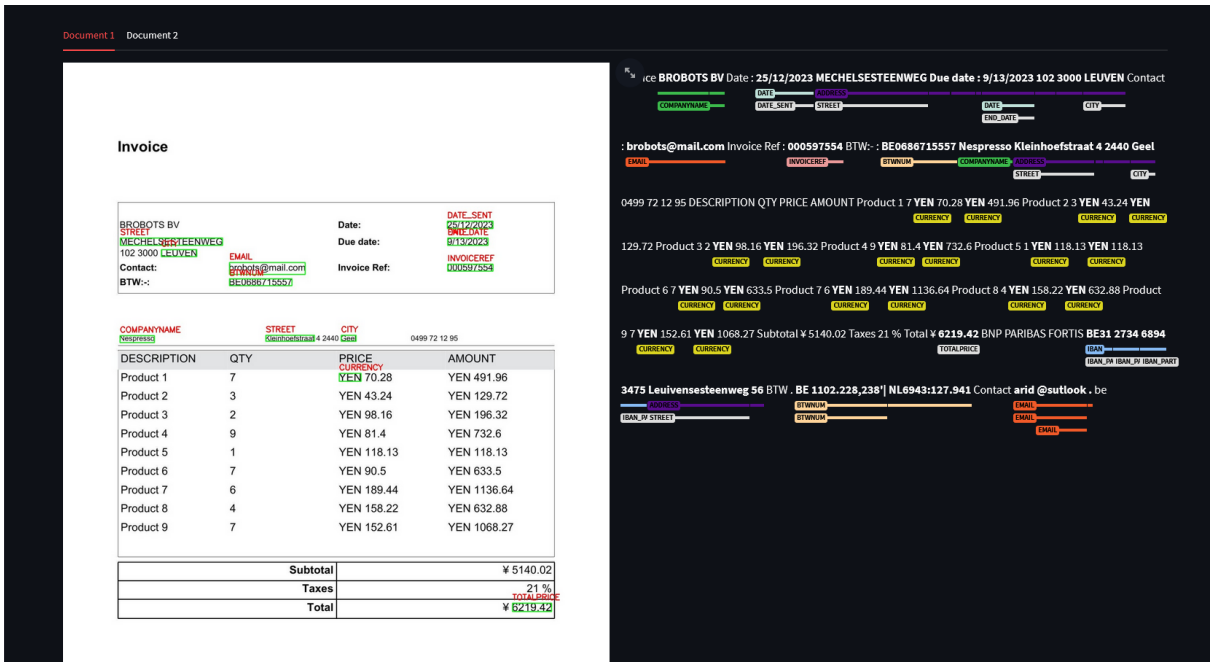
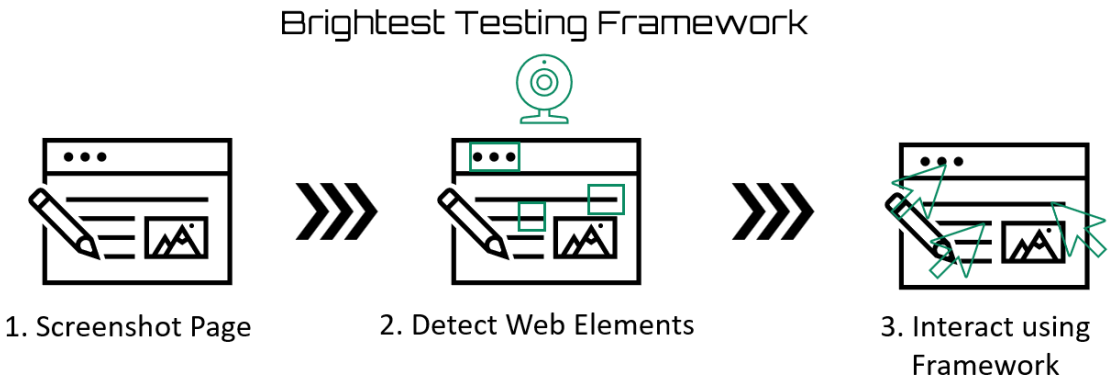


Figure 2 Example GUI image

To provide an overview, the document will be displayed on the left side of the interface, while the various elements will be visually represented on top of it. On the right side, the extracted text and the corresponding labels assigned to different entities will be presented. This arrangement offers a clear depiction of the resulting text generated by the OCR engine, as well as the distinct entities identified through the NLP model. By visually mapping the extracted information alongside the document, users can easily grasp the output of both the OCR engine and the NLP model, gaining valuable insights into the document's content and the entities extracted.

2.2 Brightest Testing Framework

In the realm of IT, ensuring the reliability and performance of applications is paramount, which necessitates robust testing strategies. To minimize errors when testing recently modified web pages, leveraging a Computer Vision model can prove advantageous as it is not dependent on hardcoded information. Such a model can be trained to detect specific web elements that require testing and extract their precise coordinates on the web page to then use those coordinates to simulate clicking behaviour.



To achieve this, a collection of diverse screenshots showcasing various web pages and their web elements was taken. These screenshots were then annotated using Doccano, an open-source annotation software, the same annotation tool I employed in the Document Data Extraction project for labelling textual data. This annotated dataset was subsequently adjusted to adhere to the YOLO data format, enabling the utilization of the YOLOv8 model—an AI model created by Ultralytics, a company known for high-end and easy to use Computer Vision models. The YOLOv8 model, like its predecessors, sets the benchmark in the field of computer vision and seamlessly integrates with modern applications.

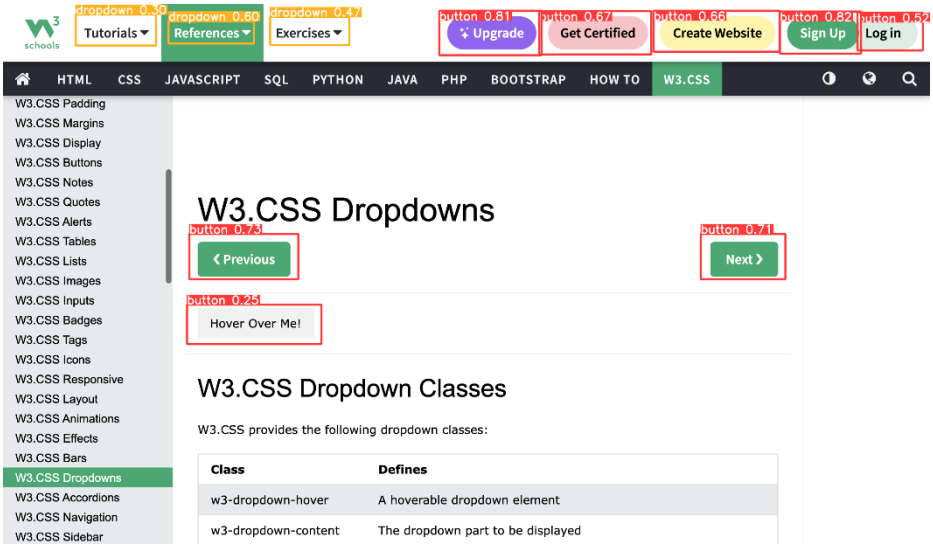


Figure 3 Result of the CV model

To ensure the model's accuracy and effectiveness, we performed extensive data cleaning, adjustment, and augmentation. This involved removing irrelevant data, adjusting

annotations, and augmenting the dataset by generating synthetic examples to address data scarcity issues for certain web elements. By meticulously preparing and refining the dataset, we aimed to train a robust Computer Vision model capable of accurately detecting and analyzing web elements, empowering developers with a powerful testing tool within the Brightest Framework.

Afterwards I handed the trained model and the code required to use it to my colleague intern at Brightest, who then implemented it within his application so that it could be added to the Brightest Framework.

2.3 Query Screen

During my internship, I also undertook a small side project that involved creating a query screen to be integrated within a (RPA) bot. While this task was initially outside the scope of my internship responsibilities, I took the initiative to offer my assistance and expressed my willingness to contribute and I was granted the opportunity to work on this aspect, despite it not being initially assigned to me. This experience highlights the flexibility and adaptability I strive to embody within a professional environment. By embracing new challenges and taking on additional responsibilities, I aim to continuously grow and contribute effectively to the organization's goals.

This solution, developed using JavaScript and PHP, aimed to facilitate the querying of an SQL database. By implementing this feature, I contributed to enhancing the functionality and usability of the bot, providing users with a streamlined interface to interact with the underlying database.

3. In-Depth Overview

In the following sections, I will delve into the details of the implemented solutions for both the Data Extraction and the Brightest CV Model. For these sections we will not be going over the Query Screen as this was only a small part of the internship. These two sections will offer a comprehensive explanation of the step-by-step process involved in creating the solutions, highlighting the key considerations, methodologies, and technical intricacies. Hopefully this will help in providing a more in-depth look into the methodologies and techniques employed to address the challenges at hand.

3.1 Document Data Extraction

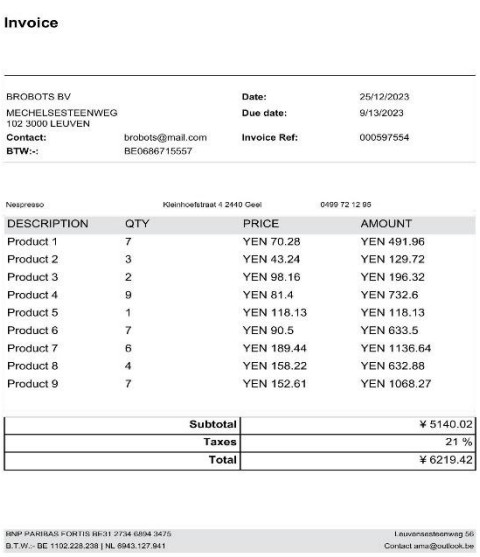
Initially, as part of the data extraction process, the conversion of invoice images into text was a critical step. To accomplish this, I developed a Python pipeline that involved using a PDF converter to convert the invoices into JPG format. Before proceeding with Optical Character Recognition (OCR), I implemented pre-processing techniques to enhance the quality of the images and subsequently extracted the data from the processed images.

Step 1: Dummy data generation

An inherent challenge encountered during the project was the scarcity of available data for the company's purposes. Upon conducting online research for publicly accessible datasets, various document collections, including invoices, were discovered. However, a significant drawback was that the majority of these datasets comprised outdated documents. While these datasets could have sufficed if the focus was solely on analyzing layout structures, the text and information contained within them were often obsolete. To address this limitation, I took the initiative to develop a Python script capable of generating additional data, thereby augmenting the training dataset beyond the provided resources. This approach allowed for a more comprehensive and robust dataset, enabling improved model performance and accuracy.

Step 2: Preprocessing

During the pre-processing stage, the images underwent grayscaling followed by the application of a binary threshold. To preserve the black elements while accounting for varying shades of gray, a threshold value of 220 and a maximum value of 255 were chosen. This crucial step prevented the potential loss of inner text within tables present in the



Invoice

BROBOTS BV
MECHELSESTEENWEG
102 3000 LEUVEN
Contact: brobots@mail.com
BTW:-: BE0686715557

Date: 25/12/2023
Due date: 9/13/2023
Invoice Ref: 000597554

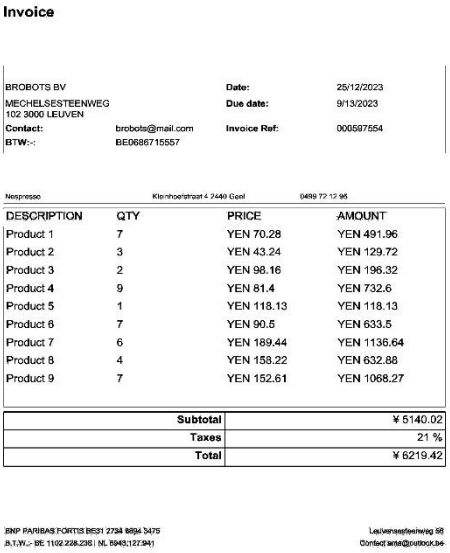
Nespresso Kleinhoefstraat 4 2440 Geel 0499 72 12 95

DESCRIPTION	QTY	PRICE	AMOUNT
Product 1	7	YEN 70.28	YEN 491.96
Product 2	3	YEN 43.24	YEN 129.72
Product 3	2	YEN 98.16	YEN 196.32
Product 4	9	YEN 81.4	YEN 732.6
Product 5	1	YEN 118.13	YEN 118.13
Product 6	7	YEN 90.5	YEN 633.5
Product 7	6	YEN 189.44	YEN 1136.64
Product 8	4	YEN 158.22	YEN 632.88
Product 9	7	YEN 152.61	YEN 1068.27

Subtotal	¥ 5140.02
Taxes	21 %
Total	¥ 6219.42

BNP PARIBAS FORTIS BE31 2734 6894 3475
B.T.W.:- BE 1102.228.238 | NL 6943.127.941
Leuvensesteenweg 56
Contact:ama@outlook.be

Figure 4 Original Document



Invoice

BROBOTS BV
MECHELSESTEENWEG
102 3000 LEUVEN
Contact: brobots@mail.com
BTW:-: BE0686715557

Date: 25/12/2023
Due date: 9/13/2023
Invoice Ref: 000597554

Nespresso Kleinhoefstraat 4 2440 Geel 0499 72 12 95

DESCRIPTION	QTY	PRICE	AMOUNT
Product 1	7	YEN 70.28	YEN 491.96
Product 2	3	YEN 43.24	YEN 129.72
Product 3	2	YEN 98.16	YEN 196.32
Product 4	9	YEN 81.4	YEN 732.6
Product 5	1	YEN 118.13	YEN 118.13
Product 6	7	YEN 90.5	YEN 633.5
Product 7	6	YEN 189.44	YEN 1136.64
Product 8	4	YEN 158.22	YEN 632.88
Product 9	7	YEN 152.61	YEN 1068.27

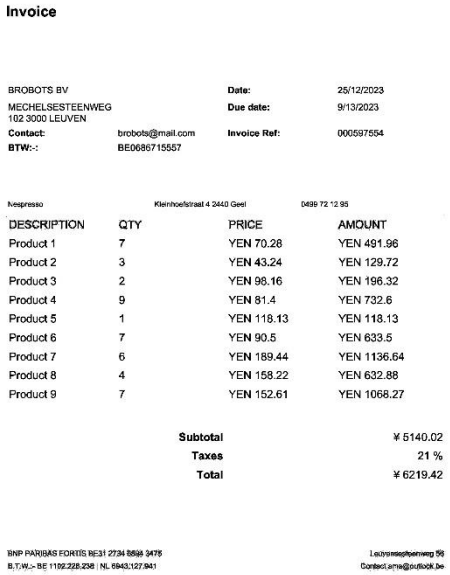
Subtotal	¥ 5140.02
Taxes	21 %
Total	¥ 6219.42

BNP PARIBAS FORTIS BE31 2734 6894 3475
B.T.W.:- BE 1102.228.238 | NL 6943.127.941
Leuvensesteenweg 56
Contact:ama@outlook.be

Figure 5 Grayscale Document

invoices, ensuring accurate extraction of the desired information.

Additionally, I applied morphological changes to remove lines by detecting long adjacent pixels with values higher than 25 pixels for the horizontal lines and values higher than 35 pixels for the vertical lines. This approach effectively eliminated unnecessary lines, which could interfere with the accuracy of the OCR engine, particularly when text appeared below such lines. It also allowed smaller vertical lines, that were part of the letters, to remain untouched.



Invoice

BROBOTS BV
MECHELSESTEENWEG
102 3000 LEUVEN
Contact: brobots@mail.com
BTW:-: BE0686715557

Date: 25/12/2023
Due date: 9/13/2023
Invoice Ref: 000597554

Nespresso Kleinhoefstraat 4 2440 Geel 0499 72 12 95

DESCRIPTION	QTY	PRICE	AMOUNT
Product 1	7	YEN 70.28	YEN 491.96
Product 2	3	YEN 43.24	YEN 129.72
Product 3	2	YEN 98.16	YEN 196.32
Product 4	9	YEN 81.4	YEN 732.6
Product 5	1	YEN 118.13	YEN 118.13
Product 6	7	YEN 90.5	YEN 633.5
Product 7	6	YEN 189.44	YEN 1136.64
Product 8	4	YEN 158.22	YEN 632.88
Product 9	7	YEN 152.61	YEN 1068.27

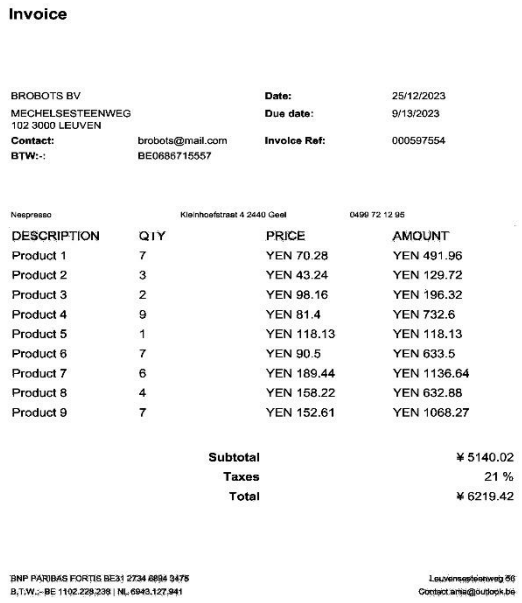
Subtotal	¥ 5140.02
Taxes	21 %
Total	¥ 6219.42

BNP PARIBAS FORTIS BE31 2734 6894 3475
B.T.W.:- BE 1102.228.238 | NL 6943.127.941
Leuvensesteenweg 56
Contact:ama@outlook.be

Figure 6 Document with cleaned up lines

Step 3: OCR (Object Character Recognition)

Subsequently, the pre-processed images were fed into the Tesseract OCR Engine to extract the text content. Tesseract does not only give the text values found within an image but also their location within the document as data, which we will use later to draw bounding boxes on the document later. Google Tesseract is an open-source OCR service meaning its accuracy will be slightly reduced.



Invoice

BROBOTS BV
MECHELSESTEENWEG
102 3000 LEUVEN
Contact: brobots@mail.com
BTW:-: BE0686715557

Date: 25/12/2023
Due date: 9/13/2023
Invoice Ref: 000597554

Nespresso Kleinhoefstraat 4 2440 Geel 0499 72 12 95

DESCRIPTION	QTY	PRICE	AMOUNT
Product 1	7	YEN 70.28	YEN 491.96
Product 2	3	YEN 43.24	YEN 129.72
Product 3	2	YEN 98.16	YEN 196.32
Product 4	9	YEN 81.4	YEN 732.6
Product 5	1	YEN 118.13	YEN 118.13
Product 6	7	YEN 90.5	YEN 633.5
Product 7	6	YEN 189.44	YEN 1136.64
Product 8	4	YEN 158.22	YEN 632.88
Product 9	7	YEN 152.61	YEN 1068.27

Subtotal ¥ 5140.02
Taxes 21 %
Total ¥ 6219.42

BNP PARIBAS FORTIS BE31 2734 6894 3475
B.T.W.:- BE 1102.228.238 | NL 6943.127.941

Leuvensesteenweg 56
Contact ama@outlook.be

Figure 7 Document to be used for OCR

Invoice BROBOTS BV Date : 25/12/2023 MECHELSESTEENWEG Due date : 9/13/2023 102 3000 LEUVEN Contact : brobots@mail.com Invoice Ref : 000597554 BTW:- : BE0686715557 Nespresso Kleinhoefstraat 4 2440 Geel 0499 72 12 95 DESCRIPTION QTY PRICE AMOUNT Product 1 7 YEN 70.28 YEN 491.96 Product 2 3 YEN 43.24 YEN 129.72 Product 3 2 YEN 98.16 YEN 196.32 Product 4 9 YEN 81.4 YEN 732.6 Product 5 1 YEN 118.13 YEN 118.13 Product 6 7 YEN 90.5 YEN 633.5 Product 7 6 YEN 189.44 YEN 1136.64 Product 8 4 YEN 158.22 YEN 632.88 Product 9 7 YEN 152.61 YEN 1068.27 Subtotal ¥ 5140.02 Taxes 21 % Total ¥ 6219.42 BNP PARIBAS FORTIS BE31 2734 6894 3475 Leuivensesteenweg 56 BTW . BE 1102.228,238'| NL6943:127.941 Contact arid @sutlook. be

Figure 8 Extracted text

Tesseract offers different configurations tailored for various document layouts, including single lines of text, tables, and auto-detection. However, after conducting extensive experimentation, it was determined that utilizing the uniform block of text configuration yielded the best overall performance. Although customizing the configuration per document showed slight improvements in specific cases, these gains were marginal and not consistently guaranteed. Therefore, the decision was made to adopt the universal block of text configuration as the default setting, ensuring reliable and efficient performance across a wide range of documents.

The extracted text using the open-source version of Google Tesseract exhibits reduced accuracy, with minor misinterpretations of text located within previously gray areas. These inaccuracies can be attributed to the presence of small optical artifacts resulting from the grayscaling process.

BNP PARIBAS FORTIS BE31 2734 6894 3475 Leuvensesteenweg 56
B.T.W.:- BE 1102.228.238 | NL 6943.127.941 Contact ama@outlook.be

In an attempt to mitigate this issue, erosion was applied; however, it was found to significantly degrade the overall document quality, rendering it an impractical solution.

Step 4: Annotate data

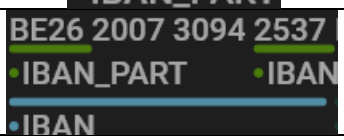
Once I obtained the resulting text files, I proceeded with the annotation process using Doccano, which is an open-source annotation tool which can be used by multiple people for the same project by hosting it using a docker container. Initially, I performed annotations following the standard Named Entity Recognition (NER) scheme to leverage spaCy's NER functionality.

Figure 9 Doccano Annotation GUI

Subsequently, in order to enhance the annotation capabilities, I made further adjustments to the annotations by incorporating both traditional Named Entity Recognition (NER) entities and annotations in the context of pure spans or tokens. These modifications focused on the utilization of each label and the manner in which they were annotated, rather than altering the fundamental types of annotations within Doccano itself.

In NER annotations, it is possible to encompass multiple spans or words, allowing for the annotation of complete addresses or names as Named Entities. However, for the SpanCat component, the objective was to isolate singular spans, such as individual street names, to optimize its functionality. Therefore, the adjustment involved ensuring that the SpanCat annotations exclusively encompassed singular spans during the annotation process, while NER annotations retained their capacity to encompass multiple spans.

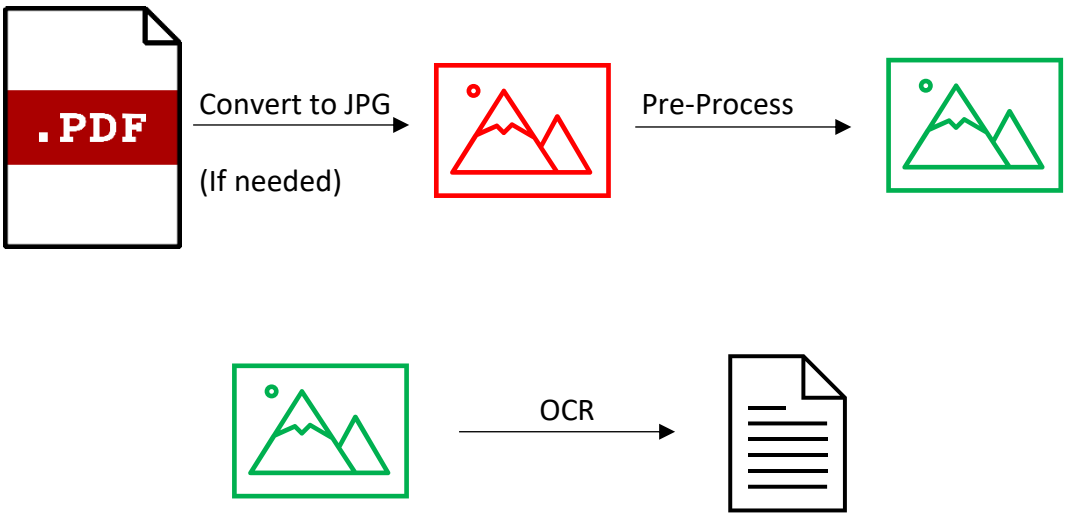
A good example to show the difference between a singular span for the SpanCat component and the normal NER component would be the different ways of annotating an IBAN number.

	Only one part of the IBAN is annotated meaning it's a single span.
	The entirety of the IBAN is annotated here as well (while still allowing for single spans) meaning it will be used for NER.

By incorporating both types of annotations and adjusting their usage accordingly, the expanded annotation capabilities provided a more comprehensive and versatile approach to capturing and categorizing entities within the text data.

To facilitate these enhanced annotations, I utilized spaCy's SpanCat component, which functions similarly to the regular NER component but focuses on singular spans. This approach allowed for the creation of composite annotations, a concept akin to overlapping annotations seen in some paid annotation services. By annotating singular spans within the SpanCat component and later incorporating them into the NER component further down the pipeline, I aimed to improve the overall performance of the NER component. Moreover, this approach enabled the individual detection of specific elements, such as street numbers or parts of an IBAN, even when the complete entity could not be found.

To summarize the process so far; the original documents in PDF or image format undergo conversion to the appropriate JPG format. These processed images are then subjected to pre-processing techniques. The next step involves performing optical character recognition (OCR) using Tesseract, which extracts the desired textual data from the images. This sequential workflow enables the transformation of the original documents into a format that can be efficiently processed and analyzed, ultimately obtaining the required textual information from the OCR output.



The primary objective of the previous steps was to obtain a clean and high-quality dataset as a foundation for developing the NLP model. The quality of the data plays a crucial role in determining the performance of the model, which is especially critical when dealing with

important, and at times financial, information. Having acquired the refined dataset, the subsequent step focuses on outlining the methodology employed to construct the current NLP model. This section will delve into the specific techniques and approaches utilized to leverage the meticulously curated data and build a robust and high-performing model tailored to the task at hand.

Step 5: NLP modelling

Before I could start training the actual model, I had to make one slight adjustment. Since I opted to use the spaCy library I also need to turn my current dataset into '.spacy' files. For this I wrote a small Python script to go through all the annotations and convert them.

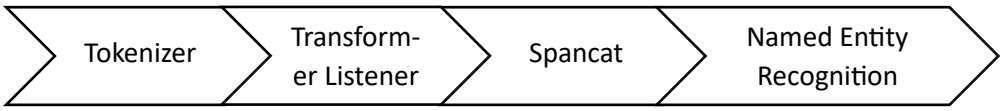
Afterwards, I conducted benchmarking experiments to compare the performance of different models. Specifically, I evaluated both the standard Tok2Vec model and spaCy's transformer model, which is based on the 'bert-base-multilingual-uncased' transformer. Through rigorous testing and analysis, it became evident that the transformer model outperformed the Tok2Vec model. This superiority can be attributed to the transformer's advanced ability to comprehend contextual information within a written document. Despite these documents typically containing relatively limited amounts of text, the transformer model still demonstrated noticeable improvements, suggesting that even subtle contextual cues contribute to enhanced performance.

I built my own spaCy model based on the transformer model spoken about above. A custom spaCy model is constructed by assembling and configuring a pipeline of different components. Each component in the pipeline performs a specific task, which work together sequentially to process the input text and generate the desired output.

By combining these different components into a pipeline, the custom spaCy model enables efficient and accurate processing of text data, providing valuable linguistic insights and facilitating a wide range of natural language processing tasks.

The final configuration of my spaCy pipeline was designed as follows:

The process commenced with the application of a spaCy tokenizer, which segmentized the text into individual tokens. This was followed by the integration of a Transformer Listening Layer, which leveraged the power of transformer models to capture contextual information.



In terms of the pipeline order, the SpanCat component was positioned before the NER component. This deliberate arrangement aimed to utilize the results from the SpanCat component to enhance the accuracy and performance of the NER component. By leveraging the information derived from SpanCat, we could enrich the NER results and improve the identification and classification of named entities within the text.

In the context of training a machine learning model, epochs and steps are two distinct concepts. An epoch refers to a complete pass over the entire training dataset during the training process. In contrast, a step refers to a single update of the model's parameters based on a batch of data.

During the process of benchmarking and experimenting with different configurations, it was observed that utilizing steps instead of epochs offered flexibility and made it easier to work with. This led to the decision of employing a maximum of 20,000 steps, rather than explicitly setting the number of epochs, in the training process. By focusing on steps, it provided finer control over the training iterations and allowed for more nuanced adjustments. Additionally, incorporating an early stopping mechanism set at 1,600 steps ensured that the training could be efficiently halted if desired performance criteria were not met. This approach provided the benefits of improved performance and greater adaptability in fine-tuning the model's training process.

E	#	LOSS TRANS...	LOSS SPANCAT	LOSS NER	SPANS_SC_F	SPANS_SC_P	SPANS_SC_R	ENTS_F	ENTS_P	ENTS_R	SCORE
0	0	14192.95	8945.19	504.33	0.40	0.20	33.47	0.23	0.12	2.11	0.00
6	200	495702.79	289667.33	50053.05	35.26	79.02	22.69	64.92	64.58	65.26	0.50
12	400	1223.15	14969.90	17069.98	70.17	86.63	58.96	74.03	77.91	70.53	0.72
19	600	859.93	7057.19	16798.80	76.95	88.97	67.79	79.10	85.37	73.68	0.78
25	800	901.75	5136.85	15985.61	76.21	88.21	67.09	75.68	77.78	73.68	0.76
32	1000	644.47	4209.06	15176.43	77.95	87.46	70.31	77.09	82.14	72.63	0.78
38	1200	591.83	3881.98	14894.68	77.79	89.29	68.91	81.91	82.80	81.05	0.80
45	1400	718.95	3389.04	13493.03	78.37	88.97	70.03	81.08	83.33	78.95	0.80
51	1600	587.00	2595.31	13511.12	80.22	89.48	72.69	80.00	82.22	77.89	0.80
58	1800	555.43	1535.82	11623.87	80.80	90.31	73.11	81.25	80.41	82.11	0.81
64	2000	775.71	1181.82	11597.21	81.60	90.17	74.51	81.52	84.27	78.95	0.82
70	2200	846.09	1116.18	9554.33	80.80	89.46	73.67	80.21	81.52	78.95	0.81
77	2400	594.49	1003.00	9076.07	81.42	90.57	73.95	79.37	79.79	78.95	0.80
83	2600	663.38	935.58	7611.59	80.37	88.81	73.39	81.05	81.05	81.05	0.81
90	2800	405.30	843.59	6572.29	81.83	91.36	74.09	77.84	80.00	75.79	0.80
96	3000	441.14	805.85	5407.17	81.42	91.00	73.67	78.92	81.11	76.84	0.80
103	3200	349.05	796.89	4216.38	81.46	89.04	75.07	77.66	78.49	76.84	0.80
109	3400	199.89	714.53	3422.41	81.48	90.72	73.95	80.00	82.22	77.89	0.81
116	3600	206.62	720.37	2410.44	81.33	90.55	73.81	81.32	85.06	77.89	0.81

Figure 10 Results from the training process for the NLP model

In the provided image, the training results of our spaCy model are depicted, revealing a best total F score of 0.82. Although this score demonstrates significant progress, it falls short of the desired 0.9+ threshold typically required for production-ready applications. The primary reason for this suboptimal performance can be attributed to the limited amount of training data available for spaCy's internal checks. Normally, a dataset of 272 documents would be utilized, but some of these documents are allocated for validation and testing purposes and are not included in the training process. As a result, the model is trained on only 218 documents, which is considered insufficient for developing production-ready models. However, it is worth noting that the current implementation serves as a Proof of Concept, and further improvements and iterations can be made in the future to enhance the model's performance.

```
✓ Pipeline can be initialized with data
✓ Corpus is loadable

============================== Training stats ==============================
Language: xx
Training pipeline: transformer, spancat, ner
218 training docs
27 evaluation docs
⚠ 1 training examples also in evaluation data
⚠ Low number of examples to train a new pipeline (218)

============================= Vocab & Vectors ==============================
ℹ 53149 total word(s) in the data (7983 unique)
ℹ No word vectors present in the package

========================== Span Categorization ==========================

Spans Key   Labels
---------   ----------------------------
sc          {'TOTALPRICE', 'END_DATE', 'CITY', 'DATE_SENT', 'COMPANYNAME', 'BTWNUM', 'DATE', 'EMAIL', 'INVOICEREF', 'CURRENCY', 'IBAN_PART', 'STREET'}
```

Figure 11 Results of internal spaCy evaluations

After conducting further testing on my model, it has shown respectable F scores across various entity types. However, two specific scores draw attention: the SPANS for ADDRESS and IBAN. This observation aligns with our expectation, as these entities often consist of multiple spans meaning the Span Categorizer wouldn't be able to detect them. Additionally, we noticed slightly lower scores for END_DATE and DATE_SENT, which can be attributed to the fact that these entities provide supplementary information in addition to the date itself. However, the lower Recall score for DATE is somewhat perplexing and may be attributed to the diverse types of dates annotated in the data. Furthermore, it is worth considering that the interpretation and standardization of dates can vary, potentially influencing the model's performance in this regard.

```
================================== Results ==================================

TOK       100.00
NER P     86.25
NER R     79.31
NER F     82.63
SPAN P    91.12
SPAN R    73.60
SPAN F    81.43
SPEED     4756


============================== NER (per type) ==============================

               P       R       F
IBAN       95.65   95.65   95.65
ADDRESS    82.46   73.44   77.69


============================== SPANS (per type) ==============================

                    P       R       F
STREET          97.01   94.20   95.59
INVOICEREF      94.12   88.89   91.43
CURRENCY        94.51   97.73   96.09
BTWNUM          94.74   92.31   93.51
END_DATE        83.33   51.72   63.83
TOTALPRICE      85.71   80.00   82.76
DATE_SENT       76.92   55.56   64.52
ADDRESS          0.00    0.00    0.00
CITY            87.88   87.88   87.88
DATE            85.42   51.90   64.57
IBAN             0.00    0.00    0.00
COMPANYNAME     81.03   90.38   85.45
EMAIL           93.10   96.43   94.74
IBAN_PART       97.78   93.62   95.65
```

Figure 12 Results of testing the NLP model

Considering the limitations in terms of available data and time constraints, it is reasonable to conclude that the current performance of the model represents its optimal state without extensive fine-tuning. While further improvements could potentially be achieved through meticulous adjustments and refinements, such an endeavor would require substantial additional time and resources. Given the existing constraints, the model has already achieved a commendable level of performance, and any further enhancements would likely necessitate a more comprehensive and time-intensive fine-tuning process.

Step 6: GUI

In this subsequent step of the Data Extraction project, a user-friendly Graphical User Interface (GUI) was developed using Streamlit. This GUI simplifies the process for users by providing an intuitive interface where they can effortlessly upload their invoices in PDF, JPG, or PNG format. The system then automatically performs image pre-processing, Optical Character Recognition (OCR), and data extraction on the uploaded invoices. Notably, the GUI supports the processing of multiple invoices simultaneously, creating a dedicated tab for each uploaded invoice. This streamlined approach enhances the user experience and facilitates efficient extraction of data from invoices, making the entire process seamless and convenient.

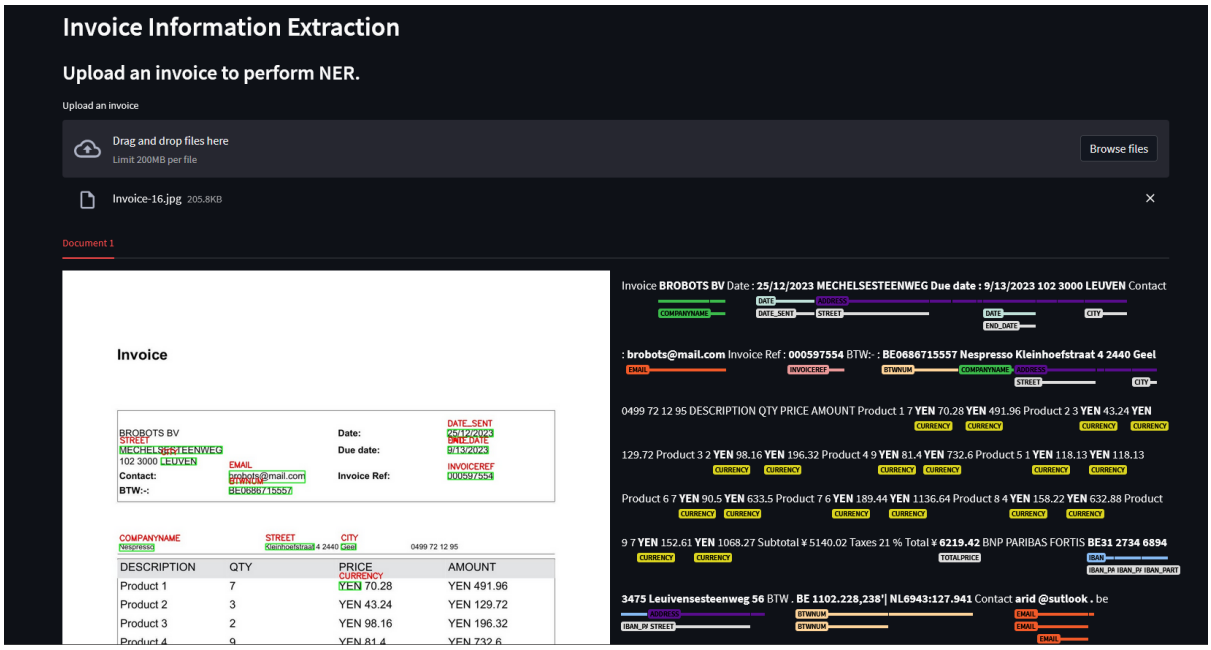


Figure 13 GUI front screen

In addition to the uploaded invoices, the GUI also displays the extracted text alongside the detected information. This presentation format provides users with a comprehensive view of both the Named Entity Recognition (NER) entities and the SPAN entities. To achieve this unified display, all entities are converted to SPAN entities and presented in a consistent manner. By harmonizing the representation of NER and SPAN entities, users can easily interpret and analyse the extracted information, gaining valuable insights from the displayed data.

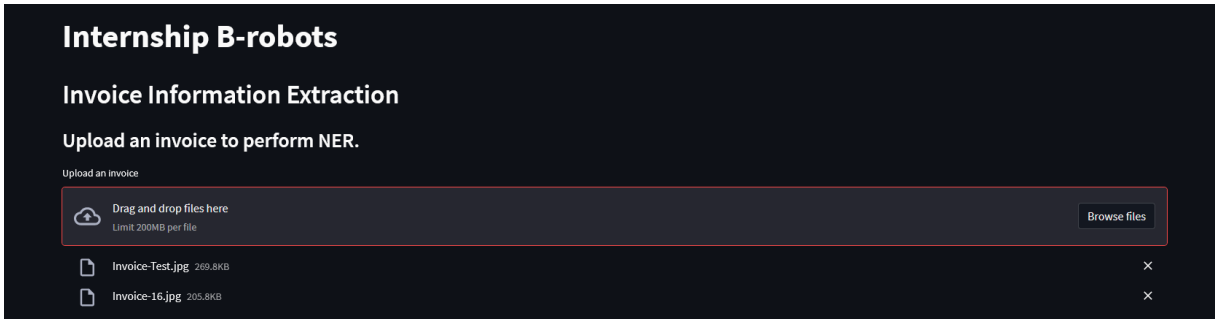


Figure 14 Showing multiple upload files

To enhance user convenience, the GUI supports the simultaneous upload of multiple documents, with a combined maximum total size of 200MB. Users can select and upload multiple documents at once, and the system will create separate tabs for each uploaded document. This feature enables users to efficiently handle and process multiple documents within the same session, streamlining their workflow and improving effectivity.

Additionally, the GUI also offers a customizable component that allows users to tailor the display of important data based on the document type. This feature enables developers to define specific rules or criteria for each document type, ensuring that the most relevant information is prominently displayed and easily accessible. By customizing the display based on document type, users can streamline their data analysis process, focus on the key data points, and extract meaningful insights more efficiently. Currently this is set only set up for invoices due to time constraints and the inherent focus on invoices. The actual ability to change the different key data has not been made available to normal users and must be altered manually at the time of writing.

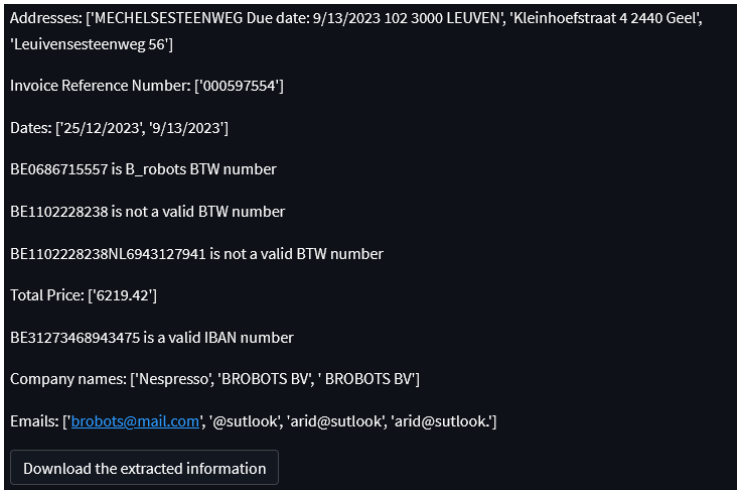


Figure 15 GUI showing customizable import entities

Furthermore, the GUI offers the capability to download the extracted information in the form of a JSON file. This functionality allows users to easily retrieve and store the extracted data for further analysis or integration with other systems. By providing a convenient JSON download option, users can effortlessly access the structured data obtained from the invoice extraction process, enabling seamless data sharing and utilization. This JSON document is however, made within the Doccano data format for easy use between the two applications.

Beyond the base functionality of extracting text out of documents I also provided an Auto Labelling functionality to speed up the internal annotation process of the company for future annotation or NLP project needs.

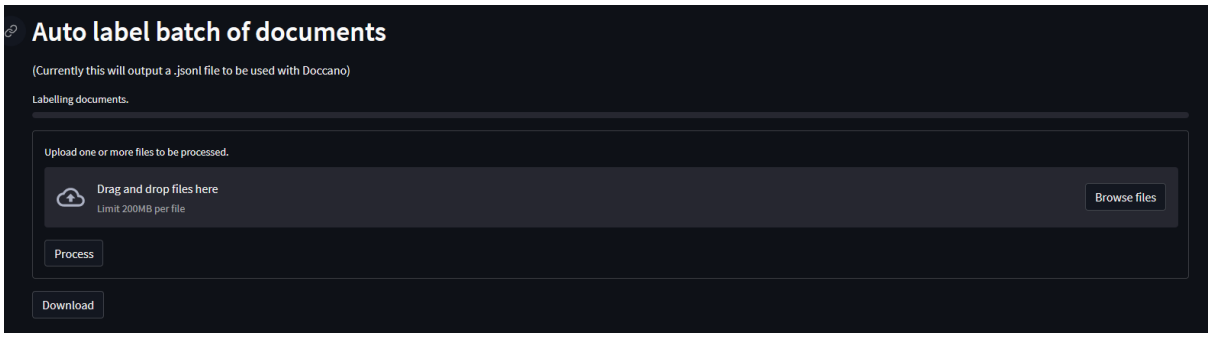


Figure 16 GUI showing auto labeler interface

To accommodate varying annotation needs, the maximum upload size for the application has been set at 200MB. However, for larger annotation jobs, this limit can be adjusted accordingly. To enhance user experience and provide transparency, a loading bar has been implemented, allowing users to easily track the progress of their annotation jobs. This feature ensures that users can conveniently monitor the status of their tasks, providing a more streamlined and user-friendly annotation process.

Throughout this process, every uploaded file undergoes the previously explained steps, including image pre-processing and OCR. These steps ensure that the uploaded documents are properly prepared for further analysis. Once pre-processing and OCR are completed, the text is tokenized, breaking it down into individual units such as word. This tokenization enables efficient processing and analysis of the textual content. Subsequently, data extraction techniques are applied to identify and extract relevant information from the tokenized text.

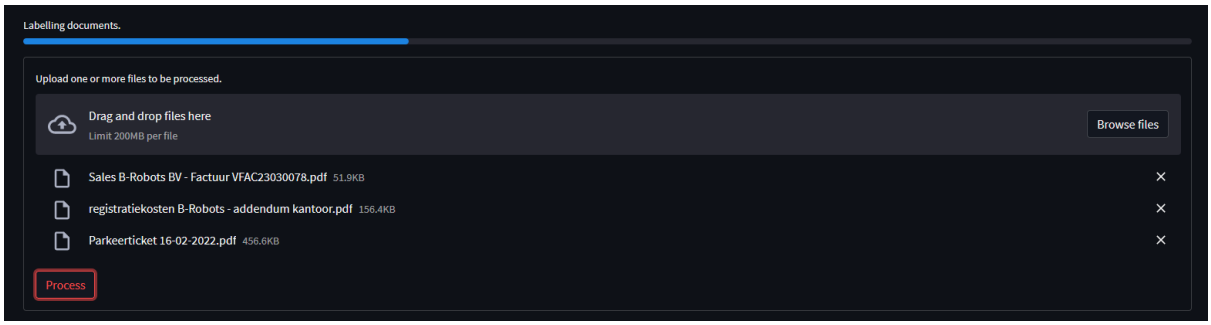


Figure 17 GUI showing auto labeler in proces

Lastly, I incorporated a feature that allows the conversion of annotation files into the '.spacy' file format, facilitating the training or re-training of future NLP models, as long as it uses the spaCy library. By providing this functionality, the GUI empowers users to efficiently leverage their existing annotations for model training, saving valuable time and effort in the development and refinement of future NLP models.

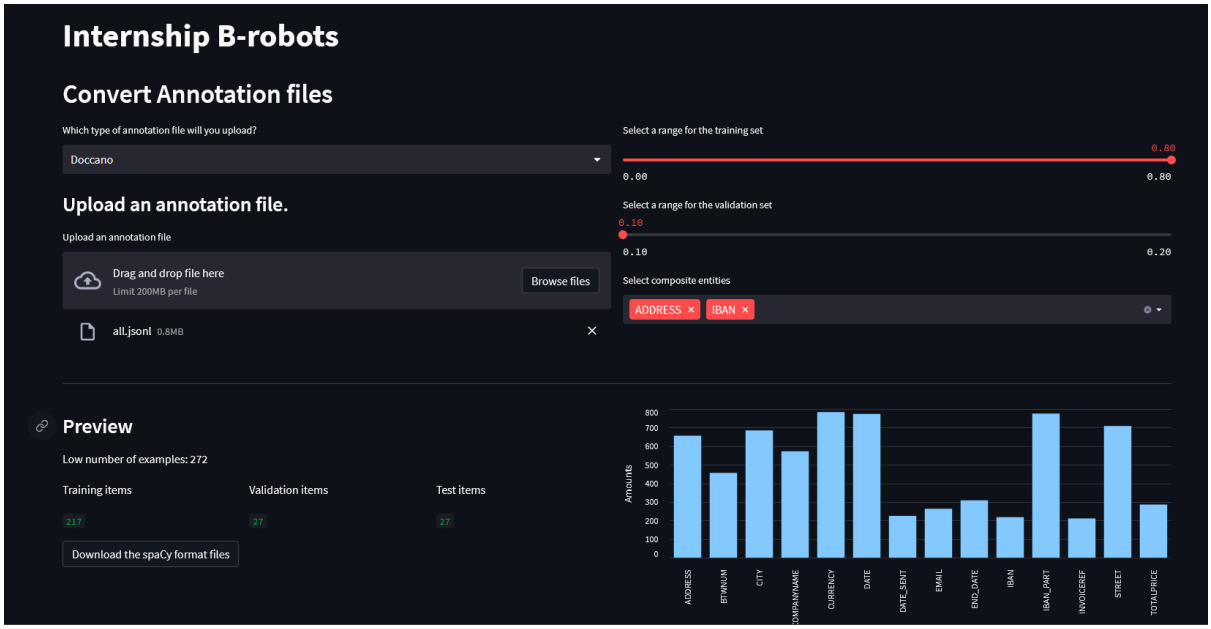


Figure 18 GUI showing conversion tab

3.2 Brightest Testing Framework

The objective is to utilize screenshots to identify and detect various web elements. By analyzing these elements, we can extract their X and Y coordinates on the screen. This information becomes valuable when integrating with the testing framework of Brightest. The extracted coordinates are then employed to automate the process of clicking on specific elements during the testing phase. This approach significantly accelerates the testing of diverse web applications, as it eliminates the need for manual interactions.

Step 1: Labelling

The annotation process was divided between me and the other intern, and we utilized the Doccano annotation tool for this task. Although Doccano does not support direct annotation in the YOLO format, this obstacle was overcome by developing a Python conversion script. This script efficiently converted the annotations from Doccano into the required YOLO format, enabling seamless integration with the subsequent stages of the project.

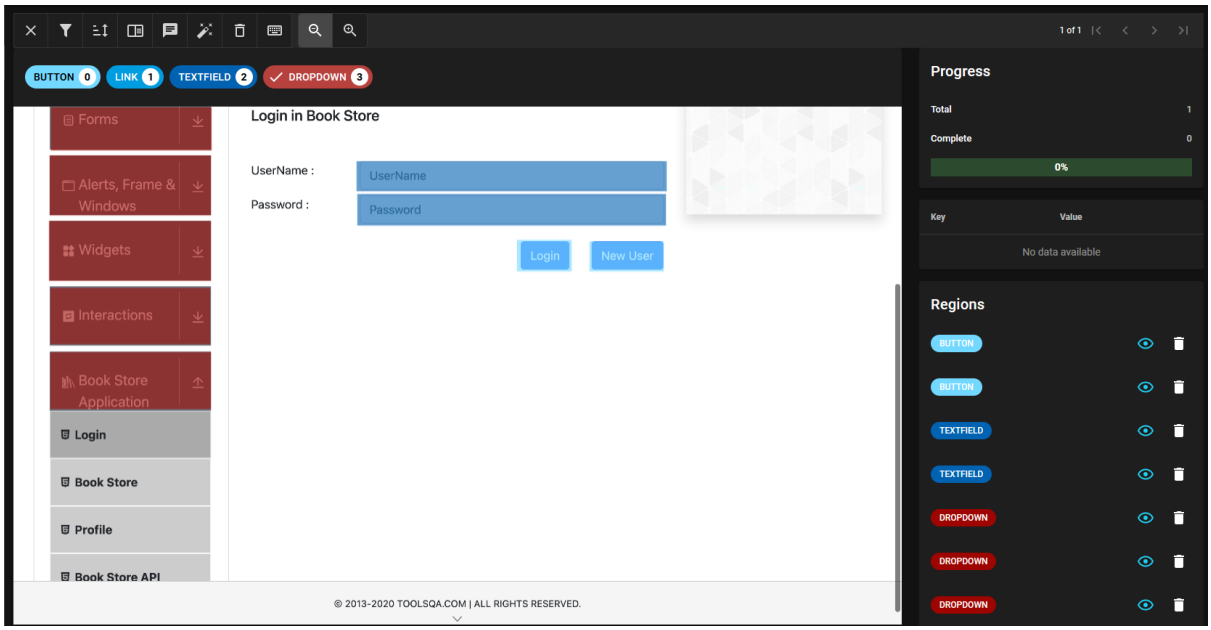


Figure 19 Doccano Object Detection labelling

As depicted in the image above, the labeling process for Computer Vision tasks differs slightly from the labeling process for NLP tasks. However, the underlying principle remains consistent. Accuracy remains a crucial in both cases, however we need to pay slightly more mind to this during this labelling process than for the NLP model. When labeling for Computer Vision, it is important to ensure that the labeled areas precisely encompass the target elements. If areas around the elements are also labeled, the model will learn to recognize and include those areas during inference. In the context of text labeling, the process involves delineating the boundaries around words, whereas in Computer Vision, additional attention is required to ensure that the labeled edges do not extend excessively beyond the target elements.

Step 2: Select and train model

I started with benchmarking various Computer Vision models, including YOLOS, an experimental Visual Transformer, YOLOv5, and YOLOv8. While each model presented its own merits, I ultimately opted to proceed with YOLOv8 due to its higher degree of customization and the ability to adjust input parameters. This flexibility enabled the handling of larger screenshots, which was beneficial for our specific requirements.

The models previously mentioned serve two main purposes: object detection and localization. Object detection involves identifying and classifying various objects present in an image, while localization focuses on determining the precise location of these objects by creating bounding boxes around them. By utilizing bounding boxes, we can extract the coordinates of the outer edges, enabling us to calculate the center coordinates of the objects. This information is crucial for interacting with specific elements accurately.

Furthermore, the object detection aspect not only enables the detection of clickable elements but also provides the ability to classify each element. This classification feature allows for further customization and specific handling of different types of elements based on their identified class.

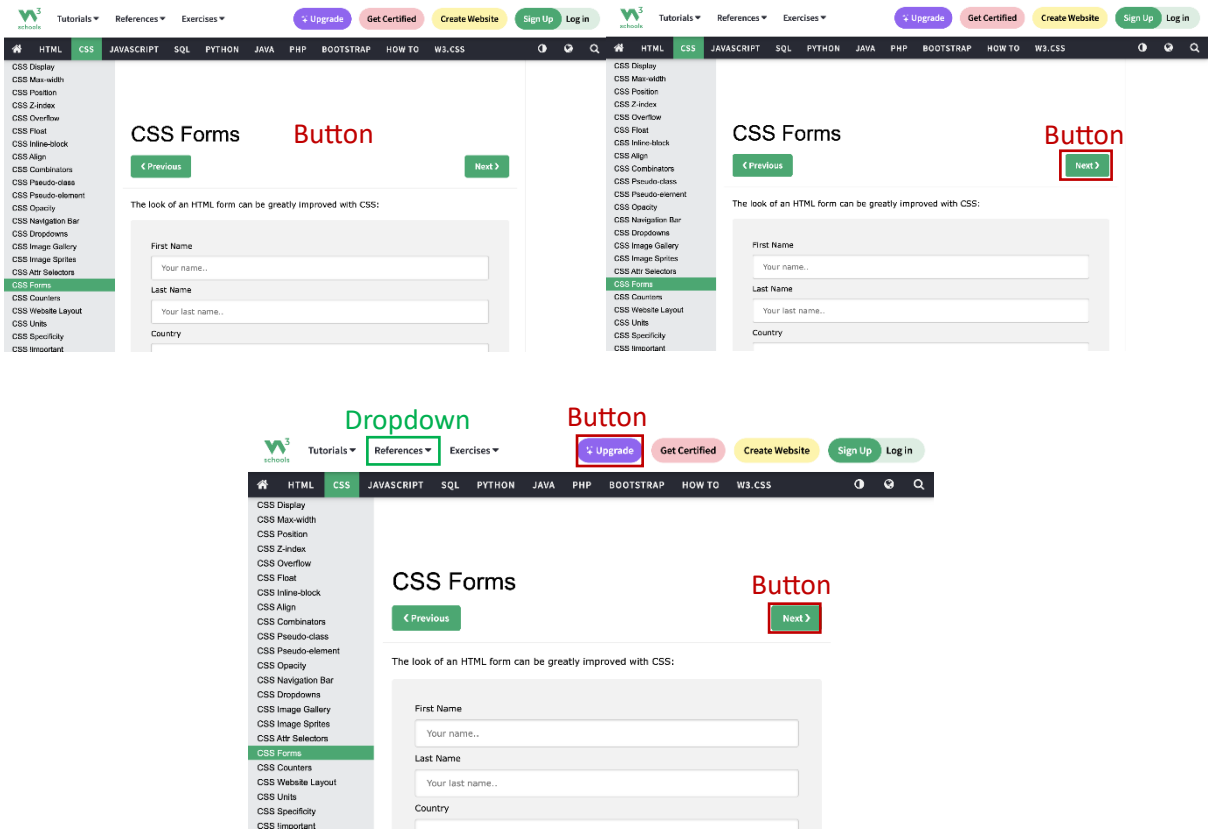


Figure 20 Differences between types of CV

To conduct benchmarking for the YOLOv8 model, I had to convert the dataset from the original Doccano annotation format to the YOLO format. This conversion process was relatively straightforward thanks to the use of Python scripts. Although there were certain aspects and details in each format that required careful attention and manual handling, they did not pose significant obstacles. Overall, the conversion was a manageable task, and I successfully prepared the dataset in the desired YOLO format for benchmarking the YOLOv8 model.

Afterwards, I proceeded to train the model for 100 epochs to assess its performance. Unfortunately, the results were rather underwhelming, as indicated by our mean Average Precision at 50% IoU threshold (mAP50 score).

```
Model summary (fused): 168 layers, 3006428 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:01<00:00,  1.04s/it]
                   all         22        477      0.705      0.491      0.524       0.26
                button         22        122      0.546      0.434      0.454      0.283
             textfield         22         69      0.807      0.391      0.495      0.137
                  link         22        236      0.826      0.818      0.814       0.41
              dropdown         22         50      0.642       0.32      0.335      0.208
Speed: 0.4ms preprocess, 1.2ms inference, 0.0ms loss, 3.0ms postprocess per image
```

Figure 21 Results training process of CV model

The mean Average Precision at 50% Intersection over Union is a commonly used evaluation metric in object detection tasks. It measures the accuracy of the model in detecting objects by comparing the predicted bounding boxes with the ground truth bounding boxes.

The Intersection over Union (IoU) is a measure of overlap between two bounding boxes. If the IoU between a predicted bounding box and the ground truth bounding box is above a certain threshold, in this case 50%, it is considered a correct detection.

The Average Precision (AP) is calculated by computing the precision-recall curve based on different confidence thresholds for the predicted bounding boxes. It represents the average precision across all recall levels.

The mAP50 score specifically focuses on the average precision at the 50% IoU threshold, which is a common threshold used in many object detection tasks. It provides an indication of the model's ability to accurately detect objects with a moderate level of overlap with the ground truth.

In the context of the mentioned results, having a lower mAP50 score suggests that the model’s performance in accurately detecting objects, means our results aren’t exactly desirable. However, it is important to consider the limitations of the dataset size and the diversity of the data, as these factors can impact the model's overall performance.

Despite the lower confidence in its predictions, the model still demonstrates a reasonably good performance in detecting various objects. It successfully identifies the presence of objects in the input data but exhibits a lack of certainty in the accuracy of its predictions. This implies that while the model can identify objects, there is room for improvement in terms of increasing its confidence and reducing false positives or false negatives.

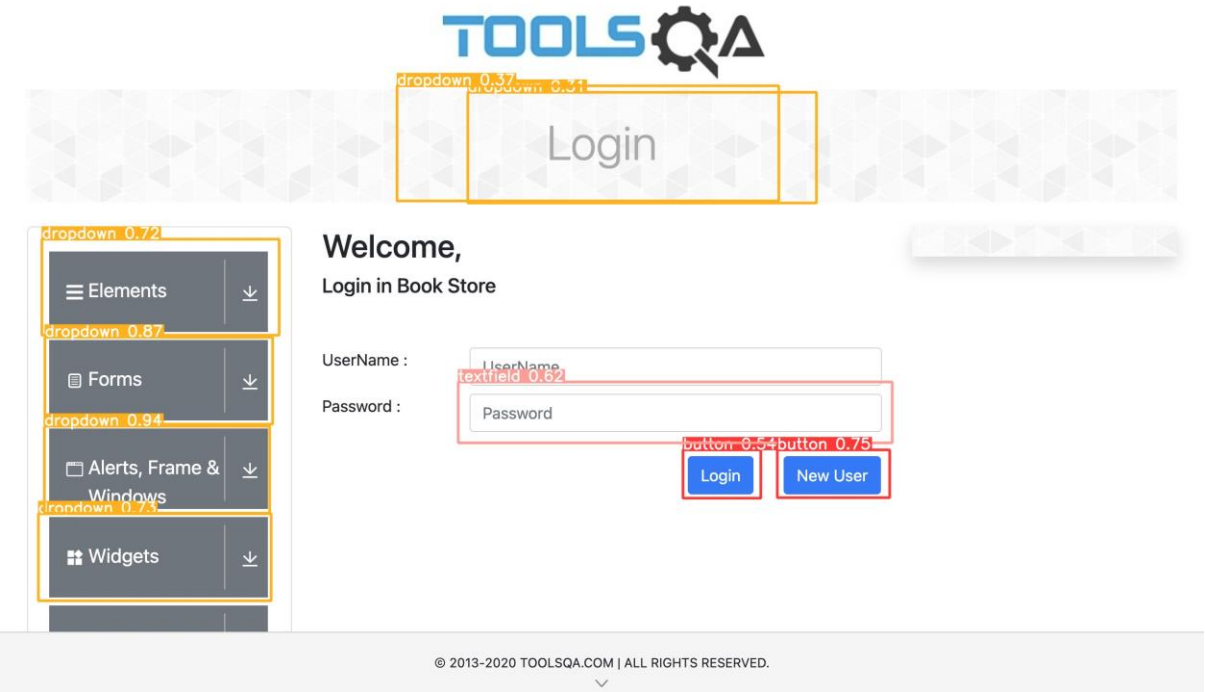


Figure 21 Results of predictions of CV model

Step 3: Optimizing the model

After completing the benchmarking phase, I delved into investigating the factors contributing to lower performance. Three main issues surfaced during my research.

Issue 1:

Firstly, some of the training data proved to be too large for the model, resulting in information loss due to excessive downscaling. Addressing this challenge necessitated exploring strategies to preserve critical details while accommodating the model's limitations.

This issue was solved relatively quickly by simply making sure that future data (screenshots) would be maximum of 1280 by 1280. We chose to keep the original bigger training data as well because of time constraints and not having the will the cut them up and re-annotate them. I gave the option to try and cut them using Python scripts but this wasn't deemed necessary.

Issue 2:

Secondly, we encountered an imbalance in the distribution of labels within the training data. While certain labels had sufficient representation with an adequate number of training instances, others suffered from limited availability. This imbalance introduced challenges in accurately detecting and classifying specific web elements, prompting the need to explore techniques for augmenting the training dataset and achieving a more balanced representation across all labels.

This problem was partially solved by doing some data exploration, looking into which labels we have and how many we have of each. In the end we only chose the labels of which we had at least 200, which still isn't that much but at least it was a start.

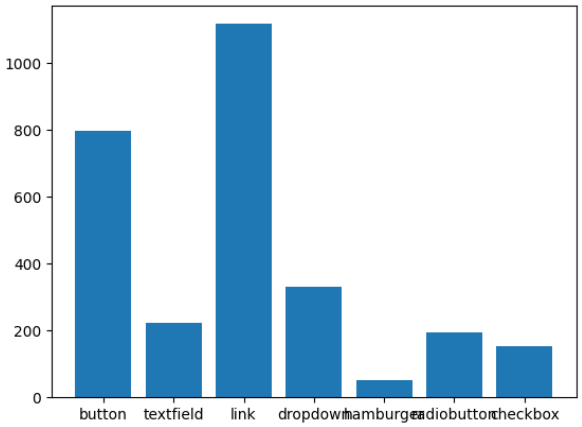


Figure 21 Labels before pruning

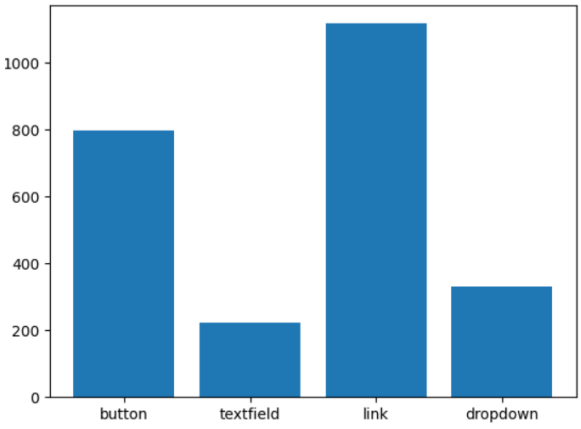


Figure 22 Labels after pruning

Issue 3:

Lastly, the quantity and diversity of the available data presented additional complexities. The dataset itself exhibited significant variations, which posed challenges for the model's generalization capabilities.

The limited amount of available data also posed a significant challenge in achieving optimal performance for the model. However, rather than addressing this issue directly, we approached it as a proof of concept.

We recognized that with a larger and more diverse dataset, the model's performance would likely improve significantly. Therefore, the focus was on showcasing the potential of the model given the existing data, while acknowledging the need for additional data to unlock its full capabilities.

After having fixed some of the issues with the data I decided to re-train the model with the new dataset.

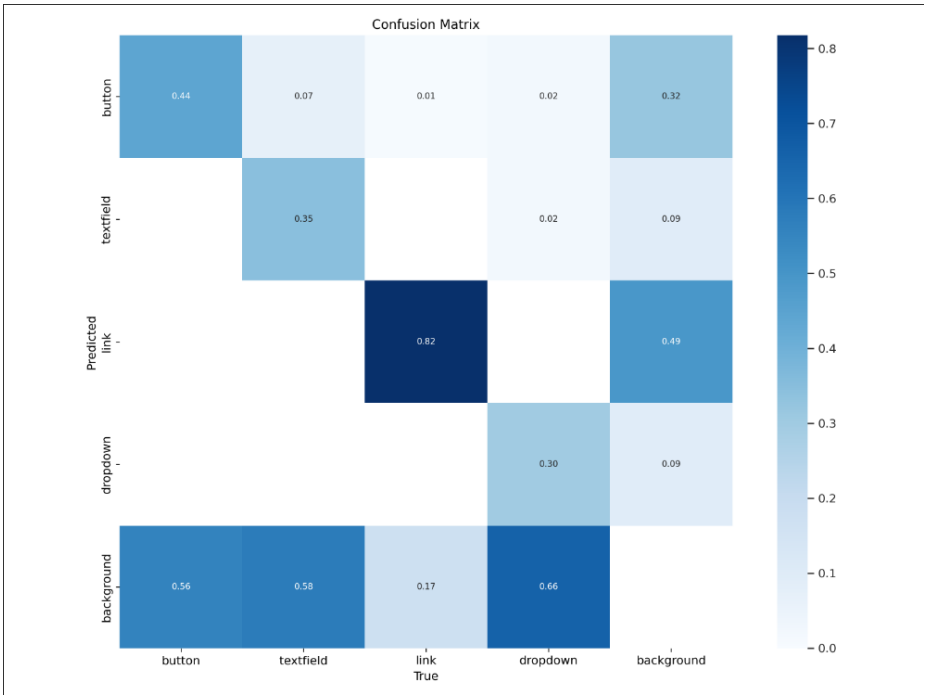


Figure 23 Confusion Matrix of CV model

It's certainly not perfect but with the small dataset that we have we decided to stop it here. In figure 8 we can see that the model is better than the confusion matrix shows us. This could be that the validation set wasn't perfect, which is once again possible due to the small size.

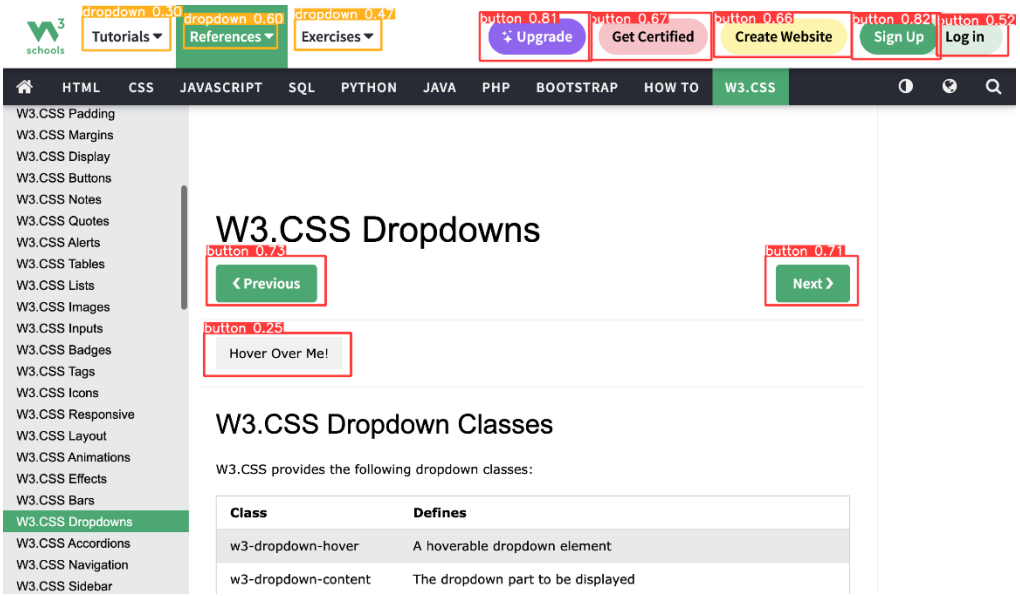


Figure 24 Finalized predictions of CV model

Conclusion

In conclusion, my internship experience at B-Robots has been an enriching and rewarding journey, during which I worked on two primary tasks: data extraction from invoices and collaborating on a computer vision project for a testing framework.

In the realm of data extraction from invoices, I successfully developed a robust pipeline using Python, leveraging the PDF converter and Tesseract OCR Engine to convert invoice images into text. The annotated dataset, prepared using the Doccano annotation tool, facilitated the training of NLP models, specifically spaCy's TextCat, NER and SpanCat components. This resulted in accurate extraction and recognition of crucial information from invoices, such as company names, addresses, invoice amounts, IBAN numbers and BTW numbers. The integration of advanced techniques like thresholding, grayscale conversion, and line removal further enhanced the OCR accuracy and streamlined the data extraction process.

Additionally, my collaboration with another intern from Brightest on the computer vision project for the testing framework proved to be a valuable endeavor. We explored various CV models, benchmarking their performance to identify the most suitable approach. Ultimately, YOLOv8 emerged as the preferred choice due to its customization capabilities and compatibility with larger screenshots. Overcoming challenges related to data size, label distribution imbalance, and data diversity, we successfully fine-tuned the model to detect and extract web elements accurately. The integration of the YOLOv8 model within the Brightest Framework will empower developers to perform robust testing and ensure the quality and integrity of web applications.