Tribhuvan University
Institute of Science and Technology
Bhaktapur Multiple Campus
Dudhpati, Bhaktapur

A Final Year Internship Report

ON

Automatic Document Processing using Layoutlm and Table Tansformers

AT

Dogma Group Pvt Ltd

Under the Supervision of

Asst. Prof. Surya Bam

Submitted By:

Manoj Kumal(20245/075)

Submitted To:

Institute of Science and Technology

Tribhuvan University

In Partial Fulfillment of the Requirement for

Bachelor's Degree

In

Computer Science and Information Technology

October 2023

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this report has been prepared under my supervision by **Manoj Kumal [20245/075]** entitled **"Automatic Document Processing(OCR) using Layoutlm and Table Tansformers"** in partial fulfillment of this requirement for the degree of B.Sc. In Computer Science and Information Technology (B. Sc. CSIT) be processed for evaluation.

.....

Asst.Prof Surya Bam

Supervisor

Department of Computer Science and Information Technology

Bhaktapur Multiple Campus

Dudhpati, Bhaktapur

LETTER OF APPROVAL

This is to certify that this project **Manoj Kumal [20245/075]** entitled **"Automatic Document Processing(OCR) using Layoutlm and Table Tansformers"** in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT) has been well studied. In our opinion, it is satisfactory in the scope and quality of the required degree.

Supervisor:

Asst.Prof. Surya Bam

Department of Computer Science and Information Technology

Bhaktapur Multiple Campus

HOD:

Mr.Sushant Poudel

Bhaktapur Multiple Campus

Internal Examiner: Bhaktapur Multiple Campus

External Examiner

Tribhuvan University Kirtipur, Nepal

October , 2023

ACKNOWLEDGEMENT

First of all, I would like to express special gratitude to the Institute of Science and Technology, Tribhuvan University for providing the opportunity to explore my interests and ideas in the field of computer science through this internship project as a part of my duty for the fulfillment of the requirement of bachelor's degree of Computer Science and Information Technology.

I would like to appreciate **Mr. Pramesh Gautam, AI Lead of Dogma Group Pvt. Ltd** for his valuable guidance during the intern period. Her encouragements boosted morale for the completion of this intern session. I would like to express my sincere gratitude to our wonderful and supporting mentors **Mr.Baibhav Baidya (Solution Architect)**, and **Mr. Shree Krishna Jamakatel (Sr. AI Engineer)**, for their incredible support and guidelines during the entire internship project execution. I would also like to thank **Ms. Bibhusha Manandhar, HR of Dogma Group Nepal Pvt. Ltd** for explaining the mission, views, and objectives of the company.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of the staff of BSc. CSIT department, who gave permission to use all required equipment and the necessary materials to complete the task. A special thanks go to the constant support from our seniors and every teaching staff of B.Sc.CSIT department helped me successfully to complete this project. Many thanks go to the supervisor, **Asst. Prof. Surya Bam**, who has invested his full effort in guiding me in achieving the goal.

Last but not least, we would like to thank all our teachers, seniors, and friends for their support, irrespective of the situation, and constant inspiration, which helped us withstand our dreams and make our project come to an end.

ABSTRACT

Document processing and analyzing with Optical Character recognition (OCR). OCR is the process of converting a document in any format (pdf, image, html.) to machine readable format. It is a process of extracting textual information from documents. The next step is to make sense of the extracted textual information by classifying the extracted textual information into predefined fields. Document processing and OCR is a rapidly growing field which aims to reduce manual entry work such as creating bills, manually extracting information from Sales, and Invoice orders. The rate of arrival of document in a large firm is rapid, and to extract information from those documents takes a lot of manpower resources and time. The project "Document analysis with OCR" aims to reduce the manual entry work of extracting information from documents, classifying them into certain fields etc. with satisfying accuracy. Also, the system is an AI driven solution, the system is trained continuously with more amount of incoming data periodically. The system is helpful for large firms/companies which receive a lot of text-based documents mainly in images and pdf format and need to extract relevant information from these documents. The project works as Backend service for Microsoft Dynamics 365 and Business central where the users send the documents to a specific email address and the system is triggered which returns the relevant results after extracting text from the document, analyzing and classifying them. Users would not need to do manually entry of typing each information on their own. Once the results are updated on user's end, the task of user is to validate the information if it is syntactically and semantically correct and rectify the mistakes if any. Users can view the documents, curate the bounding box where the relevant information is present, delete or add other information if needed. All these user interactions are recorded and used to improve the system in future. The current project scope is to handle the Purchase orders only which are in PDF or Image format

Keywords: OCR, Document Analysis, Document processing, PDF, Image, Purchase Orders.

Table of Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

APA	American Psychological Association
API	Application Programming Interface
BC	Business Central
HR	Human Resources
IT	Information Technology
JS	JavaScript.
JWT	JSON Web Token
KSS	Knowledge Sharing Session
OCR	Optical character Recognition
QA	Quality Assurance
SDLC	Software Development Life Cycle

Chapter 1

Introduction

1.1 Introduction

The Document Analysis with OCR project, developed by Dogma Group, represents a transformative leap in document processing automation, with a primary focus on streamlining the handling of purchase orders. In today's fast-paced business environment, the efficient processing of documents is essential to maintaining competitiveness and agility. This project addresses this need by leveraging cutting-edge Optical Character Recognition (OCR) technology to seamlessly convert PDFs and images into actionable data. Traditional document processing methods often involve labor-intensive manual tasks, prone to errors and delays. Dogma Group's system offers an innovative solution that not only eliminates these bottlenecks but also enhances overall productivity and accuracy. By receiving purchase orders in various formats, such as PDFs or images, the system applies OCR algorithms to extract textual information. It then employs advanced data processing techniques to identify and categorize relevant information.

The heart of this system lies in its ability to classify documents, thereby ensuring that each piece of information reaches the right destination. Documents are seamlessly integrated into Microsoft Business Central, where users can access a range of powerful functionalities. These functionalities extend beyond mere viewing and include validation, error rectification, and the novel capability to draw bounding boxes for the extraction of specific information. The Document Analysis with OCR system possesses an impressive spectrum of information extraction capabilities. It can discern critical details such as Invoice Numbers, Invoice Dates, Posting Dates, Due Dates, Currency Codes, and Payment Terms. Additionally, it excels at mining information from tables within documents, extracting vital data points like Description, Quantity, Unit Cost, Vat Percent, and Vat Amount. These capabilities adapt to the specific content of each document, ensuring a dynamic and precise extraction process. The Document Analysis with OCR project, exploring its technical intricacies, business implications, and the profound advantages it offers to organizations. By revolutionizing document processing, this project exemplifies Dogma Group's commitment to driving automation, efficiency, and accuracy in modern business operations.

1.2 Problem Statement

Most businesses still manually process large volumes of documents without any form of digitization. This results in higher costs, and complexity with storing documents, consuming more time while handling sensitive/confidential clients' data with no security and privacy. The process of manually processing documents results in the need for ever growing skilled human resources, the size human resources needed according to the rate of flow of documents. Since every business nowadays is moving toward digital space, processing those digital documents is still a traditional method. The manual processing method decreases a lot of efficiency and speed. Also, with changing the format of documents, it is much harder to process by humans and takes more time.

1.3 Objective

Before setting up all the parameters for the internship there should be a proper vision about what might be the goal or objectives of the internship and project. The following is the list of some of the objectives that have been reported in discussing the internship and project.

- To be able to analyze the requirements, prepare, model, and architecture of the software system.
- To be able to develop the application following agile practices.

1.4 Scope and Limitation of the Project

Scope

The main purpose of this project is to create a backend application which receives the document in any format and classifies the relevant fields which is sent back to the user to Microsoft Business Central. Some of the scopes of this project are:

- Process any kind of document format (PDF or PNG or JPG).
- Process only the purchase orders that is in English language.
- Create a feedback loop to continuously improve the system as it gets more data
- Seamless integration with Microsoft Business Central
- Proper validation methods incorporated.

Limitation

Some limitations of this project are:

- One needs to have an internet connection to access this system.
- The system only works for Purchase orders.
- The system only supports English language documents.
- Higher error rate in the case of table extraction and recognition

1.5 Report Organization

The contents of the report are organized into the following sections Chapter 1 discusses the introduction to the project with the objectives to be met. Also, the scope and limitations of the project are discussed. Chapter 2 discusses the background study of the organization and literature review. Organization introduction, hierarchical structure, working domains and department units are discussed. Studies of various systems and architecture are reviewed in this section through which the foundation of the project was set up. In Chapter 3, all the internship activities are described here. It includes roles and responsibilities during internship, weekly logs maintained during internship, description of the project involved

during internship and activities performed in it. In Chapter 4, the conclusion of the report and learning outcomes of the internship are discussed here. In the end, all the references are mentioned in the API format.

Chapter 2

Organization Detail And Literature Review

2.1 Introduction to Organization

2.1.1 Organizational Background

Dogma Group Pvt. Ltd was established in Nepal in 2016 as a IT and consultancy company.It provides complete IT solutions, data management, CRM and ERP implementation. Also,it provides, website, app development, and 24*7 support for any kind of assistance. The company is a Microsoft Gold Partner which means, dogma heavily depends upon Microsoft products and cloud adoption framework.

2.1.2 Services Provided by the Organization

Dogma Group Pvt. Ltd. provides multiple services in the area of data management, CRM management and development. The areas of expertise of the organization are:

1. Customer relationship Management
    a. Microsoft Dynamics 365
    b. Microsoft Business Central
2. Website and Software development
    a. React, Node, Mongodb, SQL
    b. Artificial Intelligence
    c. Power automate, PowerApps
3. IT Consulting
    a. CRM consultancy
    b. Data storage, management and transfer
    c. Software Consulting

2.2 Organizational Hierarchy

Dogma Group Pvt. Ltd. comprises an administrative team along with, junior and senior programmers and developers, and HR team.

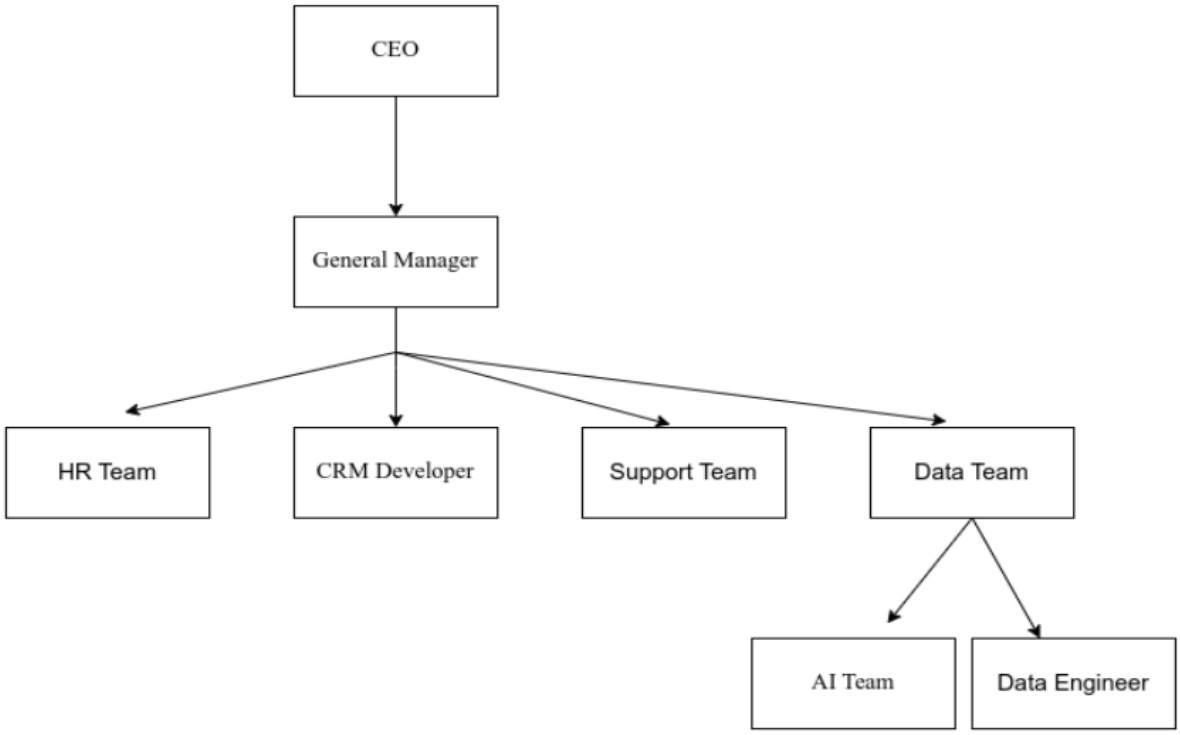

Figure 2.1: Organization Hierarchy

2.3 Working Domains of Organization

Dogma Group Nepal was established as a software development and consultancy company with headquarter located in United Kingdom. It provides complete Customer relation management using Microsoft CRM, ERP solutions, data management and migration. The company has 3 verticals: Strategic, Implementation and Support. The company is mainly focused on Business Process Mapping, Customer Journey mapping, CRM and ERP implementation, Data Migration, Website Portal and app development, providing training and services. The company is a gold Microsoft partner which means the company full leverages the Microsoft services ranging from CRM, ERP solutions to using cloud services, database

services. With highly expert Microsoft professionals, the company aims to provide clients with high quality services as needed by customizing the available services or building new services as needed

2.4 Description of Intern Department

Dogma Group has its own hierarchy, for efficient operation and technical management, and overall working the organization with higher management residing in both Nepal and UK. Dogma Group has a separate vertical "Dynamic AI" which mainly focuses on providing AI driven solutions working based on client requirement. The "Dynamic AI" is a separate vertical with its own structure, management, and projects. The resources are assigned according to project requirements and expertise. Since software development is a closed loop process, the AI team works very closely with other teams of developers, CRM developers, clients, QA and sales department and a project manager. A supervisor is assigned to look after the projects and tasks assigned, track progress, provide guidance and suggestions as needed. A daily log is maintained which outlines all the tasks status of each day which is presented in daily scrum meeting, and also a weekly update call is set up separately to efficient task the process, the hurdles and for overall career growth.

2.5 Literature Review

2.5.1 Background and Contextual Research

Most of the business facing organizations receive huge number of invoices for the purchase they have made or the items/services they sale or receive. The rate at which this document generates depends upon the size of organization, and policy. All these documents contain vital information regarding invoice number, the invoice date, items descriptions, quaintly, amount etc. The information from these documents is then transferred to some central repository system such as an SQL database or in our case into an ERP solution. The process by which the information is extracted from documents is a manual task, specific people are assigned to manually type out the information into some central repository system and carry out further processes such as processing those invoices or sending out emails regarding payment which depends upon organization. An organized and automated way to

process those documents is essential for every organization. These documents are generated every time, if any purchase is made by an organization or by any employee through an organization, when an organization must pay some subsidiary fees in exchange for some goods or services etc. It is much more beneficial for organizations that mostly deals with import-export business. There are mainly modules important steps in automated document processing. The OCR module can extract text from 7 any kind of document with less error. The OCR system is built with text and recognition algorithms (Wei, Sheikh, & Ab Rahman, 2018) and (Rajavelu, Musavi, & Shirvaikar, 1989). The second module, document understanding, is to find the relevant information from all the extracted text. All the extracted information might not be useful, it is important to filter out only the relevant information and classify it to specific fields as needed. Next modules store all the data that is extracted after validation from the user, the storage facility could be some database server, a CRM or ERP solutions as on requirement basis. The main objective is to store all information in one centralized system so that in future we can query any information as needed and also use the collected data to build a more robust system. In this case, mostly purchase invoices are processed which contain details regarding vendors, total paying amount, due date, items received, quantities, vat rate etc using Layoutlm (Xu et al., 2020) and (Huang, Lv, Cui, Lu, & Wei, 2022). Once this information is extracted it is then sent to clients for further processing (Li et al., 2021) and also to a persistent storage system which is Microsoft Dynamics 365 in this case. The document processing system is tightly integrated with Dynamics 365 on how the input is received, what kind of security mechanisms to be followed, and how to view the results by client and proper validation methods since the documents are mostly financial documents.

2.5.2 Related Works

1. **Shipmax**

   Shipmax is a company that helps logistics companies automate their back-office processes by is automatically extracting the relevant information from documents. It provides services to extract information from any unstructured documents without the need for any kind of templates to prepare beforehand. There are multiple services provided by Shipmax (Shipmax , n.d.). such as, it extracts, structure and clean

invoice data in real-time without manual data entry, BPO, or vanilla OCR.

2. **Docsumo**

   Docsumo is a Nepal based company mainly focused on document intelligent which helps convert unstructured documents such as pay stubs, invoices and bank statements to actionable data. It works with documents in any format with minimal setup. It provides customized services as required with minimal setup and can extract any kind of information within the document. The system provides API integration to any third-party sites if needed.

   Its features are:

   i. **Able to work with multiple types of documents**

      Works for different variety of documents such as Bank statements, Invoices, Bills, etc.

   ii. **Customization**

      The system is highly customizable according to need. We can add or delete what information is needed or not and train the AI driven system on our own if needed with more annotated data.

   iii. **Third party Integration**

      It provides API which is independent of any third-party sites which makes it as to integrate easily with any third-party services.

3. **Optical character recognition**

   OCR technology is the process to extract all the textual information from the document. There are two types of documents: Digital born Vs Scanned documents. Optical character recognition in digital born documents is easy using pdf rendering tools and accuracy is very high but in case of scanned documents which contains images embedded, the OCR is built with combination of two different modules: Text detection; to detect and isolate each character and Text recognition to recognize the text and output the word and its corresponding bounding box. It depends upon the quality of image, quality of the text. Dependence on multiple factors causes the OCR engine to perform less accurately in the case of scanned documents. Tesseract (Tesseract

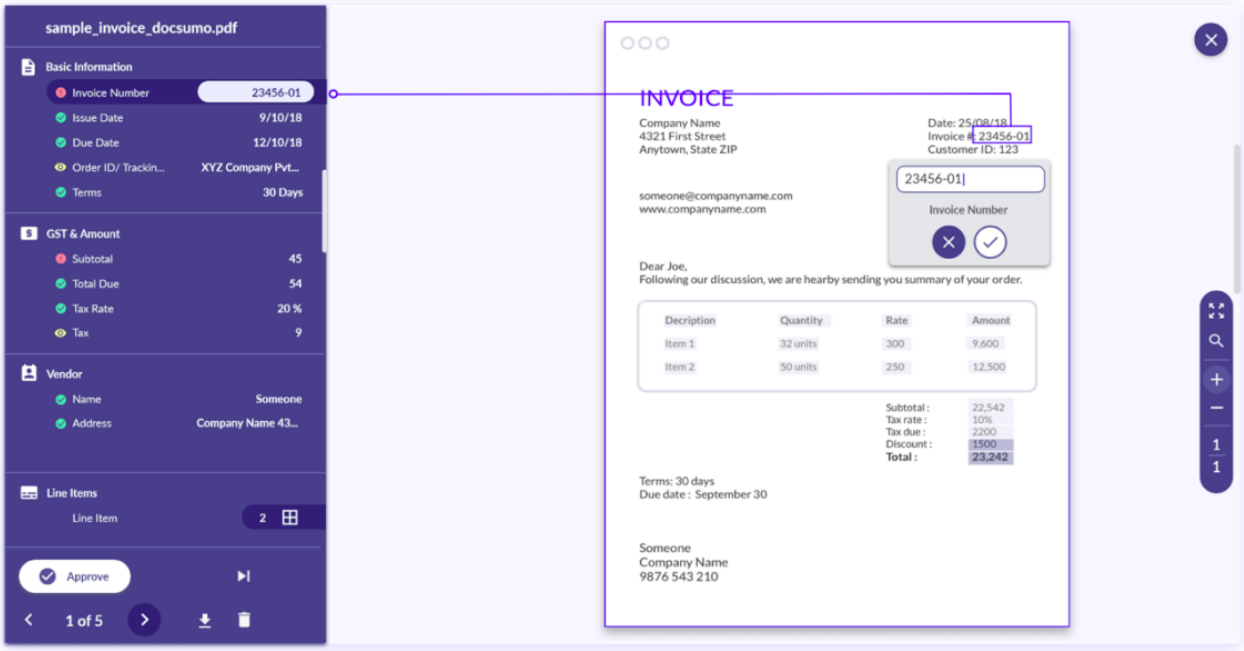

Figure 2.2: Docsumo (Dashboard Docsumo)

OCR Engine, n.d.). is one of the most famous and widely used OCR engines which is used for multiple languages.

4. **Token Classification**

   Once the OCR process is completed which gives all words and their bounding box coordinates. The next step is to filter out this information to only what is needed. To filter out the information, the obtained text should be classified into some classes such as if it is Invoice number or Invoice Date or Due Date etc. There have been many proposed techniques to classify this textual information, and it is still an active field of research. Some notable techniques are using templates I.e., template matching system according to bounding box co-ordinates, using Natural language processing techniques such as Named entity recognition. Also, recently more advanced machine learning and deep learning methodologies have been devised such as LayoutLM. which leverages the concept of computer vision and natural language processing.

Chapter 3

Internship Activities

3.1 Roles and Responsibilities

As an AI Engineer Internship Internee responsibility was to perform requirement analysis, feasibility study, and research on probable OCR engines, token classification problems, and integrating the whole system with Microsoft Dynamics 365. For this, Internee required knowledge of Python, Machine Learning, Deep learning, Natural Language processing, cloud services such as Azure, FastAPI, database, and deployment tools. Internee learned about software engineering, API design and documentation, parallel and multiprocessing for faster inference Here's an overview of my roles and tasks:

a. **Requirement analysis and Feasibility study:**

   i. Requirement analysis based on client's requirements and preparing feasibility study document for the project along with mapping the timelines.

b. **OCR Engine:**

   i. Research on appropriate OCR engine suitable for multiple different types of documents

   ii. Analyze the results of OCR and present the results and rectify mistakes as much as possible.

c. **Token Classification and Document analysis:**

   i. Train deep learning models based on client's data for token classification.

   ii. Setup data preparation, annotation, machine learning training and monitoring pipelines.

d. **Backend Development and Deployment:**

   i. Developing API's to be consumed by Microsoft Dynamics 365

   ii. Deploy all the microservices on Azure App Service.

3.2 Weekly logs

The weekly activity log during the internship period is given as follows:

Table 3.1: Week 1 Summary Detail

Date	Day	Task Completed
03/05/2023	Mon	Team and HR introduction.
04/05/2023	Tues	Getting Familiar with other teams and introduction.
05/05/2023	Wed	Session on internal office tools and technology stacks.
06/05/2023	Thu	Introduction to problem domain along with client requirements.
07/05/2023	Fri	Attended client call and started requirement analysis process.

Table 3.2: Week 2 Summary Details

Date	Day	Task Completed
10/05/2023	Mon	Dev tools - Azure Devops, Azure Introduction to Dynamics 365 and ERP solutions, Access to Azure cloud and development tools.
11/05/2023	Tues	Requirement analysis process: Gather requirements from client. Research on other available solutions Research on available data and deadlines.
12/05/2023	Wed	Requirement analysis process: Discussion with Internal cross functional teams (Dynamics 365, Developers.) Map the features to be included along with pros and cons. Map the timeline for each process along with dependency on other cross functional teams.

Date	Day	Task Completed
13/05/2023	Thu	Feasibility Study Report: Study the probable solutions, pitfalls, features that can be included in version1 or not.
17/05/2023	Fri	Computer Vision and Image processing: Started to learn about Image processing: (Binarization, Thresholding, Image cropping, dilation etc.)

Table 3.3: Week 3 Summary Detail

DATE	DAY	TASK COMPLETED
18/05/2023	Mon	Computer Vision and Image processing: Implement Image processing techniques on sample data using OpenCV and SkImage. Compare different algorithms, their functions and results.
19/05/2023	Tues	Word segmentation and recognition: Word segmentation using contour matching using OpenCV findContours() Played with different Kernel sizes and types Analyze the results on multiple documents to check accuracy and performance metrics.
20/05/2023	Wed	Research on state-of-the-art OCR methods and techniques.
21/05/2023	Thu	Research on state-of-the-art OCR methods and techniques. Tesseract by google traditional + Machine learning approach. High-level python wrapper (py-tesseract) is available. Run py-tesseract on multiple images, and analyze the results.
22/05/2023	Fri	Research on state-of-the-art OCR methods and techniques. Tesseract results affected by light and low-quality image. Tried deep learning methods PaddleOCR, DocTR

Table 3.4: Week 4 Summary Detail

DATE	DAY	TASK COMPLETED
25/05/2023	Mon	Implement an OCR system in Python: File handling (Image, pdf, word file) OCR engine implementation (for multiple pages pdf) Save result as JSON Implemented multiprocessing with Python Pool for faster inference.
26/05/2023	Tues	Create a Backend API using Python (FastAPI) JWT authentication Restful API Able to read from both PDF/Images.
27/05/2023	Wed	Containerization Containerized the whole application using Docker. Docker volumes to persistently save the data.
28/05/2023	Thu	Research on Token classification Explore Named entity recognition techniques. Train NER with Python, Spacy and NLTK.
29/05/2023	Fri	Template Based System Create rules for each vendor with Regular expression matching. Bounding Box co-ordinate matching.

Table 3.5: Week 5 Summary Detail

DATE	DAY	TASK COMPLETED
04/06/2023	Mon	Template Based System Unit testing Git pre-commit hooks. Code optimization and merge smaller bounding box into one. Code refactoring and peer review.
05/06/2023	Tues	Template Based System Analyze the results on test-set. Best performance if only template found.
06/06/2023	Wed	Team discussions regarding next steps. Template Based Systems results are not up to the mark based on test samples. Research on Machine learning based approach.
07/06/2023	Thu	Research on Named entity recognition: Tried NER from Spacy, NLTK Very domain specific and needs many annotated Data samples.
08/06/2023	Fri	Conducted Knowledge sharing session on containerization.

Table 3.6: Week 6 Summary Detail

DATE	DAY	TASK COMPLETED
12/06/2023	Mon	Research on other deep learning approaches: LayoutLM model for Token classification. Mix of both NLP and computer vision.
13/06/2023	Tues	LayoutLM finetuning: Download Pre-trained models which is trained on millions of documents. Collect training dataset.
14/06/2023	Wed	LayoutLM finetuning: Prepare dataset according to LayoutLM format. Data loader Pytorch script.
15/06/2023	Thu	Label studio for annotation: Manual Annotation of Bounding box co-ordinate and each field.
16/06/2023	Fri	LayoutLM finetuning: Test script on Pytorch. Hyperparameter tuning using Pytorch.

Table 3.7: Week 7 Summary Detail

DATE	DAY	TASK COMPLETED
19/06/2023	Mon	LayoutLM finetuning: Data loader Pytorch script, Training script on Pytorch, GPU for faster training.
20/06/2023	Tues	LayoutLm finetuning: Training in google colab and Azure VM Logging in MLFlow.
21/06/2023	Wed	LayoutLm finetuning: Use multiple versions of LayoutLM.
22/06/2023	Thu	Inference Module: Inference module in Python OCR Engines to Token Classification to JSON.
23/06/2023	Fri	Test with multiple OCR Engine.

Table 3.8: Week 8 Summary Detail

DATE	DAY	TASK COMPLETED
26/06/2023	Mon	Added spelling correction methods.
27/06/2023	Tues	Spelling correction: Implement Levinstein distance.
28/06/2023	Wed	Combine close bounding Boxes by computing Bbox areas.
29/06/2023	Thu	Mapping Fields according to Microsoft Business Central format.
3/07/2023	Fri	Knowledge sharing session on OCR and Token classification.

Table 3.9: Week 9 Summary Detail

DATE	DAY	TASK COMPLETED
4/07/2023	Mon	[API] for OCR system with different OCR Engine.
5/07/2023	Tues	[API] for Token classification using layoutLM.
6/07/2023	Wed	[API] for communicating with Power Automate and validating results from Business Central.
7/07/2023	Thu	[API] for adding more fields when required by Client.
8/07/2023	Fri	[API] for communicating in case of erroneous results.

Table 3.10: Week 10 Summary Detail

DATE	DAY	TASK COMPLETED
11/07/2023	Mon	Created different microservices using FastAPI. JWT authorization. Login Authorization with Microsoft Graph API.
12/07/2023	Tues	Containerization of microservices. Docker for containerization. Manual Testing and Unit testing.
13/07/2023	Wed	Used Docker Volumes for persistent data storage. Fix issues on docker (Shared memory, Network issues).
14/07/2023	Thu	Leave.
15/07/2023	Fri	Leave.

Table 3.11: Week 11 Summary Detail

DATE	DAY	TASK COMPLETED
16/07/2023	Mon	Explored Azure services for hosting the API.
17/07/2023	Tues	Host on Azure App Service: Store containerized images on Azure container registry using AzureCLI.
18/07/2023	Wed	Azure App Service: Create Azure app service instance. Setup load testing (Manual and with QA team).
19/07/2023	Thu	Azure App Service: Setup CI/CD pipeline with Azure Devops. Store artifacts on Azure Container registry. Setup logging and monitoring mechanism.
20/07/2023	Fri	Deploy version1 on Azure App service: Manual Testing. Monitoring logs and performance issues.

Table 3.12: Week 12 Summary Detail

DATE	DAY	TASK COMPLETED
23/07/2023	Mon	Research on Table extraction methods. Paper review and State of the art techniques.
24/07/2023	Tues	Table Extraction. Image denoising with OpenCV. Invert an image using OpenCV bitwiseNOT operator. Vertical and Horizontal line detection using OpenCV erode(), dilate() and morphological operations.
25/07/2023	Wed	Table Extraction. Combine Vertical and Horizontal lines to detect table cells. Contour detection using OpenCV findContours()
26/07/2023	Thu	Table Extraction. Merge close contours and sort Contours. Run OCR engine on detected contours.
27/07/2023	Fri	Recognizing exact tables needed in case of multiple tables: Using NER techniques. Word similarity with Word embeddings.

3.3 Description of the Project Involved During Internship

3.3.1 Placement

The internship period was three months where Internee gained experience in AI and Machine learning. The interview was taken by the team AI lead and CEO.

3.3.2 Duration

The standard internship period fixed by the Tribhuvan University is six credit hours, which is equivalent to eight weeks or two months. However, internship period was three months.

Office Hour: 9am – 5:30pm

Start Week: 3rd May 2022

Ending Week: 4th Aug 2023

3.3.3 Working Environment

The standard internship period fixed by the Tribhuvan University is six credit hours, which is equivalent to eight weeks or two months. However, internship period was three months.

1. **Programming Language**
   The project majorly used Python as programming language.

2. **Backend**
   The project used FastAPI for creating backend API's.

3. **QA**
   Manual testing was performed during the project.

4. **Time division and management**
   Azure was used for task division and time management.

5. **Deployment**
   Azure App services and Azure container registry were used for deployment of microservices.

6. **Communication**

   Microsoft teams was used for communication.

7. **Development tools**

   Azure Devops was used for code repository, version management, task assignment and management.

3.3.4 Module of the project

1. Optical Character recognition
2. Token Classification
3. Table Extraction

3.3.5 Module Description

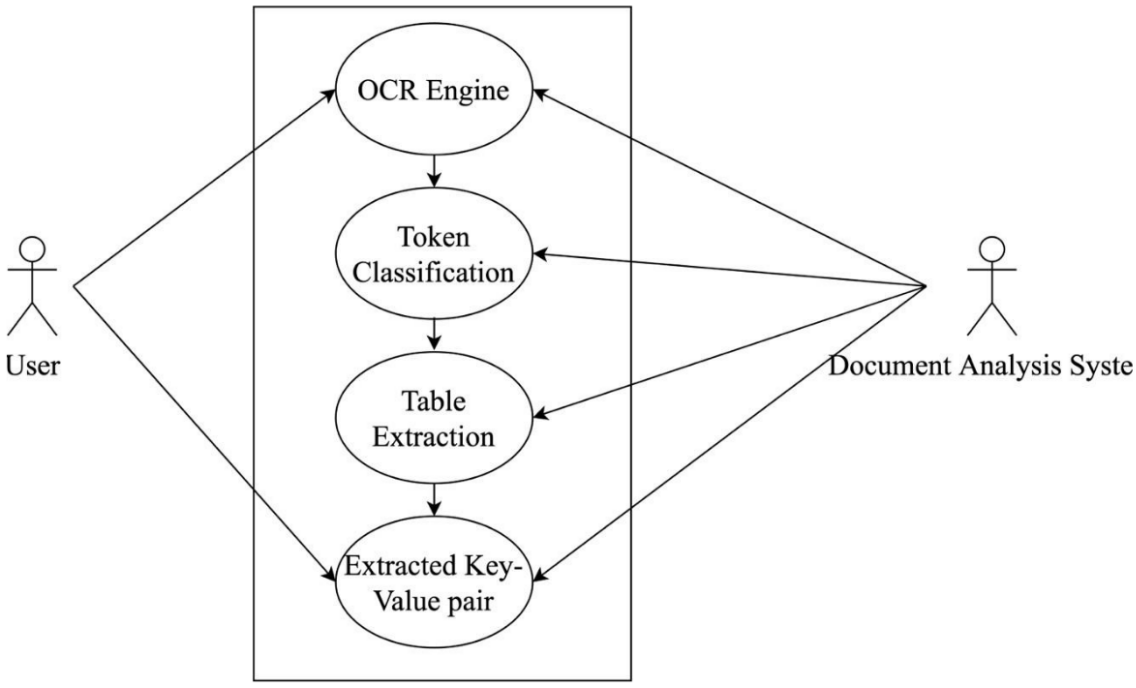

Figure 3.1: Use Case Diagram for Document Analysis

1. **Optical Character recognition**

   Document analysis is equipped with Optical character recognition feature which is used to convert any document, pdf, images to machine readable format. The Optical character Recognition feature enables to extract text (each word) and its corresponding bounding boxes. The optical recognition feature is optimized to work on documents with variable lightening, handwriting, and color combination.

2. **Token Classification**

   The Token classification feature of Document analysis by using the combination of Computer vision and Natural language processing techniques to its respective class which is finetunes for specific types of invoices. There are different token classification models, each for Purchase Invoices, Sales invoices, etc. For example, in the case of Purchase Invoices, the token can be classified into Invoice name, Invoice Date, Due Date etc. on which model is trained on.

3. **Table Extraction**

   The project has a feature to extract the fields/cells from the document if a table is present. The table extraction module is optimized to extract each row and columns, perform OCR on the extracted cells, and out a csv or excel as required. It helps in capturing purchase lines and sales lines from the document.Table extraction was completed by using Table transformers (Smock, Pesala, & Abraham, 2022). In table extraction, there is two steps involved one is table detection and table structure recognition (Smock, Pesala, & Abraham, 2023b) and (Smock, Pesala, & Abraham, 2023a).

3.3.6 Class Diagram

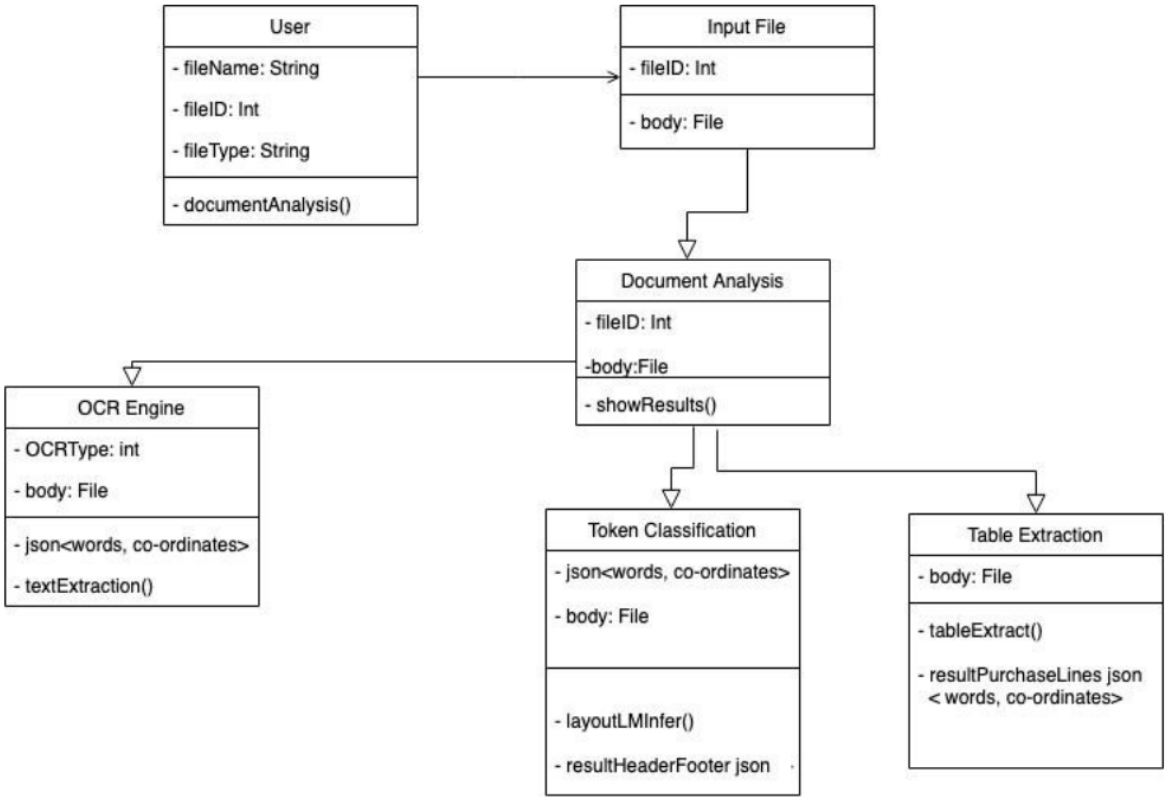

Figure 3.2: Class Diagram for Document Analysis

3.3.7 Testing

3.3.7.1 Test Cases for Unit Testing

Test Cases	Expected Results	Obtains Results	Pass/Fail
OCR Module	Should process document	Successfully process document	Pass.
Token Classification Module	Should classify token	Successfully classify tokens	Pass.
Table Extraction Module	Should extract table information	Successfully extracted table	Pass.

3.4 Tasks/Activities Performed

The 'Document Analysis' by Dogma Group is an automatic document automation project to extract information from documents and autofill necessary details.

3.4.1 My contributions

1. Document Pre-processing

    i. Document Analysis

    This is the first step in document processing. The author convert dataset into a pandas dataframe, having two columns image path and label name. The dataframe is read as a huggingface dataset object. The feature of image such as image, input ids, attention mask, token type ids, bounding box and label are extracted into a encoded input. Then author define model and initialize it with the weights of the pretrained based model LayoutLM Model. Then author trained the model with adam optimizer and weight decay is fix. Once the model is ready, the document is pass into a model and it will classify the types of documents.

    ```
    def dataset_into_dataframe(dataset_path)
    ```

```
    labels = [label for label in os.listdir(dataset_path)]
    id2label = {v: k for v, k in enumerate(labels)}
    label2id = {k: v for v, k in enumerate(labels)}
    images = []
    labels = []
    for label_folder, _, file_names in os.walk(dataset_path):
    if label_folder != dataset_path:
        label = label_folder[40:]
        for _, _, image_names in os.walk(label_folder):
        relative_image_names = []
        for image_file in image_names:
            relative_image_names.append(dataset_path + "/" +
            label + "/" + image_file)
        images.extend(relative_image_names)
        labels.extend([label] * len (relative_image_names))

    data = pd.DataFrame.from_dict({'image_path': images,
    'label': labels})
    return data

def preprocess_data(examples):
  images = [Image.open(path).convert("RGB") for path
  in examples['image_path']]
  encoded_inputs = processor(images,
  padding="max_length", truncation=True)
  encoded_inputs["labels"] = [label2id[label]
  for label in examples["label"]]
  return encoded_inputs

def encode_dataset(preprocess_data):
    features = Features({
    'image': Array3D(dtype="int64", shape=(3, 224, 224)),
```

```
    'input_ids': Sequence(feature=Value(dtype='int64')),
    'attention_mask': Sequence(Value(dtype='int64')),
    'token_type_ids': Sequence(Value(dtype='int64')),
    'bbox': Array2D(dtype="int64", shape=(512, 4)),
    'labels': ClassLabel(num_classes=len(labels), names=labels)
    })
    encoded_dataset = dataset.map(preprocess_data,
    remove_columns=dataset.column_names, features=features,
    batched=True, batch_size=2)
    return encoded_dataset
```

ii. PDF to image Conversion

when system gets input that could be in pdf or image format etc. The system can handle both types by using conversion function. If the input file is in image format then it can process directly and pass into the system. Otherwise, first pdf document is converted into image format. Then converted document is passed into the next steps. The author have written pdf to image conversion code by importing fitz. To use fitz, the author install PyMuPdf Library. The author also maintain the standard resolution while converting into image format.

```
def pdf_to_image(pdf_file, image_name, dpi):
    doc = fitz.open(pdf_file)
    pix = doc[0].get_pixmap(dpi=dpi)
    pix.pil_save(image_name)
    return image_name
```

iii. Image pre-processing

Now, documents are in image format and documents are send into pre-processing function.Image pre-processing function was wrote using filter2D which is provided by OpenCV Library. Where first image is converted into a grayscale image. The author create a convolution between the image and the given kernel for creating filters for sharpening the image. The author used depth of kernel is

-1 for filter2D.This function will simply convolute the 2d matrix with the image at pixel level and produce an output image.

```
def sharpen_image(image):
    sharpening_kernel = np.array([[-1, -1, -1],
                                  [-1,  9, -1],
                                  [-1, -1, -1]])
    sharpened_image = cv2.filter2D(image, -1,
    sharpening_kernel)
    return sharpened_image
```

iv. Document Skew Correction

The pre-process document is passed into a skew correction function. The first image rotation is calcuated by using tesseract. Then image skew was corrected according to the rotation of the image by using rotate function which is provided by OpenCV Library. If the image is skew then OCR could not able to extract information from the document so that author passes the document into skew correction function and it will correct the skew of the document then only send into the next steps, which is a OCR Engine.

```
def get_angle(pil_image):
    gray_image = cv2.cvtColor(np.array(pil_image),
    cv2.COLOR_BGR2GRAY)
    gray_image = sharpen_image(gray_image)
    try:
        data = pytesseract.image_to_osd(gray_image,
        output_type="dict")
        return data["orientation"], data["rotate"]
    except pytesseract.pytesseract.TesseractError:
        return 0, 0

def fix_rotated_image(pil_img):
    rotated_angle, to_rotate = get_angle(pil_img)
```

```
    rotated = False
    if to_rotate in angle_map.keys():
        new_img = cv2.rotate(np.array(pil_img),
        angle_map[to_rotate])
        pil_img = Image.fromarray(new_img)
        rotated = True

    return pil_img, rotated
```

2. OCR Engine

   i. DocTR OCR

   Once all pre-processing steps will completed then document or image is pass into OCR Engine. The author used DocTR OCR Engine for information extraction from the documents. DocTR OCR will give bounding box of the each words present in the documents. Bounding box contains four coordinates such as Xmin, Ymin , Xmax and Ymax. The Xmin is left top coordinate and Ymin is right top coordinate. Simillary, Xmax is the bottom left coordinate and Ymax is the bottom right coordinate. The author used pretrained model for OCR Engine. In OCR Engine, there was two step first one text detection and text recognition. Both text detection and recogniton model are pretrained model. For Text detection pretrained model is db-resnet50 (Liao, Wan, Yao, Chen, & Bai, 2020) and Text recognition model is crnn-vgg16 (Shi, Bai, & Yao, 2016). This function is responsible for generating bounding box of each words.

```
class Doctr(OCRService):
    def __init__(self, detection_model, recognition_model):
        self.detection_model = detection_model
        self.recognition_model = recognition_model

    def load(self):
        model = ocr_predictor(det_arch=self.detection_model,
        reco_arch= self.recognition_model, pretrained=True,
```

```
        preserve_aspect_ratio=True)
        self.model = model

    def ocr(self, image: str) -> list:
        try:
            doc = DocumentFile.from_images(image)
            res = self.model(doc).export()
            total_words = sum([len(res["pages"][0]["blocks"]
            [x]["lines"]) for x in
            range(len(res["pages"][0] ["blocks"]))])
            rotated = False

            if total_words > 20:
                return res, image, rotated
            else:
                try:
                    fixed_image, rotated = fix_rotated_image
                    (Image.open(image))
                    if rotated:
                        image = image.replace(".png",
                        "_rotation_fixed.png")
                        fixed_image.save(image)
                        doc = DocumentFile.from_images(image)
                        res = self.model(doc).export()
                        return res, image, rotated
                    else:
                        return res, image, rotated
                except Exception as ex1:
                    import traceback
                    logger.error("error run doctr Engine", ex1)
            return res
        except Exception as ex1:
```

```
logger.error("error run doctr Engine", ex1)
```

3. Token Classification

   i. Data Preparation

   The author collect data or document types according to requirements. This data should include information about token classifications for both text and layout elements. The is annotated by using a Label studio which is free opensource tool for annotation. The author annotate only specific fields such as vendor name, vendor invoice number, vendor registration number, company name etc. The Label Studio will give annotated json which contains the field text and coordinates. The Label studio json file and OCR json file will compare and finally data will be ready for training.

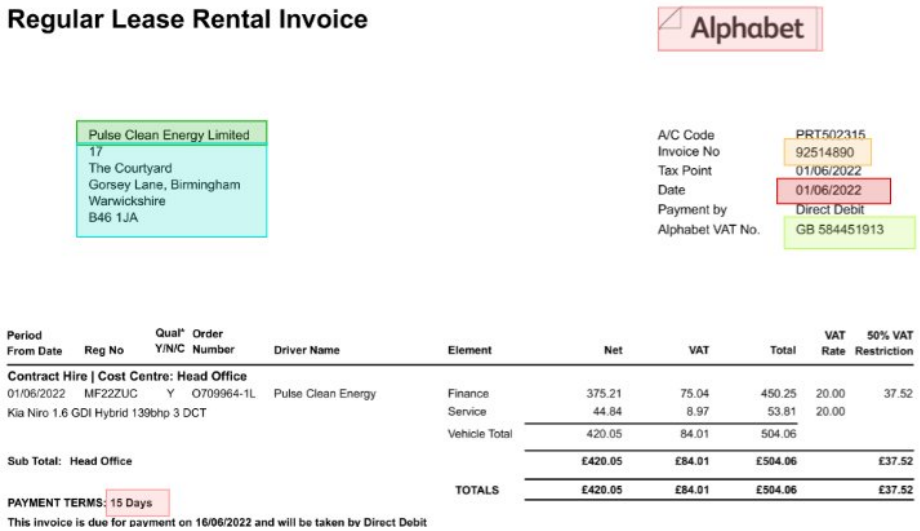

Figure 3.3: Sample Annotated document for token classifiction

   ii. Model Training

   The author uses the pre-trained LayoutLM model and fine-tune on own annotated dataset. The pretrained LayoutLMTokenizer is used as tokenizer. Then author prepared training data loader and test dataloader which will provide batch data for training. The token classification techniques during fine-tuning to enhance the model's understanding of both text and layout information.The model training parameters are learning rate, number of epoch, optimizer etc.

The Adam as a optimizer is used, learning rate is 0.001 and epoch is 10. The author adjust hyperparameters such as learning rate and batch size based on the size and characteristics of the dataset. Fine-tune hyperparameters to achieve optimal performance during training.

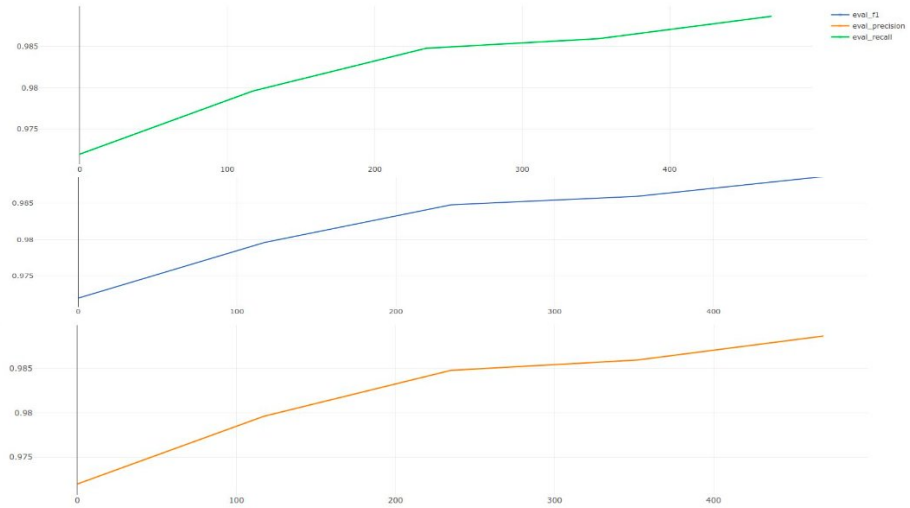

Figure 3.4: Metrics of token classifiction

```
def training_dataloader(args, tokenizer, labels,
    pad_token_label_id, batch, mode="train"):
    train_dataset = FunsdDataset(args, tokenizer, labels,
    pad_token_label_id, mode="train")
    train_sampler = RandomSampler(train_dataset)
    return DataLoader(train_dataset,
        sampler=train_sampler,
        batch_size=batch)
```

```
def testing_dataloader(args, tokenizer, labels,
    pad_token_label_id,batch, mode="test"):
    eval_dataset = FunsdDataset(args, tokenizer,
    labels, pad_token_label_id, mode="test")
    eval_sampler = SequentialSampler(eval_dataset)
```

```
    return DataLoader(eval_dataset,
        sampler=eval_sampler,
        batch_size=batch)

def layoutlm_training(train_dataloader, device, model, lr,
    epochs, eval_dataloader, pad_token_label_id, label_map,
    writer):
    optimizer = AdamW(model.parameters(), lr=lr)
    global_step = 0
    num_train_epochs = epochs
    t_total = len(train_dataloader) * num_train_epochs
    model.train()
    loss_dict = {}
    for epoch in range(num_train_epochs):
        running_loss = 0.0
        step = 0
        for batch in tqdm(train_dataloader, desc="Training"):
            input_ids = batch[0].to(device)
            bbox = batch[4].to(device)
            attention_mask = batch[1].to(device)
            token_type_ids = batch[2].to(device)
            labels = batch[3].to(device)
            outputs = model(input_ids=input_ids, bbox=bbox,
            attention_mask=attention_mask,
            token_type_ids=token_type_ids,
                           labels=labels)
            loss = outputs.loss
            if global_step % 100 == 0:
                print(f"Loss after {global_step} steps:
                {loss.item()}")
            loss.backward()
```

```
            # update
            optimizer.step()
            optimizer.zero_grad()
            global_step += 1

            running_loss += loss.item()
            step += 1
        out_label_list, preds_list, eval_loss =
        layoutlm_evaluation(
            model, eval_dataloader, device, pad_token_label_id,
            label_map)
            results = layoutlm_eval_metrics(
            out_label_list, preds_list, eval_loss)
    return loss_dict, global_step, num_train_epochs,
    t_total, lr
```

iii. Training Monitoring

The author integrated training process with MLFlow for monitoring the performance. The monitoring the training process to ensure that the model is converging and learning from the dataset.The tools used for monitoring is MLFlow to track key metrics during training.MLflow track the number of parameters, metrics such as F1-score, precison, recall and each epoch loss.Also, save the requirements files and model. The author monitor all these things by using MLFlow.

```
mlflow.set_experiment("layoutlm training")
with mlflow.start_run(run_name=experiment_name):
  loss_dict,global_step,num_train_epochs,t_total,lr =
  layoutlm_training(train_dataloader,device,model,
  config_args['lr'],
  config_args['epochs'],eval_dataloader,pad_token_label_id,
  label_map,writer)
```

```
mlflow.log_param("global_step",global_step)
mlflow.log_param("train_epochs",num_train_epochs)
mlflow.log_param("learning rate",lr)
mlflow.log_dict(loss_dict,config_args['loss_file_name'])
mlflow.pytorch.log_model(model,config_args['model_name'])
out_label_list,preds_list,eval_loss = layoutlm_evaluation(
model,eval_dataloader,device,
pad_token_label_id,label_map)
results = layoutlm_eval_metrics(out_label_list,
preds_list,eval_loss)
mlflow.log_dict(results,config_args['eval_loss_name'])
mlflow.log_metric("avg_eval_loss",results['loss'])
mlflow.log_metric("avg_eval_precision",results['precision'])
mlflow.log_metric("avg_eval_recall",results['recall'])
mlflow.log_metric("avg_eval_f1",results['f1'])
```

iv. Evaluation on Validation Data

The author evaluate the model on a validation dataset to assess its performance. Use metrics such as precision, recall, and F1 score to measure the model's accuracy in capturing both text and layout elements. The final precision, recall and F1-score is 0.986. Furthermore, the test the model performance by using real dataset.

```
def layoutlm_evaluation(model,eval_dataloader,device,
    pad_token_label_id,label_map):
    eval_loss = 0.0
    nb_eval_steps = 0
    preds = None
    out_label_ids = None

    # put model in evaluation mode
    model.eval()
```

```
for batch in tqdm(eval_dataloader, desc="Evaluating"):
    with torch.no_grad():
        input_ids = batch[0].to(device)
        bbox = batch[4].to(device)
        attention_mask = batch[1].to(device)
        token_type_ids = batch[2].to(device)
        labels = batch[3].to(device)
        outputs = model(input_ids=input_ids, bbox=bbox,
        attention_mask=attention_mask, token_type_ids=
        token_type_ids,
                        labels=labels)
        tmp_eval_loss = outputs.loss
        logits = outputs.logits

        eval_loss += tmp_eval_loss.item()
        nb_eval_steps += 1

        if preds is None:
            preds = logits.detach().cpu().numpy()
            out_label_ids = labels.detach().cpu().numpy()
        else:
            preds = np.append(preds, logits.detach().
            cpu().numpy(), axis=0)
            out_label_ids = np.append(
                out_label_ids, labels.detach().cpu().numpy(
                axis=0
            )

# compute average evaluation loss
eval_loss = eval_loss / nb_eval_steps
preds = np.argmax(preds, axis=2)
```

```
out_label_list = [[] for _ in range(out_label_ids.shape[0])
preds_list = [[] for _ in range(out_label_ids.shape[0])]

for i in range(out_label_ids.shape[0]):
    for j in range(out_label_ids.shape[1]):
        if out_label_ids[i, j] != pad_token_label_id:
            out_label_list[i].append(label_map
            [out_label_ids[i][j]])
            preds_list[i].append(label_map[preds[i][j]])
return out_label_list,preds_list,eval_loss
```

4. Table Extraction

   i. Data Collection

   The author collected a diverse and representative dataset of tables.The dataset covers various table structures, layouts, and content types to enhance the model's generalization capabilities. The table dataset are collected from the different types of documents such as purchase invoice, vendor invoices, sales order etcs. The dataset contains the different types of documents such as structure documents, unstructure documents that have a border table, border less table and semi border table. The author collected this kind of documents for the table training.

   ii. Data Annotation

   The author annotated the table, table column and table row from different types of documents . Annotate the collected dataset with labeled information about table structures, table column header, table column header and table row, table column, table and table spanning cell. This annotated data serves as the ground truth for training the model and evaluation of the model. The Label studio is used for table data annotation which is a free tools for annotation.

   iii. Training

   The author fine tuned Table Transformers pre-trained model on collected new

dataset. The labeled dataset obtained from the data collection which is pass into preprocessing step and perform different task such tokenization, encoding categorical features, handling missing values, and scaling numerical features. The preprocessed table data is send into the table transformer model's architecture, which includes the layers, attention mechanisms, and other components. This architecture defines how the model processes input data and generates predictions.The initialized model with randomly initialized weights. Initialization is a crucial step that sets the starting point for training.The initialized model and the preprocessed dataset and The trained model with optimized weights.The model takes a batch of table data as input. In Forward pass, he input data is processed through the model to generate predictions, Compare the model's predictions with the ground truth labels to compute a loss.Then use backpropagation to update the model's weights based on the calculated loss.This process will repeated for multiple epochs until the model converges.The table transformers training parameters are learning rate is 5e-5, batch size is 2, weight decay is 0.01, epoch is 100, lr drop is 0.000000000001 and dropout is 0.1. This is the parameter for model training after hyperparameter tuning.

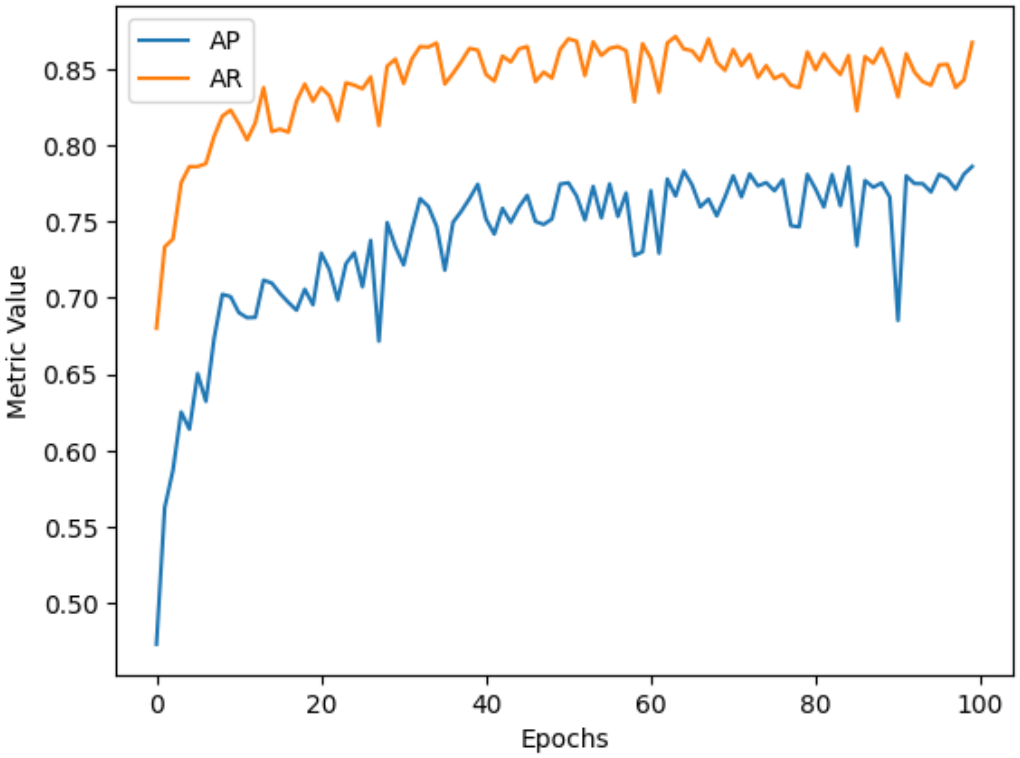

Figure 3.5: Metrics

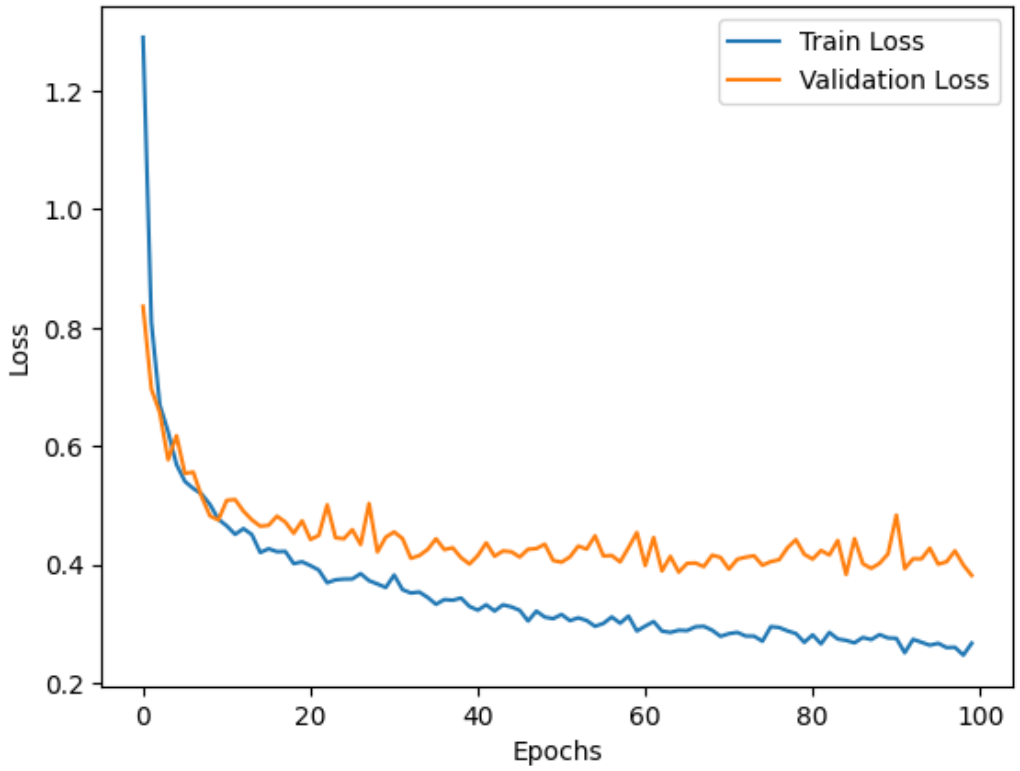

Figure 3.6: Training Loss Vs Validation Loss

```
def get_data(args):
    class_map = get_class_map(args.data_type)
    if args.mode == "train":
        dataset_train = PDFTablesDataset(
            os.path.join(args.data_root_dir, "train"),
            get_transform(args.data_type, "train"),
            do_crop=False,
            max_size=args.train_max_size,
            include_eval=False,
            max_neg=0,
            make_coco=False,
            image_extension=".jpg",
            xml_fileset="train_filelist.txt",
            class_map=class_map)
        dataset_val = PDFTablesDataset(
        os.path.join(args.data_root_dir, "val"),
       get_transform(args.data_type, "val"),
       do_crop=False,
       max_size=args.val_max_size,
       include_eval=False,
       make_coco=True,
       image_extension=".jpg",
       xml_fileset="val_filelist.txt",
       class_map=class_map)
        sampler_train = torch.utils.data.RandomSampler(
        dataset_train)
        sampler_val = torch.utils.data.SequentialSampler(
        dataset_val)
        batch_sampler_train = torch.utils.data.BatchSampler(
        sampler_train,args.batch_size,drop_last=True)
```

```
        data_loader_train = DataLoader(dataset_train,
                    batch_sampler=batch_sampler_train,
                    collate_fn=utils.collate_fn,
                    num_workers=args.num_workers)
        data_loader_val = DataLoader(dataset_val,
                        2 * args.batch_size,
                        sampler=sampler_val,
                        drop_last=False,
                        collate_fn=utils.collate_fn,
                        num_workers=args.num_workers)
        return data_loader_train, data_loader_val,
        dataset_val, len(dataset_train)
    elif args.mode == "eval":
        dataset_test = PDFTablesDataset(os.path.join
        (args.data_root_dir,"test"),
        get_transform(args.data_type, "val"),
        do_crop=False,
        max_size=args.test_max_size,
        make_coco=True,
        include_eval=True,
        image_extension=".jpg",
        xml_fileset="test_filelist.txt",
        class_map=class_map)
        sampler_test = torch.utils.data.SequentialSampler(
        dataset_test)
        data_loader_test = DataLoader(dataset_test,
          2 * args.batch_size,
          sampler=sampler_test,
          drop_last=False,
          collate_fn=utils.collate_fn,
        num_workers=args.num_workers)
        return data_loader_test, dataset_test
```

```
    elif args.mode == "grits" or args.mode == "grits-all":
        dataset_test = PDFTablesDataset(os.path.join(
        args.data_root_dir, "test"),
        RandomMaxResize(1000, 1000),
        include_original=True,
        max_size=args.max_test_size,
        make_coco=False,
        image_extension=".jpg",
        xml_fileset="test_filelist.txt",
        class_map=class_map)
        return dataset_test


def get_model(args, device):
    model, criterion, postprocessors = build_model(args)
    model.to(device)
    if args.model_load_path:
        print("loading model from checkpoint")
        loaded_state_dict = torch.load(args.model_load_path,
                                       map_location=device)
        model_state_dict = model.state_dict()
        pretrained_dict = {
            k: v
            for k, v in loaded_state_dict.items()
            if k in model_state_dict and model_state_dict[k].
            shape == v.shape
        }
        model_state_dict.update(pretrained_dict)
        model.load_state_dict(model_state_dict, strict=True)
    return model, criterion, postprocessors


def train(args, model, criterion, postprocessors, device):
```

```
print("loading data")
dataloading_time = datetime.now()
data_loader_train, data_loader_val, dataset_val,
train_len = get_data(args)
print("finished loading data in :",
datetime.now() - dataloading_time)
model_without_ddp = model
param_dicts = [
    {
        "params": [
            p for n, p in model_without_ddp
            .named_parameters()
            if "backbone" not in n and
            p.requires_grad
        ]
    },
    {
        "params": [
            p for n, p in model_without_ddp.
            named_parameters()
            if "backbone" in n and p.requires_grad
        ],
        "lr":
        args.lr_backbone,
    },
]
optimizer = torch.optim.AdamW(
param_dicts,lr=args.lr,
weight_decay=args.weight_decay)
lr_scheduler = torch.optim.lr_scheduler.
StepLR(optimizer,step_size=args.lr_drop,
gamma=args.lr_gamma)
```

```
max_batches_per_epoch = int(train_len /
args.batch_size)
print("Max batches per epoch: {}".format(
max_batches_per_epoch))
resume_checkpoint = False
if args.model_load_path:
    checkpoint = torch.load(args.model_load_path,
    map_location='cpu')
    if 'model_state_dict' in checkpoint:
        model.load_state_dict(
        checkpoint['model_state_dict'])
    model.to(device)
    if not args.load_weights_only and
    'optimizer_state_dict' in checkpoint:
        optimizer.load_state_dict(checkpoint
        ['optimizer_state_dict'])
        resume_checkpoint = True
    elif args.load_weights_only:
        print("*** WARNING: Resuming training and ignoring
        optimzer state. "
      "Training will resume with new initialized values. "
      "To use current optimizer state, remove the
      --load_weights_only flag.")
    else:
        print("*** ERROR: Optimizer state of saved
        checkpoint not found. "
      "To resume training with new initialized values add
      the --load_weights_only flag.")
        raise Exception("ERROR: Optimizer state of saved
        checkpoint not found. Must add --load_weights_only
        flag to resume training without.")
```

```
    if not args.load_weights_only and 'epoch' in
    checkpoint: args.start_epoch = checkpoint['epoch'] + 1
    elif args.load_weights_only:
        print("*** WARNING: Resuming training and ignoring
        previously saved epoch. "
          "To resume from previously saved epoch,
          remove the --load_weights_only flag.")
    else:
        print("*** WARNING: Epoch of saved model not found.
        Starting at epoch {}.".format(args.start_epoch))
if args.model_save_dir:
    output_directory = args.model_save_dir
# If resuming from a checkpoint with optimizer state, save
same directory
elif args.model_load_path and resume_checkpoint:
    output_directory = os.path.split(args.model_load_path)
    [0]
else:
    run_date = datetime.now().strftime("%Y%m%d%H%M%S")
    output_directory = os.path.join(args.data_root_dir,
    "output",run_date)
if not os.path.exists(output_directory):
    os.makedirs(output_directory)
print("Output directory: ", output_directory)
model_save_path = os.path.join(output_directory,
'model.pth')
print("Output model path: ", model_save_path)
if not resume_checkpoint and os.path.exists(
model_save_path):
    print("*** WARNING: Output model path exists but
    is not being
    used to resume training; training will overwrite it.")
```

```
if args.start_epoch >= args.epochs:
    print("*** WARNING: Starting epoch ({}) is greater or
    equal to the
    number of training epochs ({}).".format(
        args.start_epoch, args.epochs
    ))
print("Start training")
start_time = datetime.now()
for epoch in range(args.start_epoch, args.epochs):
    print('-' * 100)
    epoch_timing = datetime.now()
    train_stats = train_one_epoch(
        model,
        criterion,
        data_loader_train,
        optimizer,
        device,
        epoch,
        args.clip_max_norm,
        max_batches_per_epoch=max_batches_per_epoch,
        print_freq=1000)
    print("Epoch completed in ",
    datetime.now() - epoch_timing)
    lr_scheduler.step()
    pubmed_stats, coco_evaluator = evaluate(model,
    criterion,postprocessors,data_loader_val,
    dataset_val,device, None)
    print("pubmed: AP50: {:.3f}, AP75: {:.3f}, AP: {:.3f},
    AR: {:.3f}".format(pubmed_stats['coco_eval_bbox'][1],
             pubmed_stats['coco_eval_bbox'][2],
             pubmed_stats['coco_eval_bbox'][0],
             pubmed_stats['coco_eval_bbox'][8]))
```

```
            # Save current model training progress
            torch.save({'epoch': epoch,
                'model_state_dict': model.state_dict(),
                'optimizer_state_dict': optimizer.state_dict(),
                }, model_save_path)
            # Save checkpoint for evaluation
            if (epoch+1) % args.checkpoint_freq == 0:
                model_save_path_epoch = os.path.join(
                output_directory,
                'model_' + str(epoch+1) + '.pth')
                torch.save(model.state_dict(),
                model_save_path_epoch)
        print('Total training time: ', datetime.now() - start_time)
```

iv. Model Evaluation

The Author Evaluate the trained model on the validation set to assess its accuracy, precision, and recall in recognizing table structures and content. Fine-tune the model based on evaluation results.Author evaluate the model's performance on a separate dataset not used during training to assess generalization.

5. Deployment

The author created three microservices, each dedicated to a specific task.OCR Engine Microservice is responsible for extracting text information from images or documents, Token Classification Microservice is handles the classification of tokens within the extracted text, possibly identifying entities or categories and Table Extraction Microservice is Focuses on extracting structured tabular data from text or documents.Each microservice is packaged into a Docker container. Docker containers encapsulate the application, its dependencies, and runtime environment, ensuring consistency across different environments. The microservices are deployed as API endpoints.

Chapter 4

CONCLUSION AND LEARNING OUTCOMES

4.1 Conclusion

To conclude, the author found that the internship was very beneficial as a part of the development of career in the field of Artificial Intelligence and the experience gained through this would be helpful and beneficial for future opportunities. Some of learning Outcomes of me after completion of this intern session are:

- Understood deploy about the different Machine learning and deep learning methods.
- Understood software deployment methodologies and process.
- ExploreFile Handling different technology stack such as Microsoft Dynamics 365 and Business Central.
- Gained knowledge about company workflow and their structure.
- Network with professional in the field.

4.2 Learning Outcomes

Over the course of the internship, the intern demonstrated substantial growth and development across various dimensions. Firstly, in terms of skill development, the intern significantly enhanced their communication skills, particularly in unfamiliar professional interactions. Additionally, formal writing skills were refined, focusing on the creation of reports and effective communication with superiors. The intern also displayed improved research proficiency, becoming more efficient in obtaining relevant information.Secondly, with regards to understanding real-world application, the intern gained a comprehensive understanding of workplace dynamics within the legal environment. This included a deep dive into the operating procedures of state and federal court systems, as well as hands-on experience in streamlining human resource management policies. The integration of new applications and skills further enriched the intern's academic knowledge, providing a more holistic perspective on the field. Some of them are listed below:

- Improved communication skills in unfamiliar professional interactions.
- Refined formal writing skills for reports and communication with superiors.
- Enhanced research proficiency for more efficient information gathering.
- Gained practical experience in implementing AI models to solve real-world problems.
- Acquired insights into the challenges and considerations involved in deploying AI solutions in a business context.

References

Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. (2022). Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th acm international conference on multimedia* (pp. 4083–4091).

Li, Y., Huang, Z., Yan, J., Zhou, Y., Ye, F., & Liu, X. (2021). Gfte: graph-based financial table extraction. In *Pattern recognition. icpr international workshops and challenges: Virtual event, january 10–15, 2021, proceedings, part ii* (pp. 644–658).

Liao, M., Wan, Z., Yao, C., Chen, K., & Bai, X. (2020). Real-time scene text detection with differentiable binarization. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 11474–11481).

Rajavelu, A., Musavi, M. T., & Shirvaikar, M. V. (1989). A neural network approach to character recognition. *Neural networks*, *2*(5), 387–393.

Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, *39*(11), 2298–2304.

Smock, B., Pesala, R., & Abraham, R. (2022). Pubtables-1m: Towards comprehensive table extraction from unstructured documents. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 4634–4642).

Smock, B., Pesala, R., & Abraham, R. (2023a). Aligning benchmark datasets for table structure recognition. *arXiv preprint arXiv:2303.00716*.

Smock, B., Pesala, R., & Abraham, R. (2023b). Grits: Grid table similarity metric for table structure recognition. In *International conference on document analysis and recognition* (pp. 535–549).

Wei, T. C., Sheikh, U., & Ab Rahman, A. A.-H. (2018). Improved optical character recognition with deep neural network. In *2018 ieee 14th international colloquium on signal processing & its applications (cspa)* (pp. 245–249).

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1192–1200).