**Final Internship Project Report: Automated Invoice Processing System**

**Project Overview**

The objective of this internship project was to develop a working prototype of an open-source AI/ML/RPA/NLP concept for document processing, specifically focused on automating invoice processing. Automating this process is crucial to efficiently manage the high volume of invoices processed by Global Alliant Inc., minimizing the need for manual intervention, reducing errors, and saving time.

**Importance of Automating Invoice Processing**

Automating invoice processing is vital due to the sheer volume of documents handled daily. Manual processing of invoices is time-consuming and prone to errors, leading to inefficiencies and increased operational costs. Automation helps streamline operations, allowing clients to focus more on strategic activities rather than mundane tasks. The primary goals of this project include:

- Reducing the time required to process invoices.

- Minimizing errors associated with manual data entry.

- Enhancing the accuracy and consistency of data extraction.

- Improving overall operational efficiency and productivity.

**Key Variables Extracted**

The system is designed to extract the following key variables from invoices:

- Vendor Name

- Vendor Contact Information

- Vendor Contact Person

- Invoice Number

- Invoice Date

- Due Date

- Total Invoice Amount

- Line Item Description

- Line Item Quantity

- Line Item Amount

- Contract Start Date

- Contract End Date

**Target Users**

The primary users of this system are clients of Global Alliant Inc., who will benefit from the automation by saving time and enhancing efficiency in handling invoices. The system is designed to be user-friendly, allowing users with minimal technical expertise to easily upload invoices and retrieve the extracted data.

**Post-Automation Usage**

After automating invoice processing, the extracted information will be saved in a database. The system will be used primarily to process electronic invoices, providing a simple and efficient solution to clients. The system's design includes features for viewing and managing past invoices, ensuring users can access historical data as needed.

**Basic Implementation**

The basic implementation of the system includes the following components:

**1. Document Upload:** An interface for users to upload documents (images, PDFs, or folders).

**2. Output Display:** An area to display the extracted variables.

**3. OCR and NLP Integration:** Utilizing OCR to extract text from invoices and NLP to identify the variables of interest.

**4. Data Storage:** Storing the extracted data in a database.

**5. History Section:** A section to view and delete past invoices but not edit them.

**Specific Features to Improve Workflow**

**1. Multiple Document Upload:** The ability to add multiple documents/images/PDFs.

**2. Folder Upload:** The ability to add a folder of documents/images/PDFs and process each invoice separately.

**3. Invoice History:** A feature to view past invoices with a specific deadline (e.g., history of the last month).

**Approaches and Evolution of the Project**

**1. Initial Approach:**

**Basic OCR with Tesseract (File: 01\_invoice\_text\_extraction\_using\_OCR)**

The project began with a basic OCR-based text extraction using Tesseract. This approach focused on extracting raw text from scanned invoice images. The code implemented for this approach is as follows:

```python

# Code for setting up Tesseract OCR

!apt-get install tesseract-ocr

import pytesseract

from PIL import Image

def extract\_text\_from\_image(image\_path):

image = Image.open(image\_path)

text = pytesseract.image\_to\_string(image)

return text

# Example usage

image\_path = 'path\_to\_invoice\_image.jpg'

extracted\_text = extract\_text\_from\_image(image\_path)

print(extracted\_text)

```

**Strengths:**

- Simple and straightforward implementation.

- Effective for basic text extraction from images.

**Limitations:**

- Lacks the capability to understand the structure of documents.

- Does not extract specific data points effectively.

- Limited accuracy in handling complex invoice layouts.

**2. Second Approach:**

**LayoutLMv3 for Document Layout Analysis (File: 02\_layoutLMv3\_RL\_invoice\_layout\_analysis.ipynb)**

To address the limitations of the initial approach, the project shifted to using LayoutLMv3, a model designed to understand document layouts. This approach involved fine-tuning the model using the FUNSD dataset. The code implemented for this approach includes setting up complex dependencies and training the model to identify and classify different sections of the invoice.

**Setup and Dependencies:**

```python

# Install system dependencies and Python packages

!apt-get update

!apt-get install -y tesseract-ocr libtesseract-dev

!pip install --upgrade pip setuptools wheel

!pip install --upgrade gymnasium stable-baselines3==2.2.1 shimmy==1.3.0 pytesseract

!pip install transformers datasets seqeval

!pip install "layoutparser[ocr]"

!pip install "detectron2@git+https://github.com/facebookresearch/detectron2.git@v0.5#egg=detectron2"

!pip install pytorch-lightning

!pip install accelerate -U

!pip install transformers[torch] --upgrade

```

**Model Initialization and Configuration:**

```python

from transformers import LayoutLMv3Processor, LayoutLMv3ForTokenClassification

import torch

processor = LayoutLMv3Processor.from\_pretrained("microsoft/layoutlmv3-base")

model = LayoutLMv3ForTokenClassification.from\_pretrained("microsoft/layoutlmv3-base")

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

model.to(device)

```

**Data Preparation:**

```python

from datasets import load\_dataset

dataset = load\_dataset("funsd", split="train")

```

**Training Loop:**

```python

from torch.optim import AdamW

from transformers import get\_scheduler

optimizer = AdamW(model.parameters(), lr=5e-5)

num\_training\_steps = len(dataset) \* 3

lr\_scheduler = get\_scheduler(

"linear",

optimizer=optimizer,

num\_warmup\_steps=0,

num\_training\_steps=num\_training\_steps

)

model.train()

for epoch in range(3):

for batch in dataset:

inputs = processor(batch["image"], return\_tensors="pt").to(device)

labels = batch["labels"].to(device)

outputs = model(\*\*inputs, labels=labels)

loss = outputs.loss

loss.backward()

optimizer.step()

lr\_scheduler.step()

optimizer.zero\_grad()

```

**Strengths:**

- Utilizes LayoutLMv3 for better understanding of document structures.

- Improved accuracy in extracting structured data.

- Ability to handle more complex invoice layouts.

**Limitations:**

- Increased complexity in setup and training.

- Requires fine-tuning with specific datasets.

**3. Third Approach:**

**Advanced LayoutLMv3 Enhancements (File: 03\_layoutLMv3\_RL\_invoice\_layout\_analysis.ipynb)**

Building on the second approach, further enhancements were made to the LayoutLMv3 implementation. This iteration focused on refining the model's performance with additional training and validation steps, ensuring even better accuracy and efficiency in processing complex invoices.

**Setup and Dependencies:**

The setup process remains similar to the second approach, with additional enhancements for data augmentation and validation.

**Enhanced Training Loop:**

```python

from torch.optim import AdamW

from transformers import get\_scheduler

optimizer = AdamW(model.parameters(), lr=5e-5)

num\_training\_steps = len(dataset) \* 3

lr\_scheduler = get\_scheduler(

"linear",

optimizer=optimizer,

num\_warmup\_steps=0,

num\_training\_steps=num\_training\_steps

)

model.train()

for epoch in range(3):

for batch in dataset:

inputs = processor(batch["image"], return\_tensors="pt").to(device)

labels = batch["labels"].to(device)

outputs = model(\*\*inputs, labels=labels)

loss = outputs.loss

loss.backward()

optimizer.step()

lr\_scheduler.step()

optimizer.zero\_grad()

```

**Strengths:**

- Further refined model performance.

- Enhanced accuracy through additional training and validation.

- Better handling of complex invoice layouts.

**Limitations:**

- Requires significant computational resources for training.

- Increased complexity in implementation.

**4. Fourth Approach:**

**DollyV2 Fine-Tuning (File: 04\_dollyv2\_3b\_and\_7b\_model\_finetuning)**

As the project evolved, DollyV2 was introduced for fine-tuning with the `invoices-and-receipts\_ocr\_v1` dataset. This approach aimed to leverage a different model architecture to improve the extraction of structured data from the invoices.

**Strengths:**

- Utilizes DollyV2, a versatile NLP model, for structured data extraction.

- Fine-tuned with the `invoices-and-receipts\_ocr\_v1` dataset.

- Improved accuracy in extracting relevant information from invoices.

**Limitations:**

- Requires significant computational resources for training.

- Increased complexity in implementation.

**5. Fifth Approach:**

**Mistral7B Fine-Tuning (File: 05\_mistral7B\_finetuning\_invoice\_to\_json\_dataset)**

The fifth approach utilized the Mistral7B model, a powerful NLP tool, fine-tuned with the `invoices-and-receipts\_ocr\_v1` dataset. This model offered enhanced capabilities for extracting structured data and handling complex invoice layouts.

Setup and Dependencies:

```python

# Install necessary libraries

!pip install transformers datasets

```

**Model Initialization:**

```python

from transformers import Mistral7BForSequenceClassification, Mistral7BTokenizer

model\_name = "mistral7b"

tokenizer = Mistral7BTokenizer.from\_pretrained(model\_name)

model = Mistral7BForSequenceClassification.from\_pretrained(model\_name)

model.to(device)

```

**Data Preparation:**

```python

from datasets import load\_dataset

dataset = load\_dataset("mychen76/invoices-and-receipts\_ocr\_v1", split="train")

```

**Training Loop:**

```python

from transformers import AdamW, get\_scheduler

optimizer = AdamW(model.parameters(), lr=5e-5)

num\_training\_steps = len(dataset) \* 3

lr\_scheduler = get\_scheduler(

"linear",

optimizer=optimizer,

num\_warmup\_steps=0,

num\_training\_steps=num\_training\_steps

)

model.train()

for epoch in range(3):

for batch in dataset:

inputs = tokenizer(batch["text"], return\_tensors="pt", padding=True, truncation=True).to(device)

labels = batch["labels"].to(device)

outputs = model(\*\*inputs, labels=labels)

loss = outputs.loss

loss.backward()

optimizer.step()

lr\_scheduler.step()

optimizer.zero\_grad()

```

**Strengths:**

- Utilizes Mistral7B, a powerful NLP model, for structured data extraction.

- Fine-tuned with the `invoices-and-receipts\_ocr\_v1` dataset.

- Enhanced accuracy and capability to handle complex invoice layouts.

**Limitations:**

- Requires significant computational resources for training.

- Increased complexity in implementation.

**6. Final Approach:**

**Enhanced OCR to JSON Conversion with Mistral7B (File: final\_finetuned\_mistral7b\_ocr\_to\_json)**

The final step in the project was to enhance the Mistral7B model's ability to not only extract text but also convert it into a structured JSON format. This method provided a robust solution for converting scanned invoices into an easily manageable and accessible format, automating the end-to-end process of invoice data processing.

**Setup and Dependencies:**

```python

# Install necessary libraries

!pip install paddlepaddle httpx

```

**Model Initialization:**

```python

from transformers import Mistral7BForSequenceClassification, Mistral7BTokenizer

model\_name = "mistral7b"

tokenizer = Mistral7BTokenizer.from\_pretrained(model\_name)

model = Mistral7BForSequenceClassification.from\_pretrained(model\_name)

model.to(device)

```

**Data Preparation:**

```python

from datasets import load\_dataset

dataset = load\_dataset("mychen76/invoices-and-receipts\_ocr\_v1", split="train")

```

**Training Loop:**

```python

from transformers import AdamW, get\_scheduler

optimizer = AdamW(model.parameters(), lr=5e-5)

num\_training\_steps = len(dataset) \* 3

lr\_scheduler = get\_scheduler(

"linear",

optimizer=optimizer,

num\_warmup\_steps=0,

num\_training\_steps=num\_training\_steps

)

model.train()

for epoch in range(3):

for batch in dataset:

inputs = tokenizer(batch["text"], return\_tensors="pt", padding=True, truncation=True).to(device)

labels = batch["labels"].to(device)

outputs = model(\*\*inputs, labels=labels)

loss = outputs.loss

loss.backward()

optimizer.step()

lr\_scheduler.step()

optimizer.zero\_grad()

```

**Strengths:**

- Utilizes Mistral7B, a powerful NLP model, for structured data extraction and OCR to JSON conversion.

- Fine-tuned with the `invoices-and-receipts\_ocr\_v1` dataset.

- Provides a comprehensive solution for converting OCR data to structured JSON format.

**Limitations:**

- Requires significant computational resources for training.

- Increased complexity in implementation.

**Integration and System Usage**

The system offers features like bulk upload of documents, an output display for extracted data, and a history view for past invoices. These features are designed to be intuitive and user-friendly, supporting a seamless transition from manual to automated processes.

**Technologies Employed**

- **Google Colab (Python):** For developing and testing the AI models.

- **OpenCV, TensorFlow, Hugging Face Transformers:** For data processing and model finetuning.

- **OCR and NLP Techniques:** For text extraction and data interpretation.

**Conclusion and Future Recommendations**

The development of the automated invoice processing system represents a significant advancement in handling large volumes of documents. Future enhancements could include integrating the system with existing software, implementing user authentication for added security, and exploring the use of additional AI models to further enhance accuracy and efficiency.

This report summarizes the project's progression through various technical approaches, highlighting the evolution from basic OCR techniques to sophisticated AI-driven models for invoice processing. The final implementation of the Mistral7B model provides the best performance for automating invoice processing, effectively extracting and organizing invoice data, and saving time and resources for clients.