Efficient Invoice Interpretation: Practical and AI-Powered MicroService for Automated Data Extraction

HASSAN HUSSIN AND MARTIN LIND
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Bachelor's Thesis

Efficient Invoice Interpretation: Practical and AI-Powered MicroService for Automated Data Extraction

By

Hassan Hussin and Martin Lind

Department of Electrical and Information Technology

Faculty of Engineering, LTH, Lund University

SE-221 00 Lund, Sweden

LTH School of Engineering

Lund University

Box 882

SE-251 08 Helsingborg

Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg

Lunds universitet

Box 882

251 08 Helsingborg

Printed in Sweden

Lunds universitet

Lund 2024

Abstract

This thesis details the development, implementation, and optimization of a local microservice designed to enhance invoice processing at Hulo, a company that manages automatic payment invoices. This is conducted through Artificial Intelligence (AI) and Machine Learning (ML) techniques, focusing particularly on Computer Vision and Object-detection. Motivated by the need to address the inefficiencies of traditional manual processing which is error-prone, slow, and costly, this project employs a structured methodological approach. Initial data gathering, meticulous data labeling, and extensive training and validation of an AI model using tools like Roboflow [1] are key components. The chosen You Only Look Once (YOLO) model [2], known for its robust object detection capabilities, demonstrates significant improvements in recognizing and classifying invoice data through successive training phases.

Challenges such as class imbalances are tackled through dataset enrichment and augmentation techniques, enhancing the model's robustness and generalization capability across diverse invoice formats. The system effectively transforms invoices into structured JSON data, thus automating and streamlining business practices.

The thesis encapsulates the project's journey from conceptualization to deployment, highlighting strategic solutions for data quality enhancement and computational efficiency. The challenges and opportunities of the transition of this microservice into a production environment, where it undergoes rigorous real-world testing, ensures robustness, data security, and operational reliability. The successful implementation of this project underscores AI's role in improving business processes and aligning with industry trends toward digital automation and intelligent systems.

Keywords

Artificial intelligence (AI), Automation, Computer vision, Invoice extraction, Machine learning (ML), Object-detection, YOLO

Acknowledgments

We would first like to express our profound gratitude to HULO for their invaluable support and the opportunity to pursue this thesis. We are also immensely thankful to Sara Ramezanian for her supervision and guidance in structuring this report. Additionally, we appreciate Marcus Klang for his assistance in navigating our search for AI tools. Lastly, we wish to thank Christian Nyberg for his role as our examiner.

Hassan Hussin and Martin Lind

May 2024, Helsingborg

Contents

List of Figures

List of Tables

Preface

This thesis represents a collaborative effort between Hassan and Martin, focusing on object detection models with an equitable division of labor. The project began with both collaborators jointly defining the goals and research questions, which established the direction of the study.

Hassan is responsible for writing the introduction and addressing research questions three through five in the analysis and conclusion sections. Martin handles research questions one and two, and he also focused on the future work section, proposing potential extensions and applications of the research.

The technical background, methodology, and results sections are collaborative efforts. The division of labor is clearly defined by each collaborator's part in the implementation: Hassan focuses on training the object detection model, while Martin concentrates on data extraction. This cooperative approach not only enhances the depth of the research but also ensures that each section is managed by the collaborator most familiar with that aspect of the project. For a detailed overview of the division of responsibilities, see Table 1.

Table 1: Division of labor in thesis

Section	**Responsibility**
Introduction	Hassan
Goal formulation and research questions	Joint effort
Research question 1-2	Martin
Research question 3-5	Hassan
Technical Background	Joint effort
Method	Joint effort
Results	Joint effort
Future work	Martin
Implementation part 1 (Training of object detection model)	Hassan
Implementation part 2 (Data extraction)	Martin

Chapter 1

Introduction

This chapter outlines the subject and scope of the bachelor thesis. It provides a comprehensive background to the reader and engaging them in a discussion of the problem that paves the way to the main purpose and research questions of the study. Additionally, it covers the objectives of the study and its limitations. Information regarding the target and the structure of the bachelor thesis is also included.

1.1 Background

In the evolving landscape of digital transformation, businesses are increasingly turning to technological innovations to streamline operations and enhance efficiency. Among these operational challenges, invoice processing stands out as an area ripe for optimization. Traditionally reliant on manual labor, the process of managing invoices is often inconvenient, error-prone, and time-consuming. The introduction of Artificial Intelligence (AI) technologies has initiated a significant shift towards greater efficiency and accuracy in invoice processing [3]. These solutions is instrumental in reducing manual labor and operational costs.

The deployment of Intelligent Document Processing and AI technologies like Computer Vision, Machine Learning (ML), and Text Classification facilitates the handling of unstructured data, enabling the automation of more complex processes. Building upon this foundation, recent research conducted by Hedberg [4] has delved into the application of ML for automated decision-support in the context of invoice processing. This approach

aims to provide accountants with suggestions on appropriate accounts and cost centers for invoices, thereby streamlining the financial operations within organizations. The findings of the research conducted by Hedberg suggests that decision-support systems powered by ML offer significant benefits, including time savings, reduced mental effort, more coherent book keeping, error detection, and enhanced levels of automation [4]. These systems are perceived as a valuable addition to the arsenal of tools available for optimizing invoice processing [5].

However, the research conducted by Hutter et al. [6] also highlights potential challenges associated with implementing machine learning automation. Differences in how different organizations use accounts and cost centers coupled with the complexity of processing some invoices, may lead to uneven performance. Despite these challenges, experiments conducted within the study demonstrated the potential of machine learning to accurately suggest appropriate accounts and cost centers, with accuracy rates ranging from 73–76% for accounts and 50–62% for cost centers. Additionally, a method for filtering machine learning output was developed, aiming to improve the accuracy of automated suggestions, with some filtered suggestions achieving up to 100% accuracy [4].

The project outlined in this thesis seeks to contribute to this field by developing a local microservice for reading invoices. By exploring the potential of AI and machine learning through training of models from scratch this project addresses the need for an automated, efficient, and reliable invoice processing system. Hulo, the company at the center of this project, handles and leverages automatic payment invoices to other companies. Hulo stands to benefit from a solution that enhances productivity, reduces dependency on external services, and ensures data security and operational autonomy. By leveraging the insights and methodologies outlined in the aforementioned research, this project aims to deliver a microservice capable of transforming Hulo's invoice management processes. In doing so, the project aligns with the broader industry trend towards digital automation and the specific challenges and opportunities presented by machine learning in invoice processing.

The thesis project is undertaken in partnership with Hulo IT AB, a Swedish firm with

approximately 20 employees. Based in Malmö, this enterprise specializes in providing direct debit invoicing services to businesses globally and developing innovative digital solutions for their needs.

1.2 Purpose

The main goal of this project is to develop a microservice that can read and interpret invoices accurately and quickly. This project aims to reduce the need for outside services, giving Hulo full control over the service. This control is important for avoiding issues with internet connectivity and data security, ensuring a smooth and dependable process for reading invoices that meets Hulo's needs for independence and local use.

The purpose is to give Hulo a reliable and efficient service for processing invoices in-house. Expected benefits include a big increase in productivity, less risk of disruption from external problems, and better control and oversight of how invoices are managed. Currently, Hulo mainly uses manual methods to pull information from invoices, which takes a lot of time, effort, and is prone to mistakes. Moving to an automated system aims to make this process faster and more accurate, and make it easier to use invoice data in the company's financial systems.

With this commitment in mind, the project aims to include automation technology to improve the current state of invoice management by Hulo. Shifting from manual, error-prone methods to a more accurate and efficient approach will not only elevate the company's overall operations but also improve data security and privacy. Consequently, this will lead to stricter control and enhanced protection of Hulo's financial records.

1.3 Goal formulation

The project aims to develop an AI model tailored for invoice interpretation, targeting a minimum accuracy of 90% in data extraction. Achieving this accuracy is fundamental for providing Hulo with a reliable invoice reading service. Alongside this, the initiative seeks to develop a local microservice capable of processing invoices in just 3 seconds on

average. The emphasis on speed ensures the automation process not only meets the demand for swift response times but also supports Hulo's goal of maintaining operational autonomy with a system that is both efficient and secure. This rapid processing is not just a requirement but a strategic advantage, allowing Hulo to enhance productivity and data privacy simultaneously.

Another crucial objective of the project is to ensure support for both Swedish and English, with a minimum accuracy rate of 90% in correctly interpreting invoices in these languages. This language support is vital to accommodate the linguistic diversity of the invoices received by Hulo. Moreover, the project prioritizes the capability for model training from scratch. This involves the development and implementation of a training module that allows the AI model to learn directly from local data, providing flexibility to either utilize pre-trained models or develop a custom model tailored to Hulo's specific needs.

Lastly, a fundamental goal is to eliminate any dependence on external services, aiming for 100% autonomy in performing invoice reading tasks locally. This ensures that Hulo maintains complete control over the invoice reading service. Each of these objectives is carefully measured and directly aligned with the overarching purpose of enhancing Hulo's invoice management processes through the application of advanced technology.

1.4 Problem formulation

These questions will be answered throughout the thesis project and aim to investigate and improve the manual invoice reading process at the company Hulo. By analyzing the current situation and identifying challenges and opportunities, it is possible to develop a more efficient and secure method for invoice reading. Research questions (RQ) to be addressed in the work include:

- **RQ1**: In what way can the invoice reading service be implemented to handle different document types and easily expand with new models?
- **RQ2**: What technical and language aspects need to be considered to ensure accurate interpretation of invoices in Swedish and English?

- **RQ3**: How can the performance and speed of the developed microservice be improved to meet the requirements for fast and efficient invoice reading?
- **RQ4**: What methods and criteria should be used to train and evaluate the AI model used for invoice reading?
- **RQ5**: What opportunities and challenges may arise when transferring the developed service to a production environment?

1.5 Motivation for the thesis

The selection of the thesis project was driven by its focus on AI, a field considered crucial and highly relevant in today's industries [7]. Given the escalating demand for AI technologies across diverse sectors, acquiring expertise and comprehension in this area is paramount. The anticipation of future engagement in the industry underscores the thesis project as a pivotal opportunity to amass specific knowledge and practical experience in AI. This effort will provide a competitive advantage and prepare for future technological challenges.

Engagement with AI within the thesis project presents a chance to delve into and discern how technology can be tailored and deployed to satisfy emerging needs. This exploration is anticipated to foster innovative solutions and contribute to the advancement of technology in this domain. In essence, the thesis project was chosen for its significant alignment with and timeliness in the field of AI, promising a blend of practical skills and theoretical insight. The company perceives this as an investment in future technological progress, yielding benefits both internally and for the wider community.

1.6 Limitations

To optimize the AI model for efficiency and precision, its document reading capability has been deliberately confined to Swedish and English languages. This constraint is designed to cultivate a more specialized and focused AI service.

The initial step involves a definitive selection and refinement of the specific information

to be extracted from invoices. The focus will initially be on essential details such as the list of items or services provided, the issuer's name and contact details, the payment due date, and critical payment information including account and reference numbers, in addition to significant contact details like sender name, email addresses, and postal addresses. Establishing these foundational details is crucial for developing a robust approach to information management. Progressively, with the objective of enhancing the utility of the information gathered, there is an intention to incrementally broaden the system's capabilities. This expansion will include integrating additional types of data for extraction, such as payment dates, tax amounts, and subtotals, thereby gradually advancing the data collection and analysis methodologies.

1.7 Disposition

The thesis is structured into six detailed chapters that systematically build upon each other. Chapter 2 sets the technical foundation, covering essential concepts and tools ranging from Python programming to advanced machine learning and computer vision techniques. Chapter 3 outlines the methodological approach, detailing each phase of the research from data collection to testing the models. Chapter 4 reports the results from the training and tuning of the models. In Chapter 5, the analysis delves deeper into the challenges and solutions related to the model's development and its readiness for real-world application. The concluding chapter, Chapter 6, summarizes the findings and discusses their implications for future research. This structure allows for a clear and logical progression of research, ensuring that each chapter builds on the knowledge established in the previous ones.

Chapter 2

Technical Background

This chapter embarks on a comprehensive exploration of the foundational technologies underpinning AI, with a focus on machine learning, computer vision, and Python. It delves into how machine learning algorithms harness data to predict outcomes, the ways in which computer vision seeks to emulate human visual perception, and Python's pivotal role as a programming language in AI development. This foundation sets the stage for understanding advanced AI applications, from the construction and execution of AI models to the practical applications of OpenCV and the intricacies of Optical Character Recognition (OCR) technologies.

2.1 Data

The significance of possessing both a substantial quantity and diversity of data in AI development is paramount and cannot be overstated [8]. The process of pre-processing data, which includes collecting, cleaning, and annotating, constitutes more than 80% of the workload in AI development [9]. Therefore, it is crucial to allocate resources where they yield the highest returns. A diverse dataset enhances the model's ability to accurately detect varying data inputs. Additionally, the volume of data is equally critical. As noted by Imran et al. [10], training a model reveals a clear inflection point between 150 to 500 images per class. It is at this range where the initial rapid improvement in performance begins to level off.

2.2 Python

Python is an open-source, high-level programming language created in the 1990s. It is designed to be highly readable and user-friendly. Compared to other languages, Python allows for achieving desired software behavior with less code [11]. It also supports different ways of programming, including object-oriented, functional, and imperative programming. Additionally, Python boasts a large community on platforms like Stack Overflow, making it easy to obtain support for the language [11].

2.3 Machine learning

Machine learning is a part of artificial intelligence where we are trying to make computers learn by studying different kinds of data and statistics. By analyzing incoming data and comparing it with former data, the computer is trying to find patterns and predict an outcome.

2.3.1 Machine learning systems

Machine learning systems contain algorithms for the purpose of training AI models. Selecting algorithms for a certain system can be challenging. If an algorithm is chosen wrongfully, it may necessitate rebuilding the entire system, wasting precious resources. Moreover, the algorithm that is required may vary depending on the data or the desired output.

2.3.2 Machine learning process

There are several steps for a system to be created. These steps are shown in Figure 2.1.

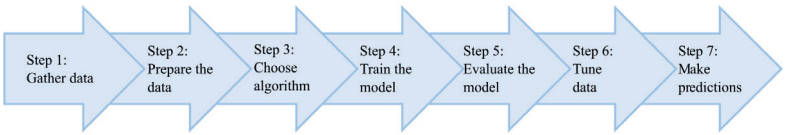

Figure 2.1: Steps for machine learning process

The process of preparing the data includes labeling, which is where we tell the computer what the data actually is. For example, if a picture is presented containing an apple we could label the apple in the image and tell the computer "this is an apple". This is also where the data is partitioned into data for training and data for testing.

In the training of the model, the chosen algorithm is being used together with the "training" data. A model can be trained in different ways, one way is having supervised training for the model. Supervised training is when the input and output are given.

2.3.3 Linear regression

Linear regression is a statistical method used to predict future outcomes by identifying a linear relationship between two variables: an independent variable and a dependent variable. The dependent variable, which is the focus of prediction, relies on the independent variable. However, the independent variable provides the basis for the prediction of the dependent variable's value. This technique is integral to predictive analysis within the fields of data science and machine learning, serving as a foundation to make predictions based on past data, offering a basic but powerful tool for predictive analysis [12].

2.4 AI model training

In machine learning, an epoch denotes a complete pass through the entire training dataset during which the model adjusts its parameters to minimize discrepancies between its predictions and the actual data. This process is fundamental as it underpins how a model learns from data over time [13].

Imagine a machine learning model trained to classify different types of fruits based on images. The training dataset includes numerous images labeled with specific fruit types such as apples, oranges, and bananas. During each epoch, the model reviews every image in this dataset, tweaking its parameters to better predict the correct fruit type for each image. After processing all images once, the model completes one epoch and initiates the next one [14].

Epochs are crucial in machine learning because they incrementally improve the model's ability to classify or predict accurately [15]. As the model cycles through more epochs, it refines its understanding of the features that distinguish different categories within the dataset. This continuous parameter adjustment helps the model make more accurate predictions on new, unseen data.

However, it's essential to balance the number of epochs to avoid overfitting—where a model learns the training data too well, including its noise and errors, which harms its performance on new data [16]. An optimal number of epochs ensures that the model learns enough to make accurate predictions while retaining the flexibility to generalize from new data. Too many epochs can make the model overly specialized to the training dataset, reducing its effectiveness on broader applications [17].

2.5 Computer vision

Computer vision seeks to replicate human visual perception through computational means. By leveraging AI technologies, computers can process and interpret visual data from various sources, including images and videos. This process involves scanning visual inputs and employing pattern recognition technologies, enabling computers to detect and identify objects within these images [18].

Within the realm of computer vision, classification models represent a specific category of machine learning algorithms. These models are tasked with assigning new input data to predefined categories or labels. The complexity of this task varies, if the model is distinguishing between two categories, it is engaged in binary classification. On the other hand, when the classification involves more than two categories, it is dealing with a multi-class classification scenario [19].

2.5.1 OpenCV

OpenCV [1] or Open Computer Vision is an open-source library for computer vision. The project for the library was started by Intel and the Alpha was released in January 1999. OpenCV is used to transform and manipulate videos and images. There are different types of transformations in OpenCV some examples that are used for the purpose of the thesis are:

- turning images to grayscale.
- turning images to black and white.
- crop parts of an image.
- Additional function in OpenCV is to show the image [20].

2.5.2 OCR

The idea of Optical Character Recognition (OCR) started in 1929 but could not be realized until the computer came during the 1950s. OCR is used to recognize, read, or extract data from images to be able to manipulate the text. Instead of storing the actual image, the text from the image is stored. This can be useful if other software or programs are used that would want to utilize the text.

Before the image is scanned it may need to go through some pre-processing. Depending on the image, the pre-processing may include a different amount of methods. These processing methods are used to make it easier to extract the desired data from an image. Here are the processing methods used in the thesis [21].

- **Inverted image**: This process inverts the pixels colors, such that, white pixels turn black and black pixel's turn white.
- **Rescaling**: Changes the scale of video or image.
- **Binarization**: Converts image to black and white.

[1] https://github.com/opencv/opencv

- **Noise removal**: Removes pixels, text, or other data that is irrelevant to the data that is needed to be extracted from the image.
- **Dilation and Erosion**: Makes for example text thicker (dilate) or thinner (erosion).
- **Rotation/Deskewing**: Rotates the image to the wanted angle.
- **Removing borders**: Removes empty space of the outline on the image.
- **Missing borders**: Adds borders to the image, used when text is right at the border of the image.

Two algorithms that are used for OCR are Pattern matching and Feature extraction. Pattern matching tires to capture a character in the image and then it compares it with other characters from a database. The problem with pattern matching is that the font needs to be similar to the font used in the database for this to work efficiently. Feature extraction however takes a character and divides it into different segments like lines, line intersections, and closed loops. Then it tries to match the divided character with something that looks similar from a database [22].

2.5.3 Tesseract

Tesseract is an open-source OCR engine that was developed at HP between 1984-1994 which utilizes OCR to recognise text from images [23].To use Tesseract within Python there is a library called Pytesseract [24].

2.5.4 Object detection

Object detection algorithms are pivotal in computer vision and can generally be categorized into two main groups based on their operational methodologies [25]:

- **Classification-based Algorithms:** This approach operates in a two-step process. Initially, regions of interest (ROIs) within the image are identified. Subsequently, these selected regions are classified using convolutional neural networks (CNNs).

A notable characteristic of this method is its relatively slower performance, attributed to the necessity of running a prediction for each identified region. Examples of classification-based algorithms include Region-based Convolutional Neural Networks (R-CNN) and its more advanced iterations, Fast R-CNN and Faster R-CNN.

- **Regression-based Algorithms:** Diverging from the classification-based approach, regression-based algorithms simultaneously predict both the classes and the bounding boxes for objects within an image in a single execution of the algorithm. This method is known for its efficiency and speed, as it eliminates the need to individually select regions of interest. A prominent example of this type of algorithm is YOLO (You Only Look Once), which is celebrated for its rapid processing capabilities [26].

Both approaches offer unique advantages and are chosen based on the specific requirements and constraints of the application at hand.

2.5.5 You only look once

YOLO simplifies object detection by treating it as a straightforward regression problem [2]. It processes an input image to simultaneously predict the classes of objects present and determine their locations with bounding box coordinates. Unlike other methods that apply a model across various sections and scales of an image, YOLO uses a single neural network for the entire image [26]. This approach involves dividing the image into a grid of regions. For each region, the network predicts bounding boxes and assigns a probability to these boxes, indicating the likelihood of an object's presence.

The probabilities are then utilized to prioritize the bounding boxes, with higher scores indicating a greater confidence in the detection. To finalize the detection process, YOLO filters out the bounding boxes, keeping only those with probabilities above a certain threshold. This efficient method enables YOLO to achieve real-time object detection, making it exceptionally fast and effective for applications requiring immediate processing [26].

2.6 Evaluating the model

The evaluation of the performance of a machine learning model is crucial to ensure its effectiveness and accuracy before deployment in real-world applications [6]. This evaluation process utilizes various metrics, each designed to assess different aspects of model performance. These metrics facilitate an understanding of how well a model has learned from training data and its ability to predict unseen data points. In tasks such as object detection and classification, which require nuanced interpretations of visual data, specific metrics are employed to quantify the precision and accuracy with which the model identifies and categorizes objects within an image. This section explores several key metrics that are essential for evaluating the robustness of machine learning models.

2.6.1 Intersection over union

Also known as the Jaccard Index [27], Intersection over Union (IoU) measures the overlap between the predicted bounding box and the actual ground truth bounding box to assess the accuracy of the prediction. The IoU score varies from 0 to 1, with a score of 1 indicating a perfect match where the predicted and ground truth bounding boxes align exactly. To validate object detections, a threshold IoU value can be established. Predictions with an IoU score exceeding this threshold are generally considered accurate and retained. This method provides a clear criteria for evaluating the precision of object detection models [27].

2.6.2 Confusion matrix

A confusion matrix is a widely utilized tool for assessing the performance of classification models [19]. It is applicable to both binary and multiclass classification scenarios. For instance, Table 2.1 illustrates a confusion matrix in the context of binary classification.

Table 2.1: Confusion matrix for binary classification

	Predicted Negative	**Predicted Positive**
Actual Negative	TN	FN
Actual Positive	FP	TP

In this matrix, the elements represent the number of predictions versus actual outcomes. TN refers to True Negatives, indicating correctly classified negative instances. TP represents True Positives, denoting accurately classified positive instances. Conversely, FP stands for False Positives, which are negative instances incorrectly classified as positive, and FN denotes False Negatives, reflecting positive instances wrongly classified as negative [28].

Rows correspond to the actual labels (ground truth) of the data, while columns mirror the labels as predicted by the model. Each cell quantifies the proportion of data the model classified into each category, with the highest values typically found in the diagonal cells [29], denoting correct predictions. Conversely, the off-diagonal cells highlight misclassifications, pinpointing errors in the model's predictions.

The term "background" in the confusion matrix of Figure 2.2, indicates true negative detections, where the model accurately identifies areas without any target objects. A prediction of "background" when a label should be present is considered a false negative, illustrating a missed detection by the model. Conversely, predicting a label where the actual situation is "background" results in a false positive, indicating an incorrect detection where none should exist [29].

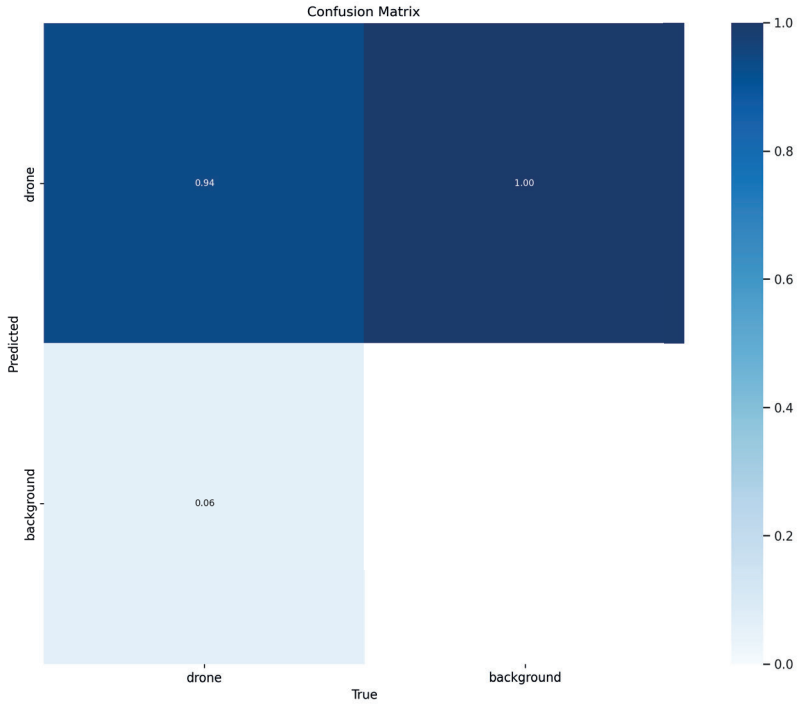

Figure 2.2: Confusion matrix by yolo

Accuracy is one of the primary metrics derived from a confusion matrix [28], calculated as the sum of true positive and true negative predictions divided by the total number of cases as seen in Equation 2.1:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

However, in the presence of imbalanced datasets, accuracy alone may not provide a complete picture of model performance [19]. This is because it can disproportionately reflect the majority class's influence. Thus, other metrics derived from the confusion matrix become crucial for a more comprehensive evaluation.

2.6.3 Precision and recall

Precision and recall are integral metrics for evaluating the performance of classification models [30]. Precision, also known as the positive predictive value, assesses the model's accuracy in predicting positive instances. It quantifies the proportion of true positive predictions in all positive predictions made. Recall, on the other hand, measures the model's ability to correctly identify all actual positive instances. It is synonymous with the model's sensitivity and evaluates the coverage of actual positive outcomes [19].

Improving recall without compromising precision is a common goal in model optimization [19], aiming for a balanced approach to classification accuracy. These metrics offer critical insights into the model's performance, particularly in scenarios where the costs of false positives and false negatives vary [30]. The computation of these metrics is based on the following formulas:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Precision scores fall within the range of 0 to 1, where a higher precision indicates a significant alignment between the detected objects and the actual ground truth objects [19]. For example, a precision of 0.8 means that when the model identifies an object, there is an 80% likelihood that this identification is accurate. This high precision score reflects the model's effectiveness in ensuring that the vast majority of its detections are indeed correct [28].

The recall metric varies between 0 and 1, with a higher score indicating a greater ability to correctly identify all relevant instances in the dataset [19]. For instance, a recall score of 0.6 suggests that the model successfully detects 60% of the actual objects. This means it has a high proficiency in capturing the majority of the ground truth objects within its predictions [28].

The Equations 2.2 & 2.3 provide a straightforward way to quantify the accuracy and sensitivity of classification models, enabling developers and data scientists to fine-tune their algorithms effectively.

2.6.4 Average precision

Average precision (AP) and mean average precision (mAP) serve as critical metrics for evaluating the efficacy of object detection models, providing a nuanced, unified measurement that encapsulates both precision and recall [31] [32].

Average precision: This metric synthesizes the model's performance into a single value by averaging the precision achieved at each recall level across the spectrum from 0 to 1. AP effectively measures how well the model identifies objects with a high degree of accuracy over the entire range of possible recall values, thereby capturing the essence of the Precision-Recall curve [31].

Mean average precision: For datasets encompassing multiple class categories (N classes), mAP refines the evaluation by averaging the AP calculated for each class, based on a defined IoU threshold for accurate detections. The computation of mAP unfolds in two primary stages [32]:

1. **Calculating AP for Each Class:** Determine the AP for every class by assessing the model's precision and recall at various thresholds, with consideration given to the specific IoU threshold that defines an accurate detection [32].
2. **Deriving the mAP:** The mean of these individual AP values across all classes produces the mAP, which succinctly reflects the model's overall precision and accuracy in detecting objects across different categories [32].

The YOLO model calculates two key metrics to evaluate its detection accuracy: mAP50 and [1] mAP50-95. The mAP50 metric stands for mean average precision at an IoU threshold of 0.50. This measurement assesses the model's accuracy by focusing on the simpler detections, often referred to as "easy" detections, where the required overlap

between the predicted and actual bounding boxes is minimal. On the other hand, mAP50-95 provides a more comprehensive evaluation by averaging the mean average precision at varying IoU thresholds, ranging from 0.50 to 0.95. This range includes varying levels of detection difficulty, thus offering a holistic view of the model's performance across different challenges in object detection.

Importantly, a higher mAP indicates that the model demonstrates greater precision and accuracy in its predictions. This metric is pivotal in quantifying the model's ability to not only correctly detect the presence of objects but also to minimize false positives and false negatives across all classes in the dataset. As such, mAP offers a comprehensive view of a model's performance, highlighting its efficacy in object detection tasks and guiding the optimization of detection algorithms for improved accuracy and precision.

2.6.5 Loss functions

Class loss and box loss are two critical metrics used in object detection models to evaluate their performance. Class loss measures how accurately the model predicts the correct class labels for each detected object [33]. It quantifies the discrepancy between the predicted class probabilities and the actual class labels, typically using a cross-entropy loss function. Lower class loss indicates better performance in classifying objects correctly [33].

Box loss, on the other hand, evaluates how well the model predicts the bounding boxes for detected objects. This involves calculating the difference between the predicted bounding box coordinates and the ground truth coordinates. A lower box loss signifies more precise localization of objects within the image [33].

Distribution Focal Loss (dfl) is an advanced loss function that enhances model performance by addressing the issue of class imbalance, where some classes are underrepresented compared to others. dfl dynamically adjusts the focus on harder, misclassified examples, giving them more weight during training. This adaptive approach helps improve the model's accuracy, particularly in datasets with significant class imbalances [33].

2.7 Natural language processing

Natural Language Processing (NLP) emerged in the 1950s [34] at the onset of AI. NLP is a way through technology to mimic human language processing [35]. NLP is used for different kinds of processing of texts such as translation, summarization, and assessing the purpose of a text [35].

2.8 Named entity recognition

Named Entity Recognition (NER) is an NLP method and the concept of NER started in the 1990s [36]. NER is used to identify so-called entities within a text. Entities are a group of terms, examples of entities could be companies, addresses, food, and planets. To identify the entities the NER method tries to find patterns in the text if not the word itself is enough to make an identification. To utilize NLP and NER methods in Python, you can use spaCy. spaCy is a free, open-source Python library that was initially released in early 2015 [36].

Chapter 3

Method

This chapter provides an overview of the various phases the thesis project is going through, as well as the specific methodology chosen by the authors to navigate through the project. Furthermore, it discusses the communication strategies employed for effective coordination and collaboration between the authors. A significant portion of the chapter is dedicated to exploring and explaining the rationale behind the decisions and choices made during the course of the work. This includes insights into how these choices have influenced the direction and outcomes of the project, as well as the challenges and solutions identified. The review aims to give the reader a deeper understanding of the work's structure and the considerations that have underpinned the research process.

3.1 Phases

This section starts with initial research and data collection, then moves to the detailed task of data labeling. This is followed by training our selected AI model. Next, we apply OCR technologies for data extraction, and conclude by organizing and storing the processed data into JSON objects for database insertion. Each phase is crucial, building upon the previous one to ensure a seamless flow and integration of processes, ultimately contributing to the robustness and efficiency of the project outcome.

3.1.1 Research and data collection

Achieving a comprehensive understanding of the project's objectives involves gathering a wide range of information, specifically about the variety of available AI models and

identifying which ones are suitable for the specific task. The requirement is to identify a model that excels in object detection and classification, specifically one that can efficiently locate and identify different types of data on invoices. YOLO version 8 (YOLOv8) is ultimately chosen for this purpose. For the scanning of invoices, OpenCV is utilized. The choice to employ both YOLO and OpenCV is driven by their compatibility within the same programming language, Python, eliminating the need to switch between languages.

Simultaneously, data collection for training and testing the model is undertaken, a critical phase underscoring that without adequate data, training the model to perform as desired is impossible [37]. Sources of this data included GitHub [38] and a cooperative company that provides two templates of their invoices for use in the project.

During the research phase, the specific labels or classes to be extracted from the invoices are identified. The established labels selected for use include:

- invoice# (invoice number)
- due_date
- company_detail
- customer_detail
- shipping_detail
- table
- table_total

These classes are chosen to capture the essential data points from the invoices for processing and extraction.

3.1.2 Labeling the data

After collecting a substantial dataset of approximately 500 invoice samples [1], the next step involves annotating these invoices to identify the aforementioned labels. This annotation can be accomplished using tools like LabelImg [39] or through an online AI

[1] for more information about the invoices see Figure 1 in Appendix A

training platform such as Roboflow [1]. Roboflow is the preferred tool due to its collaborative features [1], allowing the dataset to be shared and annotated jointly by the authors. Additionally, Roboflow offers robust data augmentation capabilities, such as converting images to grayscale or inverting them, which can significantly enhance the dataset by creating varied conditions for the model to learn from [10]. After completing the annotation and utilizing these augmentation features, the dataset is partitioned into training, validation, and testing sets with respective proportions of 80%, 15%, and 5%.

3.1.3 Training the YOLO model

Once a portion of the data, roughly around 100 invoices, is annotated with the various labels, the next step involved converting these annotated images into YOLOv8 file format. These files are then used to adapt the data into a format suitable for training the model. The Roboflow platform is used to convert the annotated dataset directly into a YOLOv8 file format [1].

The training process is conducted using a Jupyter notebook provided by Ultralytics on GitHub [38], offering a straightforward approach to model training [2]. Adjustments to the code are minimal, mainly involving the specification of the dataset from Roboflow and the modification of training epochs to 25. The initial training phase is relatively brief, lasting about 10 minutes, as it only utilizes a subset of the complete dataset. An additional benefit of using Roboflow is its capability to upload the model to the platform after partial training [1]. This feature enables the trained model to be employed to annotate the remaining invoices. This functionality greatly increases the efficiency of the annotation process, streamlining the workflow, and enhances productivity. After continuous annotation is concluded with the assistance of the model, a final training session is conducted using the entire dataset.

3.1.4 Data extraction

The data extraction process from invoices through OCR involves several stages of image handling and processing. This sequence begins with image loading and display, where the

[2]For more information about the invoices see Figure 3 to Figure 6 in Appendix A

function initiates by loading the invoice image from a specified path. This step confirms that the image is successfully loaded into the system, and the image is briefly displayed. This temporary visualization serves an essential function during the debugging or manual verification phases, allowing developers or operators to visually confirm that the correct document is being processed.

ROI extraction is the next step. Since invoice layouts can differ greatly, it is important to accurately and flexibly pinpoint the area in the image where the data needs extraction [37]. The ROI coordinates specified as (x1, y1, x2, y2) define this region as a rectangle. These coordinates are input parameters to the function and are crucial for accurately slicing the image array to isolate the ROI. Isolation is essential as it ensures that subsequent operations focus solely on the text-containing part of the image, enhancing processing efficiency and accuracy [37].

The image pre-processing for OCR involves several established techniques to prepare the extracted ROI for text recognition. These steps include resizing the ROI to a uniform dimension (typically 500x500 pixels), converting the image to grayscale, and applying binary thresholding to enhance text visibility. Additionally, advanced techniques such as inversion, dilation, and noise reduction are used to improve the prominence of text and clarity of the image.

After these pre-processing steps, the prepared ROI undergoes text recognition. The image is processed by the Tesseract OCR engine through the Pytesseract interface, which scans the image and extracts textual data, converting it into a machine-readable string. This string represents the crucial data extracted from the invoice, ready for further processing and analysis [3].

3.1.5 Training the NER model

To train the NER model, the text data from the invoices is extracted and saved into a text file. Each row in this file represents a single instance of the extracted data. For example, if company details were extracted, each row would include the company name,

[3]For more information see Figure 7 in Appendix B

address, and email.

To label the data, the file is loaded into a designated website [4]. On this website, different labels such as company name, address, and email are created. These labels are then applied to the corresponding words in the invoice text see Figure 3.1. When all the data is annotated, a file can be downloaded from the website. This file will be used to train the model.

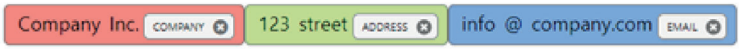

Figure 3.1: Named Entity Recognition example

3.1.6 Creating JSON objects

After successfully extracting textual data from invoices using OCR, the next essential step involves structuring this data into a usable format and storing it in a database for further processing and analysis. This process is facilitated through the transformation of data into JSON objects and the subsequent use of a MySQL connector package in Python for database operations.

The extracted data from invoices, which includes details like sender, receiver, due date, total amount, and invoice number, is first encapsulated into JSON objects. This structuring is crucial as JSON offers a flexible, text-based format that easily integrates with web applications and supports hierarchical data structures. The Python classes such as Invoice, Sender, Receiver, and related classes are used to organize the data into this format.

Each Invoice instance captures comprehensive details of individual invoices, and through the InvoiceToJson class, these instances are converted into JSON format [5]. The conversion process involves collecting all invoices into a list, which is then iterated over to create

[4] https://tecoholic.github.io/ner-annotator/

[5] For more information see Figure 8 and Figure 9 in Appendix B

a JSON-friendly dictionary for each invoice. Similar processes are employed for sender and receiver details using their respective classes.

3.1.7 Testing approach

Testing is conducted separately for each component of the implementation and then collectively to ensure that all parts function seamlessly together. This approach allows for immediate identification and resolution of specific issues in each component before integrating the entire system.

3.1.8 Object detection model testing

The object detection model is tested on its own to verify its accuracy in detecting and classifying various data types on invoices. This is crucial for ensuring that the model reliably identifies and categorizes elements like invoice numbers, due dates, and other specified classes without human intervention. The model's performance metrics, such as precision and recall, are monitored continuously to assess its effectiveness throughout the training and deployment phases.

3.1.9 Information filtering testing

Every component of the information filtering process, developed to refine and authenticate the data captured by the object detection model, is subjected to thorough testing. This involves ensuring the precision of data parsing and the validation of the extracted data against established formats. Additionally, the integrity of the final output is confirmed. The primary objective is to enhance the accuracy of the extraction process, particularly when converting data into JSON format, ensuring that each piece of information is correctly categorized and stored. This rigorous validation helps to eliminate irrelevant or incorrect data, significantly elevating the quality and reliability of the final data output.

3.1.10 Integrated system testing

Following the testing of individual components, a comprehensive evaluation of the entire integrated system is carried out. This final phase of testing involves processing complete invoices, from object detection to data extraction and conversion into JSON format. The

evaluation assesses the system's capacity for seamless end-to-end data extraction and processing across a range of conditions, including different invoice layouts and levels of data complexity. This ensures the system's robustness and adaptability in effectively handling real-world scenarios.

3.2 Documentation

The documentation for this report is carried out in parallel with the project's ongoing activities. The process begins with an initial description, followed closely by a research phase, leading seamlessly into the implementation of the microservice.

Documentation of the technical background for each part is initiated as soon as its implementation is finalized. This timely approach to writing is advantageous because it allows for the immediate recording of detailed and accurate information while the subject matter is still fresh, eliminating the need to backtrack to earlier phases of the project.

Once the technical aspects are documented, the focus shifts to compiling and analyzing results, which naturally progresses into the writing of the conclusions section. Throughout the project, updates to references are made to keep pace with the evolving research, ensuring that all aspects of the report are current and comprehensive. Additionally, adjustments to the report's format are made as needed to improve overall coherence and readability.

3.3 Source criticism

The foundation of the decisions made in this project, particularly in terms of model selection and implementation strategies, is established on high-quality and trustworthy sources. In the process of acquiring knowledge, the authors strictly utilize information from reviewed materials such as peer-reviewed academic journals and books. This is achieved to deepen understanding of specific concepts and investigate innovative solutions. Any information lacking verification from reputable sources is purposefully left out of this report, ensuring the integrity and reliability of the research undertaken.

Chapter 4

Result

This section presents the outcomes of the training and evaluation of the invoice extraction model, detailing its performance across various classes. We employ confusion matrices to explain the model's predictive accuracy and identify areas requiring refinement. Additionally, the training and validation dynamics over multiple epochs are analyzed to measure the model's progression and its adaptation to the dataset.

4.1 Initial training of YOLO

For this preliminary run, the model is trained on a dataset comprising 80 invoices. This initial dataset serves as a foundational test to verify the setup and functionality of the model, as well as to enable the model to annotate additional invoices for inclusion in the dataset. This first phase of training and validation provides critical insights into which classes are most prone to errors, guiding subsequent adjustments and improvements in the model's training process. The initial evaluation of our model employs a confusion matrix, a crucial tool for analyzing the performance and accuracy of our classification system.

From the confusion matrix of Figure 4.1, several key observations can be made regarding specific classes:

1. The model demonstrates strong performance in identifying the "Table" class, with a high score of 0.75, suggesting a high level of reliability in this area.

2. Significant errors are noted in the "Company details" and "Customer details" classes, with respective scores of 0.20 and 0.25. The frequent confusion between these classes indicates a need for focused improvement.

3. The classes "Invoice", "due_date", and "shipping_detail" exhibit the lowest accuracy, with scores hovering around 0.0, indicating significant challenges in accurately recognizing these vital invoice elements.

These observations suggest that while the model is capable of accurately identifying some invoice elements, considerable improvements are necessary, particularly in the areas of company and customer details, as well as in the accurate extraction of invoice totals and due dates.

This preliminary evaluation serves as a baseline for future improvements. As the training dataset is expanded and the methodologies of the model are refined, an enhancement in performance across all categories is anticipated. This continuous refinement will help achieve more precise and reliable model predictions.

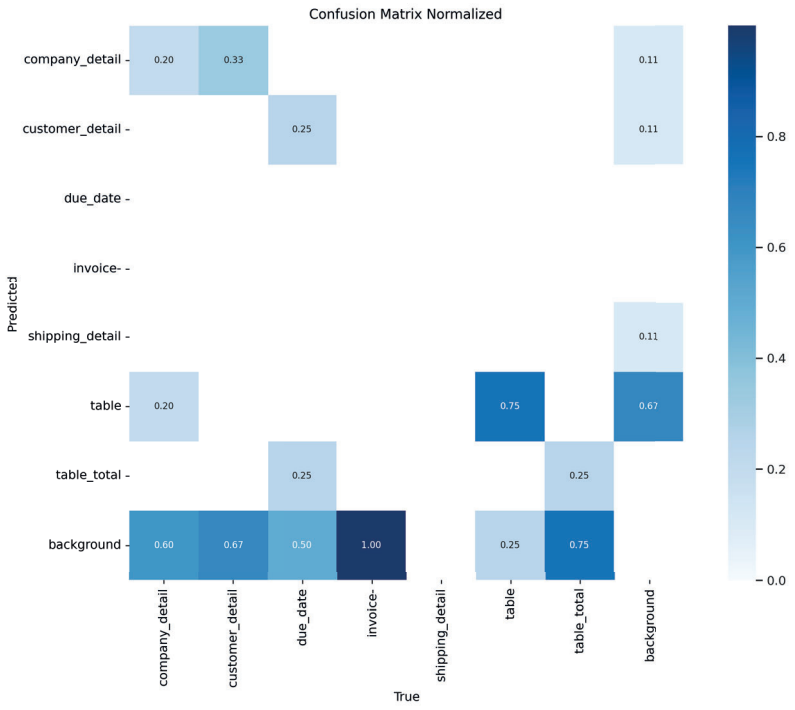

Figure 4.1: Initial normalized confusion matrix

During the evaluation of the invoice extraction model across multiple epochs, the training and validation losses, as well as various metrics, are systematically recorded. The model undergoes significant training iterations, evident from the values across the epochs. The x-axis in all graphs represents the number of epochs, ranging from 1 to 25, with each epoch signifying one complete pass through the entire training dataset, thereby providing a measure of the training duration and progression. The y-axis showcases the values of the different loss functions and metrics being monitored. These include box loss, class loss, and distribution focal loss. The training losses for box, class, and dfl of Figure 4.2, showing an overall trend of decrease across epochs as presented in Table 4.1. For instance, the training box loss reduces from 2.4544 in epoch 1 to 1.2131 by epoch 25. Similarly, the class loss decreases from 7.8587 to 1.8756, and the dfl loss from 2.1186 to 1.2678 over the same period as presented in Table 4.1.

Table 4.1: Initial train performance metrics by epoch

Epoch	Train/Box Loss	Train/Class Loss	Train/Dfl Loss
1	2.4544	7.8587	2.1186
2	2.4762	7.372	2.1184
3	2.2979	7.1205	2.0617
4	2.5142	7.1292	2.1306
5	2.3706	5.7302	2.1098
⋮			
23	1.0943	1.8103	1.2229
24	1.1458	1.8362	1.2919
25	1.2131	1.8756	1.2678

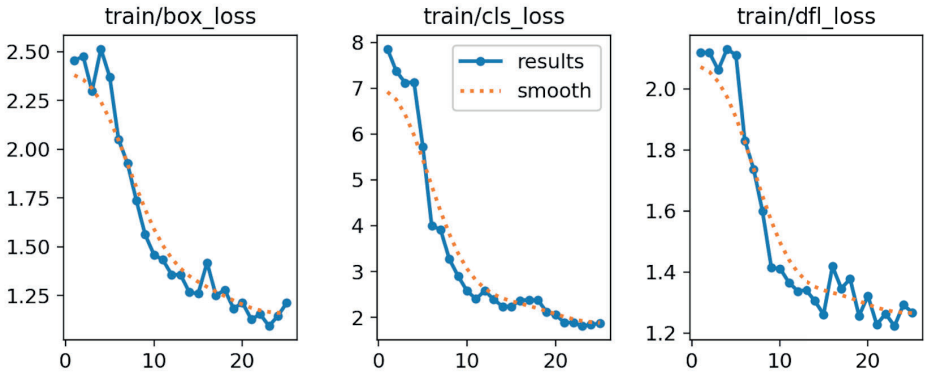

Figure 4.2: Graphical representation of train loss metrics during initial model training

In terms of validation, the box loss shows a general decrease, while the class loss exhibited considerable variation, peaking significantly in certain epochs (e.g., 47.796 in epoch 7) before tapering off to 3.5622 by epoch 25, as it is reported in Table 4.2. The dfl loss in validation also trends downward overall as seen in Figure 4.3.

Table 4.2: Initial model validation performance metrics by epoch

Epoch	Val/Box Loss	Val/Class Loss	Val/Dfl Loss
1	2.4113	6.9616	2.1081
2	2.4288	7.0176	2.1404
3	2.4176	7.2546	2.1674
4	2.4092	7.3089	2.1404
5	2.4765	8.5674	2.2334
6	2.4623	14.897	2.255
7	2.7816	47.796	2.3215
⋮			
23	1.7005	3.7832	1.6035
24	1.6793	3.6213	1.5905
25	1.6333	3.5622	1.5854

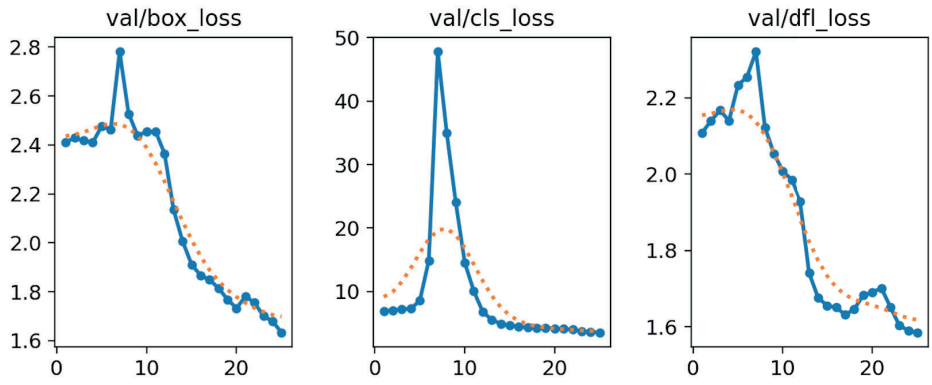

Figure 4.3: Graphical representation of validation loss metrics during initial model training

The precision metric evolves from a low of 0.01129 in the first epoch to a more robust 0.53783 by epoch 25 as seen in Table 4.3, indicating an improvement in the model's ability to correctly identify positive instances. The recall, which measures the model's

capability to detect all relevant cases, similarly showed progress, particularly notable in the significant increase in mAP50 and mAP50-95 scores from the earlier epochs to later stages, as depicted in Figure 4.4. For example, mAP50 increases from 0.015 to 0.28006, and mAP50-95 from 0.00306 to 0.142. This is equivalent to 28% and 14% respectively by the end of the training sessions.

Table 4.3: Initial evaluation metrics across epochs for the model

Epoch	**Metrics/Precision**	**Metrics/Recall**	**Metrics/mAP50**	**Metrics/mAP50-95**
1	0.01129	0.09722	0.015	0.00306
2	0.01209	0.09722	0.00968	0.002
3	0.00019	0.04167	0.00015	0.00009
4	0.00019	0.04167	0.00015	0.00009
5	0.67549	0.04167	0.00305	0.00174
		⋮		
23	0.55245	0.25227	0.28586	0.13742
24	0.53782	0.29689	0.28729	0.13478
25	0.53783	0.29722	0.28006	0.142

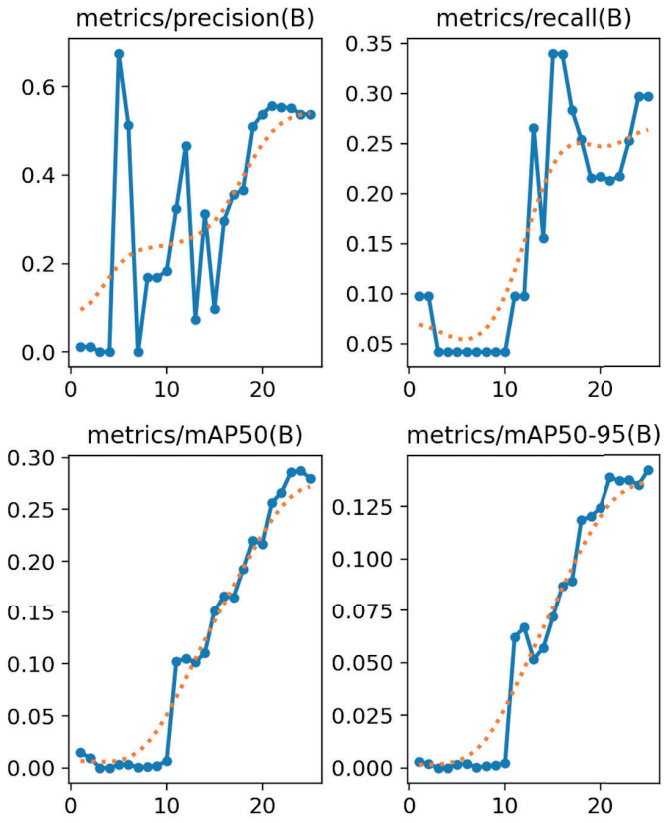

Figure 4.4: Trends in evaluation metrics across epochs from initial model performance

4.1.1 Final training of YOLO

As we expand the dataset and iteratively refined our model, we observe a notable enhancement in its predictive accuracy and reliability. This is evidenced by the gradual improvement in mAP scores and the reduction in class-specific errors, the model demonstrates commendable performance in accurately classifying several key invoice elements, see Figure 2 in Appendix A, as depicted in the confusion matrix of Figure 4.5. Notably, the classes "Company_detail", "Customer_detail", "Table", and "Table_Total" exhibited high values (around 0.9 or 1.0) on the diagonal, indicating that the model effectively recognized and categorized the majority of data points in these areas. In contrast, classifications such as "Due date," "Invoice#," and "Shipping detail" presented lower diagonal values (around 0.7 or 0.8), suggesting the model encountered more challenges in these

specific categories.

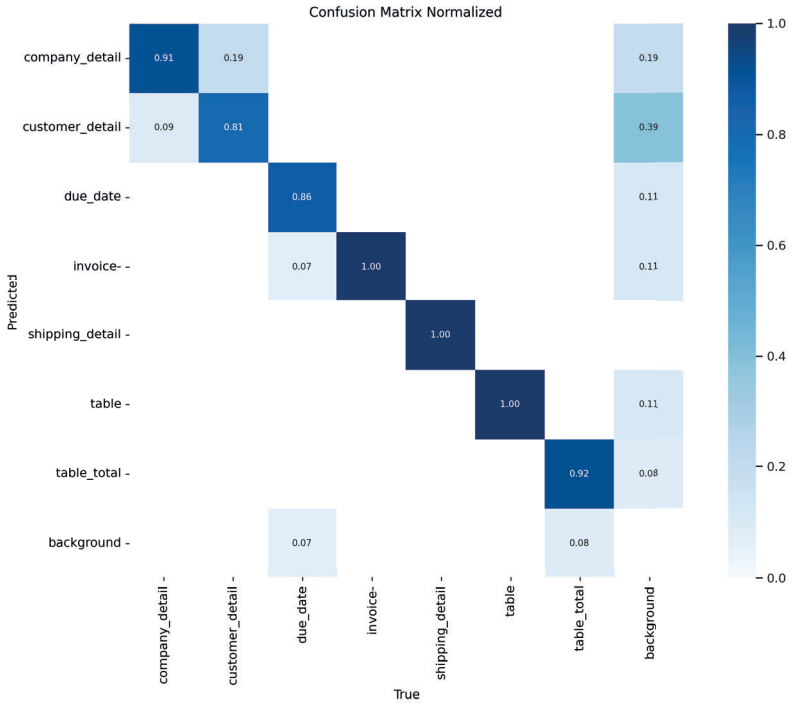

Figure 4.5: Final normalized confusion matrix

During the initial training phase, the model faced relatively high losses. Specifically, in the first epoch, training losses were recorded for bounding box, class, and dfl at 1.8148, 3.548, and 1.6402, respectively, as presented in Table 4.4. These figures gradually decreased over subsequent epochs, see Figure 4.6, reflecting the model's evolving capability to interpret and learn from the training data effectively.

Table 4.4: Final model train performance metrics by epoch

Epoch	Train/Box Loss	Train/Class Loss	Train/Dfl Loss
1	1.8148	3.548	1.6402
2	1.3634	1.8638	1.3154
3	1.3781	1.7321	1.3363
4	1.324	1.5978	1.2938
5	1.2556	1.4414	1.2609
		⋮	
23	0.92313	0.65106	1.0887
24	0.89865	0.6205	1.0792
25	0.8793	0.61117	1.0732

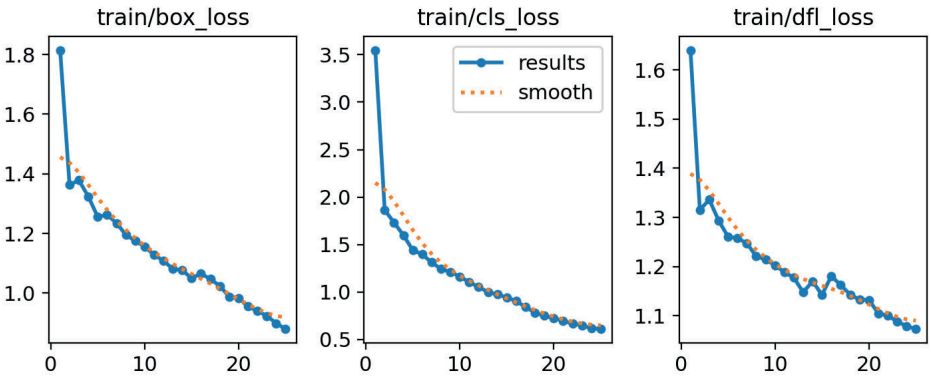

Figure 4.6: Graphical representation of train loss metrics during final model training

For instance, training and validation losses displayed a consistent decline over 25 epochs, demonstrating the model's improved ability to learn from the data. The training losses for bounding box, class, and dfl by the 25th epoch had decreased to 0.8793, 0.61117, and 1.0732 respectively, as seen in Table 4.4 and Figure 4.6, showcasing substantial improvements. Similarly, validation losses mirrored this trend as seen in Figure 4.7 and Table 4.5, contributing to a growing confidence in the model's predictive accuracy.

Table 4.5: Final model validation performance metric by epoch

Epoch	Val/Box Loss	Val/Class Loss	Val/Dfl Loss
1	1.3326	2.6805	1.3667
2	1.46	1.7951	1.4986
3	1.4192	1.7017	1.4615
4	1.4736	1.7647	1.4946
5	1.2938	1.3828	1.3557
⋮			
23	0.96094	0.60175	1.1475
24	0.92937	0.56566	1.1359
25	0.92269	0.56539	1.1416

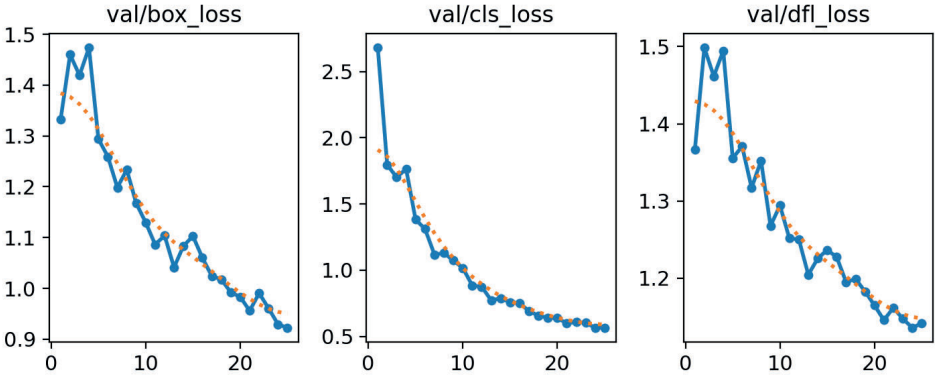

Figure 4.7: Graphical representation of validation loss metrics during initial model training

Moreover, there was a significant advancement in the precision and recall metrics. Starting from moderate values, the precision and recall progressively reached peaks of 0.91337 and 0.93389 by the 25th epoch as seen in Table 4.6 and Figure 4.8. The increase in mAP score of 95.3% further underscored the model's enhanced accuracy and robustness, indicating its reliable capacity to detect and classify diverse invoice elements across varying categories.

Table 4.6: Final evaluation metrics across epochs for the model

Epoch	Metrics/Precision	Metrics/Recall	Metrics/mAP50	Metrics/mAP50-95
1	0.53939	0.38425	0.36472	0.23302
2	0.48771	0.4922	0.44229	0.26554
3	0.49007	0.37845	0.457	0.27535
4	0.50512	0.51606	0.53116	0.30723
5	0.57689	0.64464	0.67647	0.41382
		⋮		
23	0.84932	0.93472	0.94143	0.67063
24	0.91388	0.90121	0.94514	0.68276
25	0.91337	0.93389	0.95331	0.69268

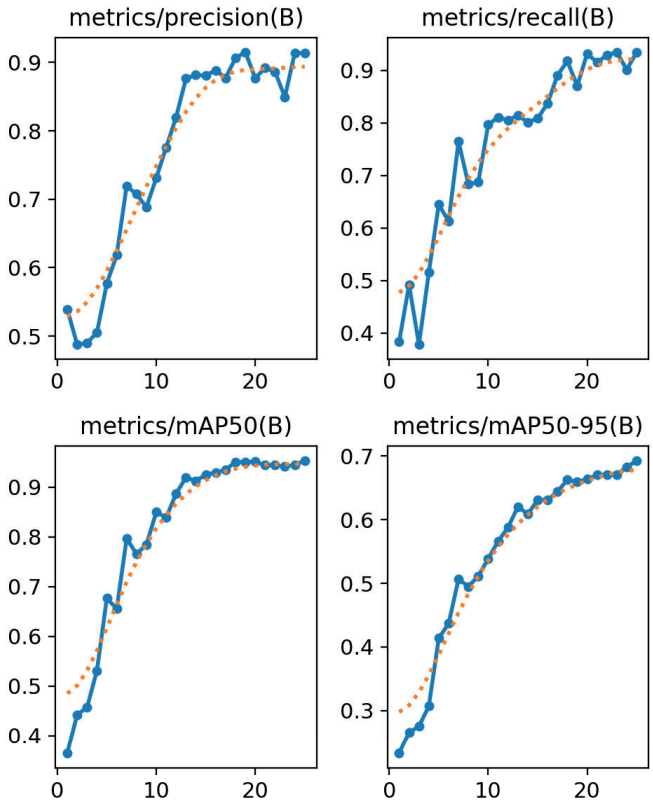

Figure 4.8: Trends in evaluation metrics across epochs from final model performance

4.1.2 Training NER

In the Tables 4.7, 4.8 and 4.9, the data represents the results obtained from training various NER models. The information displayed includes the following metrics: Epochs (number of training epochs), (number of invoices processed, ENT_P (entity precision), ENT_R (entity recall), and ENT_F (entity F-score, a mean of precision and recall).

Table 4.7 displays the results obtained from training the NER model to identify company details. The model is trained using a dataset comprising information from 500 invoices, achieving the current level of performance.

Table 4.7: Entity recognition performance metrics by epoch for company detail

Epoch	**#**	**ENTS_F**	**ENTS_P**	**ENTS_R**
0	0	0.00	0.00	0.00
13	200	97.31	97.31	97.31
29	400	98.49	98.32	98.65
49	600	99.83	99.66	100.00
73	800	99.83	100.00	99.66
103	1000	99.83	99.66	100.00
140	1200	99.83	100.00	99.66
183	1400	99.83	99.66	100.00
234	1600	99.83	99.66	100.00
300	1800	99.83	100.00	99.66
373	2000	99.83	99.66	100.00
473	2200	99.83	99.66	100.00

The data presented in Table 4.8 shows the results from the initial training of the NER model to identify various table items. This training utilizes information from 300 invoices.

The results presented in Table 4.9 are from the second training run of the table items

Table 4.8: Initial entity recognition performance metrics by epoch for table items

Epoch	**#**	**ENTS_F**	**ENTS_P**	**ENTS_R**
0	0	4.19	4.80	3.71
6	200	97.32	97.29	97.36
15	400	99.29	99.29	99.29
25	600	99.18	99.21	99.14
37	800	100.00	100.00	100.00
52	1000	100.00	100.00	100.00
71	1200	100.00	100.00	100.00
94	1400	100.00	100.00	100.00
121	1600	100.00	100.00	100.00
154	1800	100.00	100.00	100.00
194	2000	100.00	100.00	100.00
244	2200	100.00	100.00	100.00
305	2400	100.00	100.00	100.00

model. For this iteration, a dataset containing information from 500 invoices is utilized, incorporating a broader range of formats to enable the identification of different types of table items.

Table 4.9: Final entity recognition performance metrics by epoch for table items

Epoch	**#**	**ENTS_F**	**ENTS_P**	**ENTS_R**
0	0	0.00	0.00	0.00
4	200	87.90	88.00	87.81
9	400	92.94	92.89	92.99
15	600	94.50	94.50	94.50
22	800	97.95	97.95	97.95
32	1000	97.52	97.52	97.52
43	1200	98.27	98.27	98.27
57	1400	98.60	98.60	98.60
74	1600	98.71	98.71	98.71
94	1800	98.71	98.71	98.71
119	2000	98.71	98.71	98.71
150	2200	98.71	98.71	98.71
187	2400	98.71	98.71	98.71
227	2600	98.71	98.71	98.71
267	2800	98.71	98.71	98.71
307	3000	98.71	98.71	98.71
347	3200	98.71	98.71	98.71

Chapter 5

Analysis

This chapter delves into an analysis of the results and addresses the research questions (RQs) posed in Chapter 1. In this part, we explore various strategies that aim to refine the performance and adaptability of our invoice reading model. This analysis inspects the model's performance, identifying strengths and pinpointing areas that require improvement, from class imbalances to handling diverse document types and language considerations.

Each subsequent section presents targeted discussions on specific aspects of model optimization, ranging from the technical adjustments to strategic overhauls. We assess the effectiveness of different approaches to model training and adaptation, ensuring each strategy aligns with the practical demands of real-world invoice processing.

5.1 Class imbalance solutions

After achieving the results, an important observation from both the initial and final evaluations of the invoice processing model reveals consistent underperformance in specific classes, notably 'Due date,' 'Invoice,' and 'Shipping detail.' Despite various iterative improvements and adjustments over the training epochs, these classes consistently show lower accuracy. This recurring issue largely stems from the inconsistent appearance of these labels on invoices.

Often, these crucial classes are not only formatted differently across various invoice types

but are also less frequently represented in the training dataset compared to other labels. This irregular presence and variability challenge the model's ability to learn and generalize effectively, leading to diminished predictive accuracy for these categories.

To tackle this issue, it is essential to enrich the training dataset with a more diverse array of examples that include these underrepresented labels. Expanding the range of data to include a broader and more balanced representation of all invoice classes is critical. Enriching the dataset in terms of label frequency and diversity facilitates a more comprehensive learning process, enabling the AI model to perform more consistently across all categories of invoice data.

Further enhancing the dataset not only helps reduce the model's bias towards more frequently seen labels but also strengthens its capability to handle currently challenging labels due to their sparse occurrence. Dataset augmentation, involving systematic modifications to training images, becomes a practical approach to creating new training samples from existing data. Techniques such as flipping images horizontally can simulate different invoice orientations, which is useful for training the model to recognize variably oriented text. Scaling techniques can mimic invoices being closer or farther from the camera, adjusting for size differences. Additionally, modifying color variations aids the model in managing different lighting conditions and color schemes, which are common in real-world scenarios. These enhancements aim to make the model robust against the diverse formats and presentations of invoices, ultimately improving its accuracy and reliability in commercial applications.

5.2 Language considerations

The desired output for invoice extraction varies depending on whether the invoices are in Swedish or English. When identifying ROI coordinates, the model primarily focuses on invoice patterns rather than the language itself. It analyzes where different types of information are typically located and in what format. However, language still influences the search results. Additionally, the overall format of Swedish and English invoices often differs, which affects the search in a significant way.

The text extraction model, powered by OpenCV, is not affected by language differences because it autonomously identifies letters. Since Swedish and English use the same alphabet, language is inconsequential in this context.

The model most affected by language is NER. NER actively seeks specific words and patterns within the text, such as those used to identify companies, addresses, or emails. These patterns vary significantly between Swedish and English addresses and companies. However, email formats remain consistent regardless of language, as emails follow a standardized structure of text.

5.3 Strategies for document type handling

There are different approaches to handling various document types efficiently. One method involves using distinct models for each type of document. Depending on the type of document to be scanned, a specific model would be employed. This strategy often leads to higher success rates as each model specializes in a particular document type. However, a drawback is the need to determine which document types to include and how many. Developing multiple models can be time-consuming and requires a substantial amount of data collection for each model.

Alternatively, another approach is to standardize different document types into a single format. By converting all document types into a unified format, only one model needs to be trained. Several open-source libraries are available that can perform document conversions (e.g., from PDF to JPG). Even if a specific conversion library is not readily available, developing a custom converter could be less time-intensive. However, a challenge with this method is the potential loss of data and reduction in quality during the conversion process. For instance, when converting a PDF (which uses vector graphics) to a JPG (which is pixel-based), the loss of detail may occur [40]. Unlike PDFs that can be zoomed without quality loss due to their mathematical representation, JPGs lose quality when zoomed in, as they consist of fixed pixels. The reverse is also true. Zooming out in a JPG can make the content unrecognizable due to pixelation.

Additional models could be incorporated based on the desired information extraction from invoices. Initially, during the system's implementation, YOLO was utilized solely to extract the ROI coordinates from the invoices. Subsequently, openCV was employed to extract text from these ROI coordinates. In certain scenarios, this extracted information might suffice for the system's requirements, negating the need for further development. However, in our case, we aimed to refine this extracted data by identifying specific text using NER.

5.4 Refining the invoice reading model

As we look towards the continuous improvement of the invoice reading model, this section outlines a range of potential strategies that are being considered to refine its performance and adaptability. These strategies are theoretical enhancements not yet implemented but represent possible avenues for future development. Each strategy targets specific aspects of the model's functionality. This includes rigorously evaluating performance metrics, enhancing the quality and diversity of the training dataset, and managing the model's complexity. This section proposes a systematic approach to optimizing the model, ensuring it is better equipped to handle the complexities of real-world invoice processing.

5.4.1 Model evaluation and tuning

The evaluation of the model's performance utilizes essential tools such as confusion matrices and key performance metrics including precision, recall, and mAP scores. These evaluations are critical for identifying the model's strengths and pinpointing specific areas that require improvement, particularly in accurately classifying complex invoice elements like 'Due date', 'Invoice', and 'Shipping detail'. Through regular analysis of these metrics, targeted enhancements are systematically applied, allowing for precise fine-tuning of the model to address its specific weaknesses effectively.

5.4.2 Enhancing data quality

At the core of any machine learning model is its dataset. The efficiency of the invoice reading model highly depends on the quality and diversity of its dataset. Accuracy in

labeling and a diverse range of image data are crucial. High-quality, accurately labeled images ensure the model learns the correct features and relationships essential for precise predictions upon deployment.

Moreover, the diversity in the dataset, encompassing variations in lighting, angles, and backgrounds of invoice images, is vital for the model's ability to generalize effectively to new, unseen data. Invoices from varied sources exhibit different formats, colors, and conditions, including wear or distortion. Incorporating a wide spectrum of these variations into the training dataset is a strategic move to combat overfitting, a prevalent challenge in machine learning that can impede the model's performance in real-world scenarios.

5.4.3 Managing model complexity

The complexity of the model is fundamentally linked to its ability to process and learn from data comprehensively. For intricate tasks such as invoice reading, which demands handling multiple formats and detailed content, employing advanced architectures like YOLOv8 'l' (large), 'x' (extra large), or YOLOv9 is advantageous. These sophisticated models are designed to capture finer details and subtle nuances, which can significantly enhance accuracy in object detection and classification tasks.

However, deploying more sophisticated models comes with increased computational demands. This can potentially impact the microservice's efficiency, particularly in environments with limited computational resources like processing power, memory, and storage. The primary challenge lies in achieving the right balance between model complexity and computational efficiency. This balance is crucial to ensure that the model is capable enough to deliver high accuracy without exceeding the resource constraints of its deployment environment, thereby maintaining optimal processing speeds.

5.4.4 Hyperparameter optimization

Further enhancement of the model can be achieved through meticulous hyperparameter tuning. Adjusting key parameters like learning rate, weight decay, and momentum in response to trends observed in performance metrics such as loss rates and accuracy is essential. This continuous optimization process tailors the training regimen to maximize

the model's performance across various training epochs. Each iteration of training not only aims at improving precision and reducing errors but also to enhance the model's capacity to adapt dynamically to complex data landscapes.

5.5 Transition to production

Transitioning a developed AI service from the development or testing phase to a production environment marks a pivotal stage in the lifecycle of any technological solution. The production phase is where the service becomes fully operational and integrates into real-world business operations. This phase is essential as it rigorously tests the application's functionality, reliability, and usability under typical business conditions, serving as the ultimate test of its readiness and efficacy.

During this transition, various compatibility issues surface, necessitating additional customizations or modifications to the AI service or the existing enterprise systems. Such adjustments are crucial to ensure seamless data flow and functionality, integrating the new technology smoothly with the old to create a cohesive system.

Moreover, handling invoices means dealing with sensitive financial information. When such systems move to a production environment, they are subjected to heightened data security and privacy concerns. It is imperative that the AI system is secure and adheres to relevant regulations, such as the General Data Protection Regulation (GDPR), to safeguard against data breaches and maintain user trust.

Additionally, the variability of real-world scenarios often presents unforeseen challenges that the system must be prepared to handle. These challenges may include processing poorly scanned invoices, adapting to unexpected invoice formats, or correcting data input errors. Building robustness and reliability into the system to perform consistently under all operational conditions is critical to its success.

Thus, moving to the production phase is not merely a procedural step but a significant milestone that signifies the AI service's readiness for full-scale operational deployment.

This phase requires careful preparation and a well-defined strategy to navigate the complexities of real-world applications. Ensuring the system's performance meets the high standards of efficiency and effectiveness promised during its conception is essential for its long-term success.

Chapter 6

Conclusion

This section synthesizes the key findings and insights derived from addressing RQ 1 to RQ 5 posed in this study. By examining various aspects of implementing and optimizing an AI-driven invoice reading service, this conclusion offers valuable reflections on the technical considerations, performance enhancement strategies, and practical implications for real-world deployment.

- **In what way can the invoice reading service be implemented to handle different document types and easily expand with new models?**

When aiming to manage multiple document formats efficiently, adopting a standardized approach for handling a specific format is the preferred strategy. By concentrating on a single format, the number of required models is reduced. This is leading to a more streamlined and less time-consuming system development process for file conversion, as discussed in Section 5.3. This is particularly advantageous given the variety of open-source Python libraries available for file conversion tasks. With a narrowed focus on one format, the data collection efforts can be more targeted, requiring less overall data acquisition for training a model. This approach optimizes both development efforts and resource utilization.

- **What technical and language aspects need to be considered to ensure accurate interpretation of invoices in Swedish and English?**

When addressing language considerations, a primary factor to consider is the performance

of the NER model. As mentioned in Section 5.2, one of the key challenges faced by the NER model is the varying formats between English and Swedish text. This issue can be mitigated by training the model on a more extensive dataset that encompasses both languages and a wider range of text formats. By incorporating diverse data during training, the model´s ability to accurately identify the desired information will significantly improve.

- **What methods and criteria should be used to train and evaluate the AI model used for invoice reading?**

The methods and criteria for training and evaluating the AI model used for invoice reading are paramount for its effectiveness. As discussed in Section 5.1, addressing class imbalance through dataset enrichment is crucial. By incorporating a diverse range of examples, particularly for underrepresented classes the model can learn to generalize more effectively across various invoice formats. This aligns with the analysis presented in Section 5.4.2, highlighting the importance of dataset diversity and augmentation techniques in improving model performance.

For evaluation, employing tools such as confusion matrices and key performance metrics including precision, recall, and mAP scores, as discussed in Section 5.4.1, is essential. These metrics facilitate a comprehensive understanding of the model's strengths and weaknesses, enabling targeted improvements. Regular evaluation using these tools ensures that the model is finely tuned and capable of accurately classifying complex invoice elements.

The training and evaluation of the AI model for invoice reading demands a comprehensive strategy that encompasses the use of sophisticated model architectures, rigorous data handling, and meticulous performance assessment. Through such an approach, the model not only achieves high accuracy but also adapts to the complexities of real world invoice processing tasks.

- **How can the performance and speed of the developed microservice be improved to meet the requirements for fast and efficient invoice reading?**

Improving the performance and speed of the developed microservice involves managing the balance between model complexity and computational efficiency. Deploying advanced architectures like YOLOv8 'l' or 'x' enhances the model's ability to detect and classify detailed invoice content accurately, as discussed in Section 5.4.3. However, these sophisticated models demand higher computational resources, which can impact the microservice's efficiency.

To address this, optimizing hyperparameters such as learning rate, weight decay, and momentum based on observed performance metrics is critical, as outlined in Section 5.4.4. This optimization tailors the model's training to its operational environment, maximizing performance without overwhelming system resources. Additionally, ensuring a diverse and high-quality dataset, as emphasized in Section 5.4.2, is crucial for preventing overfitting and maintaining high accuracy under varying real-world conditions. These strategies collectively contribute to a model that is not only accurate but also operates with the designated speed and efficiency.

- **What opportunities and challenges may arise when transferring the developed service to a production environment?**

Transitioning the AI service for invoice processing from a development stage to a production environment presents several challenges, as detailed in Section 5.5. Key issues include integration complexities with existing enterprise systems, strict data security demands, and the necessity to manage diverse real-world scenarios effectively. Strategic adjustments are crucial to ensure that the AI service integrates smoothly with existing systems, complies with rigorous data protection regulations such as GDPR, and is versatile enough to handle varying invoice formats.

Effectively addressing these challenges is vital for the AI service to fulfill its potential and operate reliably within a real-world context. This transition phase is not merely a technical step but a critical determinant of the AI service's long-term effectiveness and scalability in commercial environments. By successfully managing these hurdles, the service can demonstrate its value and readiness for broader operational deployment.

Future Work

In this part, we discuss the possible ways the system can evolve and methods to continue the work.

One crucial step to improve the system is to find more data. To improve and build an AI model, data is necessary to make the AI behave as desired. The data or invoices used for the project can be considered limited in many ways. Currently, the data used is only in English, which might make the model less accurate when encountering invoices in different languages. The invoices are also limited in variation of formats. Here, a different format would be defined by having various types of information placed differently. Having limited formats to train the model with could lead to the model wrongfully identifying different areas of information. The risk of this happening can be minimized by using a bigger variety of data in greater quantities, including invoices of the desired languages and, within each language, many different formats.

At the time of writing this thesis, the system can only scan JPG images, while the most common format for invoices is PDFs. Indirectly, the system can scan PDFs; there is a built-in code module within the system that checks whether there are any PDFs within the folder of JPG invoices. If a PDF is found, it is converted to a JPG. The problem with this type of conversion is that it could lead to a loss of information, which could make it more difficult for the system to accurately scan and identify the information within the invoice. A possible improvement would be to enable direct scanning of PDFs to negate the potential information loss by conversion.

Currently, the system lacks a defined endpoint for processing invoice data effectively. To address this, a database solution could be implemented to store and facilitate access to invoice information. This database will serve as a centralized repository for managing invoices, making it easier to retrieve specific ones as needed. Additionally, the system requires a user-friendly interface for uploading invoices for scanning. The program will utilize this database to efficiently send and retrieve invoices.

Ethical Reflection

In the development and implementation of AI technologies for invoice processing, several ethical considerations must be prioritized to safeguard human values and societal norms. Ensuring the privacy and security of sensitive data contained in invoices is crucial. This involves implementing strong data protection mechanisms to prevent unauthorized access and misuse, clearly detailing how the AI handles and securely stores data.

Transparency and accountability also play critical roles in building trust and reliability in AI applications. It is essential that the methods by which the AI interprets and processes invoices are transparent to all stakeholders, with clear accountability measures in place to handle any discrepancies or errors that occur. Moreover, the importance of user consent is fundamental. Users must have substantial control over their data, with clear protocols for how their consent is obtained and respected.

The impact of AI on human labor, particularly through the automation of routine tasks such as invoice processing, poses significant ethical questions. While AI can enhance efficiency, it is vital to consider how it changes job roles and responsibilities, ensuring that workers are fairly and effectively transitioned into new roles. Environmental considerations are also crucial. The energy consumption and carbon footprint associated with running AI models must be minimized to promote sustainability.

Finally, continuous monitoring and evaluation are essential to ensure the AI model adheres to ethical standards over time. Regular inspections and reviews can help in quickly addressing any emerging ethical concerns, thereby maintaining the integrity and societal acceptance of AI technologies in business processes.

Bibliography

[1] B. Dwyer, J. Nelson, and T. et al. Hansen. Roboflow (version 1.0) [software]. `https://roboflow.com`, 2024. Accessed: 2024-04-25.

[2] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics yolo (version 8.0.0) [computer software]. `https://github.com/ultralytics/ultralytics`, 2023. Accessed: 2024-03-10.

[3] D. Desai, A. Jain, D. Naik, N. Panchal, and D. Sawant. Invoice processing using rpa & ai. In *Proc. of the International Conference on Smart Data Intelligence (ICSMDI 2021)*, May 2021. Accessed: 2024-03-20.

[4] N. Hedberg. Automated invoice processing with machine learning: Benefits, risks and technical feasibility, 2020. Accessed: 2024-03-20.

[5] J. D. Kelleher, B. M. Namee, and A. D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. MIT Press, Cambridge, 2 edition, 2020. Accessed: 2024-03-20.

[6] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Shi jie tu shu chu ban gong si, Beijing, 2023. Accessed: 2024-03-20.

[7] N. Kraus, K. Kraus, O. Shtepa, M. Hryhorkiv, and I. Kuzmuk. Artificial intelligence in established of industry 4.0. `https://elibrary.kubg.edu.ua/id/eprint/43298/`, 2024. Accessed: 2024-04-04.

[8] J. Cho, K. Lee, E. Shin, G. Choy, and S. Do. How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?, 2015. Accessed: 2024-04-12.

[9] Gil Press. Cleaning big data: Most time-consuming, least enjoyable data science task, survey says, 2016. Accessed: 2024-04-12.

[10] Imran et al. A survey of datasets, preprocessing, modeling mechanisms, and simulation tools based on ai for material analysis and discovery. *MDPI*, 15(4):1428, 2022. Accessed: 2024-04-20.

[11] K. R. Srinath. Python - the fastest growing programming language issue 12 december 2017. *IRJET*, Dec 2017. Accessed: 2024-03-20.

[12] S. Weisberg. *Applied Linear Regression*. John Wiley, 3 edition, 2021. Accessed: 2024-04-11.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. General reference for deep learning and epochs, Accessed: 2024-04-12.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. Example study using image classification, Accessed: 2024-04-12.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. Discussion on the importance of epochs, Accessed: 2024-04-12.

[16] Douglas M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004. Study on overfitting, Accessed: 2024-04-12.

[17] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. Accessed: 2024-04-12.

[18] What is computer vision?: Microsoft azure. `https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision#object-classification`, 2024. Accessed: 2024-04-02.

[19] A. Kulkarni, D. Chong, and F. A. Batarseh. Foundations of data imbalance and solutions for a data democracy. *Data Democracy*, 2024. Accessed: 2024-04-11.

[20] G. Bradski and A. Kaehler. *Learning Opencv: Computer Vision with the Opencv Library*. O'Reilly, Beijing, 2011. Accessed: 2024-04-02.

[21] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029–1058, July 1992. Accessed: 2024-04-02.

[22] What is ocr? - optical character recognition explained - aws. `https://aws.amazon.com/what-is/ocr/`. Accessed: 2024-04-02.

[23] R. Smith. An overview of the tesseract ocr engine, 2007. Accessed: 2024-04-02.

[24] S. Hoffstaetter. Pytesseract. `https://pypi.org/project/pytesseract/`. Accessed: 2024-04-02.

[25] Y. Xiao, Z. Tian, J. Yu, et al. A review of object detection based on deep learning. *Multimed Tools Appl*, 79:23729–23791, 2020. Accessed: 2024-04-10.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. `https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html`, 2024. Accessed: 2024-04-11.

[27] D. Prokopenko et al. Utilizing the jaccard index to reveal population stratification in sequencing data: A simulation study and an application to the 1000 genomes project. *Bioinformatics (Oxford, England)*, 2024. Accessed: 2024-04-11.

[28] A. Jindia. International journal of innovative research in technology. `https://ijirt.org/`, 2024. Accessed: 2024-04-20.

[29] ScienceDirect Topics. Confusion matrix - an overview. `https://www.sciencedirect.com/topics/engineering/confusion-matrix`, 2024. Accessed: 2024-04-12.

[30] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sept 2009. Accessed: 2024-04-20.

[31] A. Jindia. Object detection: A detailed review study. https://ijcrt.org/papers/IJCRT2112439.pdf, 2021. Accessed: 2024-04-25.

[32] P. Henderson and V. Ferrari. End-to-end training of object class detectors for mean average precision. In *SpringerLink*, 2021. Accessed: 2024-04-25.

[33] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2), April 2022. Accessed: 2024-04-31.

[34] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman. Natural language processing: An introduction. *OUP Academic*, 2011. Accessed: 2024-04-29.

[35] E. D. Liddy. Natural language processing. surface.syr.edu. Accessed: 2024-04-29.

[36] R. Sharnagat. Named entity recognition: A literature survey. https://www.cfilt.iitb.ac.in/resources/surveys/rahul-ner-survey.pdf, 2024. Accessed: 2024-04-25.

[37] A. A. Manjunath et al. Automated invoice data extraction using image processing. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 2024. Accessed: 2024-04-14.

[38] SouravG94. Invoice dataset. https://github.com/SouravG94/invoice-dataset, 2024. Accessed: 2024-03-10.

[39] T. Lin. Labelimg. https://pypi.org/project/labelImg/, 2024. Accessed: 2024-04-15.

[40] H. Yu. Converting pdf to bitmap causes partial data loss in the image - application developer. https://learn.microsoft.com/en-us/troubleshoot/windows/win32/converting-pdf-file-to-bitmap-image, 2024. Accessed: 2024-04-10.

Appendix A

This appendix provides visual and code-based resources related to the thesis. It includes annotated invoice samples, model predictions, and scripts used in the processing and analysis of the data.

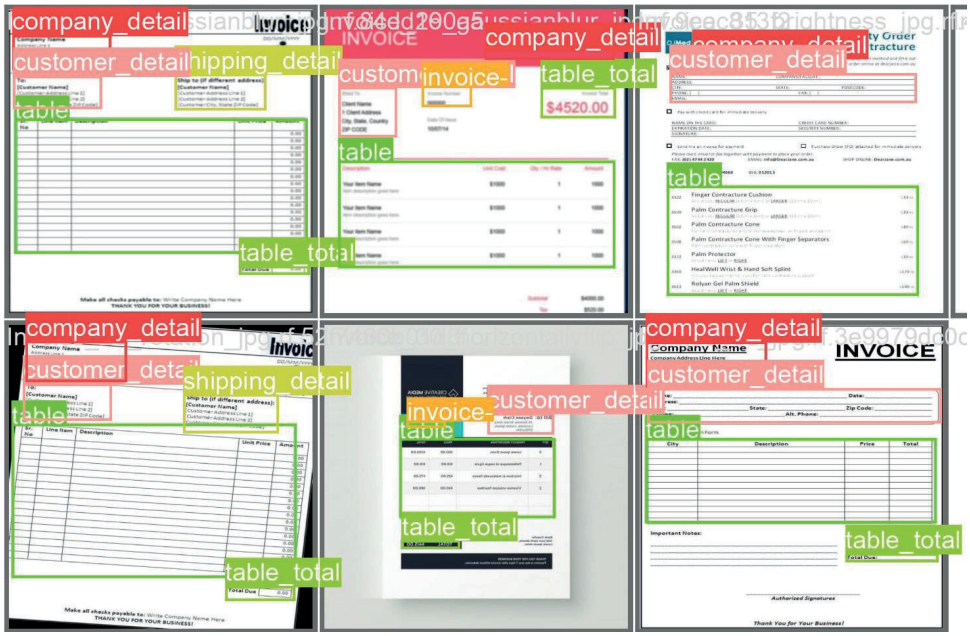

Figure 1: Samples of annotated invoices with their respective labels

Figure 2: The models prediction of the sample showcasing the probability of the labels being detected

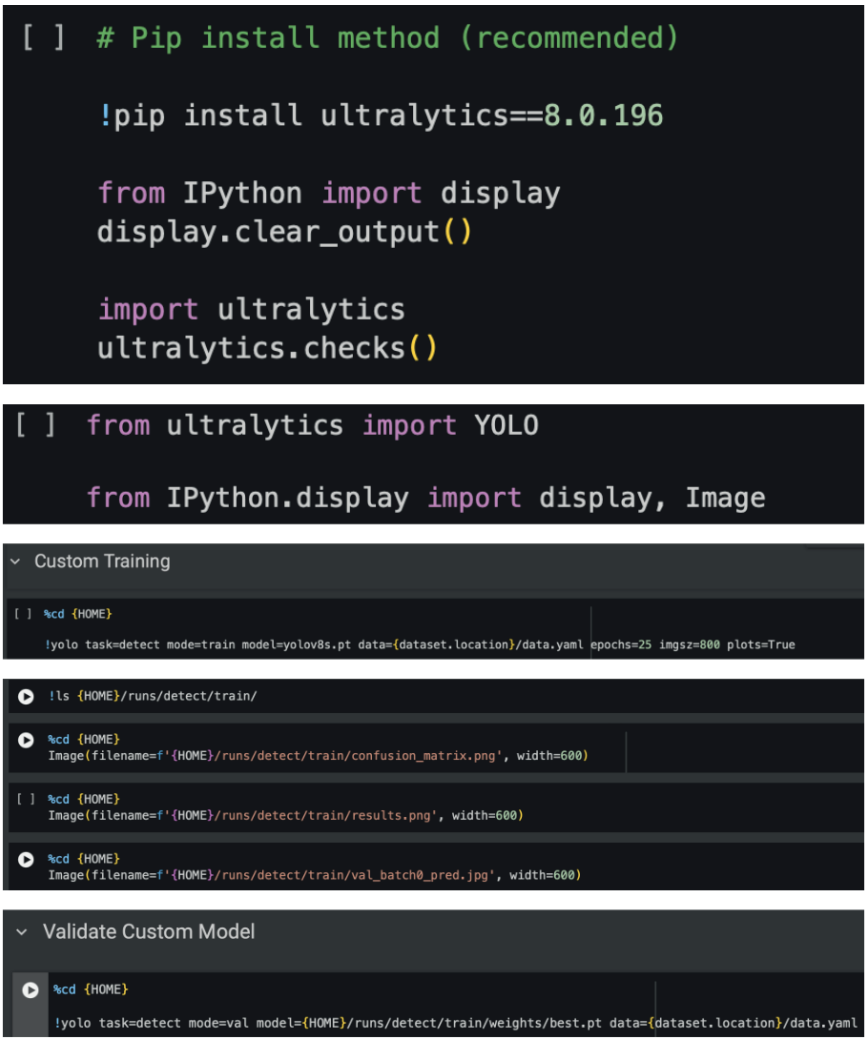

Figure 3: Python notebook to train and upload the model

```python
def perform_object_detection(image_paths, model_path):
    # Load the YOLOv8 model
    model = YOLO(model_path)

    for image_path in image_paths:
        invoice_data = {}

        # Perform inference on an image
        results = model(image_path)

        # Extract bounding boxes, classes, names, and confidences
        boxes = results[0].boxes.xyxy.tolist()
        classes = results[0].boxes.cls.tolist()
        names = results[0].names
        confidences = results[0].boxes.conf.tolist()

        # Iterate through the results
        for box, cls, conf in zip(boxes, classes, confidences):
            x1, y1, x2, y2 = box
            name = names[int(cls)]
            print(name, x1, y1, x2, y2)
            if name == 'table':
                doc = useTableNER(apply_ocr(image_path, x1, y1, x2, y2))
                description = ''
                quantity = ''
                cost = ''
                total_cost = ''
                for ent in doc.ents:
                    if ent.label_ == 'COST':
                        cost = ent.text
                    elif ent.label_ == 'TOTAL_COST':
                        total_cost = ent.text
                    elif ent.label_ == 'QUANTITY':
                        quantity = ent.text
                    else:
                        description = ent.text

            elif name == 'invoice#' or name == 'due_date':
                text = apply_ocr(image_path, x1, y1, x2, y2)
                invoice_data[name] = text #preliminary
            else:
                doc = useNER(apply_ocr(image_path, x1, y1, x2, y2))
                email = ''
                company = ''
                address = ''
                for ent in doc.ents:
                    if ent.label_ == 'EMAIL':
                        email = ent.text
                    elif ent.label_ == 'ADDRESS':
                        address = ent.text
                    else:
                        company ='COMPANY'

                sender = Sender(company, email, address)
                sender_json = sender.jsonText()

                invoice_data[name] = sender_json

            #result = insert_into_database(invoice_data)
            print(apply_ocr(image_path, x1, y1, x2, y2))
            #return result
```

Figure 4: Method for performing object detection

```python
from pdf2image import convert_from_path

2 usages
def convert_pdf_to_images(pdf_path, output_folder, fmt='jpeg'):
    # Convert PDF to images
    images = convert_from_path(pdf_path, output_folder=output_folder, fmt=fmt)
    image_paths = []
    for i, image in enumerate(images):
        # Save pages as images in the pdf
        image_path = f'{output_folder}/page{i}.jpg'
        image.save(image_path, format: 'JPEG')
        image_paths.append(image_path)
    return image_paths
```

Figure 5: Method for converting pdf files into images

```python
def process_documents(folder_path, output_folder, model_path):
    pdf_paths = find_pdfs_in_folder(folder_path)
    if pdf_paths:
        for pdf_path in pdf_paths:
            image_paths = convert_pdf_to_images(pdf_path, output_folder)
            perform_object_detection(image_paths, model_path)
    else:
        image_paths = find_images_in_folder(folder_path)
        if image_paths:
            perform_object_detection(image_paths, model_path)
        else:
            print("No PDFs or supported image files found in the specified folder.")

def main():
    pdf_folder_path = 'PDF'
    images = 'Fakturor'
    model_path = 'best.pt'

    # Process documents (either PDFs or images) in the specified folder

    for image_path in images:
        process_documents(pdf_folder_path, images, model_path)

        # Validate and insert the data into the database

        result = perform_object_detection(image_path, model_path)



if __name__ == "__main__":
    main()
```

Figure 6: Main method for processing

Appendix B

This appendix focuses on the OCR methods and the conversion of annotated data into structured formats. Included are figures depicting the specific methods used.

```python
def apply_ocr(image_path,a,b,c,d):#, roi_coordinates):
    # Read the image
    image = cv2.imread(image_path)

    roi = image[ int(b):int(d),int(a):int(c)]

    rescale = cv2.resize(roi, (500, 500), interpolation=cv2.INTER_LINEAR)

    # Convert the ROI to grayscale
    gray_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

    # Apply thresholding to enhance text
    _, threshold_roi = cv2.threshold(gray_roi, 127, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)


    img = cv2.bitwise_not(threshold_roi)
    kernel = numpy.ones((1,2), numpy.uint8)
    img = cv2.dilate(img, kernel, iterations=1)
    img = cv2.bitwise_not(img)

    text = pytesseract.image_to_string(roi)
    return text
```

Figure 7: Method to apply OCR

```python
import spacy
from spacy.tokens import DocBin
from tqdm import tqdm
import json

#pre-trained english model, used as starting point
nlp = spacy.load("en_core_web_sm")

db = DocBin()

#open the json file with the annotations for the model
f = open('annotations2.json')
TRAIN_DATA = json.load(f)

for text, annot in tqdm(TRAIN_DATA['annotations']):
    doc = nlp.make_doc(text)
    ents = []
    for start, end, label in annot["entities"]:
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("HEJ")
        else:
            ents.append(span)
    doc.ents = ents
    db.add(doc)

db.to_disk("./training_data3.spacy")

```

Figure 8: NER method to turn annotations into spacy objects

```python
import json

class Invoice:
    def __init__(self, dueDate, total, invoiceNbr):
        self.dueDate = dueDate
        self.total = total
        self.invoiceNbr = invoiceNbr

class Invoice_to_json:
    def jsonText(self, Invoice):
        invoice_temp ={ "due date": Invoice.dueDate,
                        "total": Invoice.total,
                        "invoice number": Invoice.invoiceNbr}
        return  json.dumps(invoice_temp, indent=2)

class Invoice_items:
    def __init__(self, description, quantity, price, sum):
        self.description = description
        self.quantity = quantity
        self.price = price
        self.sum = sum

class Invoice_items_to_json:
    def jsonText(self, item):
        item_temp ={"description": item.description,
                        "quantity": item.quantity,
                        "price": item.price,
                        "sum": item.sum}
        return json.dumps(item_temp, indent=2)

class Sender:
    def __init__(self, name, email, adress):
        self.name = name
        self.email = email
        self.adress = adress

class Sender_to_json:
    def jsonText(self, Sender):
        sender_temp ={"name": Sender.name,
                        "email": Sender.email,
                        "adress": Sender.adress,}
        return json.dumps(sender_temp, indent=2)

class Receiver:
    def __init__(self, name, email, adress):
        self.name = name
        self.email = email
        self.adress = adress

class Receiver_to_json:
    def jsonText(self, Receiver):
        receiver_temp ={"name": Receiver.name,
                        "email": Receiver.email,
                        "adress": Receiver.adress,}
        return json.dump(receiver_temp, indent=2)

```

Figure 9: classes to JSON

Printed by Tryckeriet i E-huset, Lund 2024

LUND
UNIVERSITY

Series of Bachelor´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-994
http://www.eit.lth.se