TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

A
PROJECT REPORT
ON

"INFORMATION EXTRACTION FROM UNSTRUCTURED DATA"

SUBMITTED BY:
AAYUSH LAMMICHHANE(075 BCT 004)
AAYUSH NEUPANE(075 BCT 006)
ANKIT PAUDEL(075 BCT 013)
ASHISH LAMSAL(075 BCT 016)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

LALITPUR, NEPAL
30TH APRIL, 2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"Information Extraction from Unstructured Data"** submitted by **Aayush Lamichhane**, **Aayush Neupane**, **Ankit Paudel**, **Ashish Lamsal** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

....
Supervisor
Aman Shakya, Ph.D
Assistant Professor
Department of Electronics and
Computer Engineering,
Pulchowk Campus, IOE, TU.

....
Internal examiner
Person B
Assistant Professor
Department of Electronics and
Computer Engineering,
Pulchowk Campus, IOE, TU.

....
External examiner
Person C
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgement

We would like to express our deepest sense of gratitude to our project supervisor **Dr. Aman Shakya, PhD** for his expert guidance and continuous encouragement throughout to align this project with its target from its commencement to its completion.

We would also like to thank **Docsumo Nepal** for providing this opportunity to work on a real world project. We would like extend our thankfulness to the **Institute of Engineering, Pulchowk Campus** itself for this opportunity to aim and achieve our goals.

Further, We would like to thank our class mates for their valuable comments and suggestions. Their suggestions gave us inspiration to continuously improve our project. And finally, We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in this project.

Sincerely,
Aayush Lamichhane (075BCT004)
Aayush Nepuane (075BCT006)
Ankit Paudel (075BCT013)
Ashish Lamsal (075BCT016)

Abstract

In today's digital age, the digitization of paper documents like invoices and receipts has taken on more significance. Nevertheless, manually entering data from these papers can take a lot of time and be prone to mistakes, which causes inefficiencies and drives up expenses for enterprises. To solve this issue, we created a software platform that automates the process of collecting important data from scanned documents using deep learning technology, more specifically the LayoutLM architecture. Users can upload their scanned papers in bulk to our platform and choose which fields, including date, merchant name, and total amount, they want to extract. The technology is scalable and can manage high document volumes while preserving precision and effectiveness.The user-friendly interface makes it easy for users to upload and extract information from their scanned documents. Our platform offers significant benefits, including increased efficiency, accuracy, and cost savings, and has the potential to transform the way businesses handle physical documents. In this project, we will provide an overview of our software platform, including the technology behind it, its key features, and its potential applications.

Keywords: Intelligent Document Processing, Key Information Extraction, Deep Learning

Table Of Contents

Page of Approval **i**

Copyright **ii**

Acknowledgement **iii**

Abstract **iv**

List of Figures **2**

List of Tables **3**

List of Abbreviations **4**

1 Introduction **5**

1.1 Background 5

1.2 Problem Statement 5

1.3 Objectives 5

1.4 Scope 6

1.5 Features and Functionalities 6

2 Literature Review **7**

2.1 Related work 7

2.1.1 Rule-based Information Extraction 7

2.1.2 Machine Learning Based Information Extraction 8

2.1.3 Sequence-to-sequence models for end-to-end extraction . . . 9

3 Related Theory **10**

3.1 Deep Learning 10
3.1.1 RNN 10
3.1.2 CNN 11
3.1.3 Transformers 12
3.2 Optical Character Recognition(OCR) 13
3.2.1 docTR 14
3.3 Information Extraction 15
3.4 LayoutLM 16
3.4.1 The BERT Model 16
3.4.2 The LayoutLM Model 16
3.4.3 LayoutLM Architecture 17
3.4.4 Embeddings 18
3.4.5 Pre-training 18
3.4.6 Fine-tuning 18
3.4.7 Transfer Learning 19
3.4.8 Sequence Labeling 19
3.4.9 Cross-Entropy Loss 20
3.4.10 Tokenization 20
3.4.11 Classification Metrices 21

4 Methodology **23**

4.1 Data Collection 23
4.2 Data Annotation 23
4.3 Pretrained LayoutLM Model 24
4.3.1 Pre-training Dataset 24
4.3.2 Pre-training Tasks 24
4.3.3 Pre-training Configuration 25
4.4 Finetuning 26

4.4.1 OCR 27

4.4.2 Labeling 27

4.4.3 Tokenization 27

4.4.4 Data Pre-Processing 28

4.4.5 Calculating Embeddings 29

4.4.6 Multiheaded Attention 29

4.4.7 Feed Forward Layer 30

4.4.8 Token Classification 31

4.4.9 Loss Calculation 31

5 Architecture and Implementation **32**

5.1 Design Implementation 32

5.1.1 Overview of System Architecture 32

5.2 Technologies used 36

5.3 Deployment 37

5.4 Database Design 38

5.5 Web Application 40

6 Result and Analysis **41**

6.1 Training Loss 41

6.2 Overall Evaluation Metrics 42

6.3 Label-wise Evaluation Metrics 43

7 Epilogue **46**

7.1 Conclusion 46

7.2 Limitation and Future Enhancements 46

8 Appendix **48**

List of Figures

3.1 Transformer's Architecutre 12

3.2 LayoutLM Architecutre 17

4.1 Fine-tuning LayoutLM for token classification 26

4.2 Pre-trained LayoutLM's Architecture 30

5.1 System Block Diagram 32

5.2 Process flow 33

5.3 System Context Diagram 34

5.4 Data Flow Diagram 34

5.5 Sequence Diagram 35

5.6 System Frontend Deployment Architecture 37

5.7 System API Deployment Architecture 37

5.8 Database Schema 38

6.1 Loss during LayoutLM Finetuning (10 epochs) 41

6.2 Evaluation Metrics after each epoch on test set 42

8.1 Collected tax invoices 48

8.2 Generated invoices 48

8.3 Collected ATM receipts 49

8.4 Train/Test loss for SRIOE test dataset 49

8.5 Train/Test loss for ATM Receipt test dataset 50

8.6 Train/Test loss for Custom invoice test dataset 50

8.7 Evaluation Metrics for SRIOE dataset 51

8.8 Evaluation Metrics for ATM Receipt dataset 52

8.9 Evaluation Metrics for Custom invoice dataset 52
8.10 Web App Homepage 53
8.11 Web App Login Page 53
8.12 Web App Dashboard Page 54
8.13 Web App Annotation Page 54
8.14 Web App Results Page 54
8.15 Web App Single Output Page 55

List of Tables

3.1 **Confusion Matrix** 22

6.1 **Evaluation Metrics after 10 epochs** 43

6.2 **Label-wise Metrics on SROIE dataset (test) using LayoutLM Model** 43

6.3 **Label-wise Metrics on ATM receipt dataset (test) using LayoutLM Model** 44

6.4 **Label-wise Metrics on Custom Invoice dataset (test) using LayoutLM Model** 44

List of Abbreviations

AI Artificial Intelligence. 1

API Application Programming Interface. 1

BERT Bidirectional Encoder Representations from Transformers. 1

CRNN Convolutional Recurrent Neural Network. 1

LSTM Long Short Term Memory. 1

ML Machine Learning. 1

NLP Natural Language Processing. 1

PAN Permanent Account Number. 1

RNN Recurrent Neural Network. 1

1. Introduction

1.1 Background

Even though everything seems to be going digital, many of the information conveying between parties happen in hard copy. As the use of computer to process data is increasing day by day, the process of manually extracting information from documents can be time-consuming, error-prone, and expensive. Intelligent Document Processing (IDP) has become a crucial tool to automate this procedure with the advancements in artificial intelligence and machine learning.

To address this problem, we created a software platform with an IDP system that use deep learning to automatically extract data from images of documents. The software platform enables customers to mass upload their scanned documents and automatically extract key information. Our platform is built using deep learning technology, more especially the LayoutLM architecture, which is intended for information extraction and document layout analysis.

By using deep learning technologies, our platform is able to accurately identify key information from scanned documents, such as date, merchant name, total amount, and more. Users can choose which fields they want to extract from their documents, and our system will automatically extract and display the relevant information. This information can be used by the customer in any which required ways to fulfill their objectives. Our targeted market mainly includes large corporations that deals with public documents like governmental ones and the companies with large influx of hard copy documents like a Insurance company which has to analyze all the fields present in the document of their customer for their daily workings.

1.2 Problem Statement

Despite the increasing digitization of business processes, many businesses still rely on physical documents such as receipts and invoices for record-keeping and accounting purposes. Many companies are facing the problem of manual labor for data entry to be quite time consuming and costly. As the field of Artificial intelligence and machine learning matures, we in-vision to solve this problem so that the operation costs and time of large company is brought down massively. Thereby pushing us and them to a better financial position.

1.3 Objectives

The main objective of this project are:

- To make repetitive job manual data entry obsolete.

- To develop a good enough system for company to rely on their day to day operations
- Development of user-friendly interface that enables users to easily upload and extract information from their scanned documents.

1.4 Scope

Our project's objective is to develop a software platform that lets customers submit their scanned receipts and extract crucial data from those records. The platform will recognize and extract important fields from the submitted documents using the LayoutLM architecture, which is based on deep learning.

1.5 Features and Functionalities

The platform will offer the following features and functionalities:

- **Bulk document uploading**
  Users can upload multiple documents at once, either by selecting them from their local device or by connecting to a cloud storage service.
- **Field selection**
  Users can choose which fields they want to extract from the uploaded documents, such as date, amount, merchant name, and payment method.
- **Data extraction**
  The platform will use deep learning to identify and extract data from the selected fields.
- **Data visualization**
  Users will be able to view the extracted data in a clear and user-friendly format, such as a table or graph.

2. Literature Review

2.1 Related work

Automated data extraction technology has evolved significantly since the '90s. In this process, data is captured from documents through software tools and recorded digitally. They used AI and ML algorithms to capture important details from physical or electronic documents and save them in an organized manner in computer files. Methods of information Extraction can be categorized into following topics.

2.1.1 Rule-based Information Extraction

Information Extraction originates from the field of Natural Language Processing(NLP). Initially knowledge based systems like ATRANS systems that employed a lexicon hierarchy were developed. DARPA, the US defence agency also funded and encouraged research in information extraction by organizing a series of conferences. The conferences hosted competitions on information extraction. Most of the information extraction systems developed in the conferences emerged from research into rule-based systems in NLP and linguistics. Most systems were designed to apply series of rules on source text to extract relevant information. NER(Named Entity Recognition) and Part-Of-Speech(POS) tagging systems were developed using hundreds of rules.

Regular Expression search systems were also developed. Usually, for each field a regular expression would be associated. To improve the precision of such systems domain heuristics were applied. The context of the matched strings were also used to search for required information. The regular expression is usually designed by a domain expert. Manually constructing regular expressions is a tedious task. Automatic regular generation methods can be employed. The regular expressions may be learned from a corpus of documents[1] or derived from set of example entities[2].

Rule based Information System is not only restricted to regular expressions. Other methods involve extracting complex structures like tabular data using expectation driven approach.[3] A set of all possible column configurations involving global and local expectations is exhaustively searched after application of heuristics. After tagging a document using only the morphological structure, a secondary tagging is generated using regular expressions. As a post processing step, to improve accuracy, the extracted information is checked with the additional input resources for consistency and validity. This may be performed using domain specific database or pre-built ontologies. [4]

The leverage of rule bases information extraction systems over other methods is that the rules are interpret-able for both the users and the experts. These systems are also more flexible in that they can be employed for different tasks with little modification. When class specific knowledge is available, the task of information extraction can be made easier by document classification. If the documents can be linked to issuers easily this can be used to classify the document. The assigned class of the document can be used to more effectively extract the document.

For highly structured documents, the system can just employ a spatial mask to a previously constructed location of the information. This extraction strategy is greatly effective for structured documents provided the scanning process artifacts are correctly handled. For semi-structured documents, where the relative positioning is maintained, systems that depend on certain structure on the document can be developed e.g. table lines. Graph based and probability based models are also available that try to capture the similarity between the documents and effectively predict the information region.

Overall, rule based information extraction have been available for a long time. However, the design and maintenance of rule based systems require expertise and tedious labor. These efforts are being infeasible in the era where the need to capture recurring and complex structures of information is essential.

2.1.2 Machine Learning Based Information Extraction

With the development of machine learning techniques, statistical ML approaches have become the mainstream for a wide range of document analysis tasks, including information extraction. Most machine learning based information extraction method involve decomposition of document into various homogeneous parts and classification of all parts. The most common decomposition is the sequence of tokens. The task then is the label the sequence. The labels possibly contain the target information. Multiple labelling schemes exist that try to separate the targeted information from the tokens that carry irrelevant information. Once the text is decomposed into tokens or segment of tokens, the ML model is responsible with tagging the document with one label from the pre-defined set of possible labels.

One approach of information extraction is feature based machine learning. In the early days of ML based information extraction, classification of documents was done using models with limited expressiveness. This low model complexity was mitigated using highly informative features. A token's text value was used to derive categorical or Boolean features. For visually rich documents, layout features and font types were incorporated.

After feature extraction, the ML model is applied to the feature vector of a document to determine the label of its tokens or segments. The ML models range from simple logistic regression, kernel-based to tree-based models. These models however only capture interactions between features of the same document and not across documents. To address this problem, probabilistic graph models are also used. These graphical models label document tokens given their feature vectors. The training and inference times of these models grow exponentially with the size of the label set and the number of tokens. Therefore, the dependency graph is restricted to only simple structures. All models except kernel-based methods are trained using maximum likelihood estimation. Even though extraction logic learned to some extent by the ML model, feature-based approach still require substantial engineering efforts to design the features that are informative enough. While feature based approaches are still proposed for information extraction, most ML methods now employ deep models.

Like many other fields the task of information extraction has also transitioned from feature based to deep learning models. This method greatly reduces the engineering efforts. The deep model can be viewed as encoder and decoder. The encoder takes in feature vectors and creates a high level meaningful representation of tokens. The decoder then assigns information labels to tokens. During the deep learning process, the model derives linguistic representations of tokens that are helpful for information extraction. The token embeddings are then combined in later layers of encoder to learn contextualized document-level representations of tokens. The combination of embedding can be done in a fixed size neighbourhoods, however such model cannot capture the long range dependencies between tokens of documents.

The first deep models leveraging document-level representations of tokens were Recurrent Neural Network(RNN) encoders. Such models extract information by processing a document as a sequence of tokens. Using the powerful LSTM and GRU cells the models are able to store in their hidden states unbounded dependencies between the document tokens. The deep contextualized tokens representations are obtained by stacking neural networks. The final layer of the network is a fully-connected layer which classifies the tokens.

2.1.3 Sequence-to-sequence models for end-to-end extraction

Token level supervision is difficult for traditional machine learning based approach. Thus, sequence to sequence models that directly extract the information from input without token generation have originated. They employ a end-to-end method where a attention based sequence to sequence model is adapted to extract information from the raw extraction schemas. These models have been found to outperform token based system.[5] End to end models have better performance since they avoid the post processing step and they do not rely on text serializer for making predictions.

Pointer Networks were the first attempt at end to end information extraction task.[6] A single BLSTM network is used to encode the sequence of words and LSTM decoders. Each decoder has its own attention mechanism and is responsible for predicting a single target field. This approach is not able to extract information such as entities grouping multiple fields. In a follow up work by the same authors[7], a sequence-to-sequence model that extracted main information from the invoice was proposed. It has multi-modal architecture that can efficiently encode visually rich documents and parse the extracted information to normalized field values format. It is able to process unstructured information.

3. Related Theory

3.1 Deep Learning

Deep learning is a subset of machine learning that is based on artificial neural networks that are inspired by the structure and function of the human brain. Deep learning models are capable of automatically learning hierarchical representations of data, which allows them to achieve state-of-the-art performance on a wide range of complex tasks, including image and speech recognition, natural language processing, and many others.

The core building block of a deep learning model is a neural network, which is composed of layers of interconnected neurons. Each neuron computes a weighted sum of its input values, applies an activation function to the result, and passes the output to the next layer. The weights of the neurons are learned through a process called back-propagation, which adjusts the weights to minimize the error between the predicted and actual outputs of the model.

In our project, we used deep learning techniques to implement the LayoutLM architecture, which is based on pre-training deep bidirectional transformers for text recognition and layout analysis. The use of deep learning allowed us to achieve state-of-the-art performance on the task of extracting structured data from scanned documents. We also used convolutional and recurrent neural networks for image processing and natural language processing tasks, respectively. We trained our models using backpropagation and evaluated their performance using various metrics such as precision, recall, and F1-score.

Deep learning has revolutionized the field of machine learning and has enabled significant advances in various areas of artificial intelligence. However, deep learning also has its limitations, such as the need for large amounts of labeled data, the difficulty of interpreting the learned representations, and the high computational requirements for training and inference. Despite these challenges, deep learning continues to be a powerful tool for solving complex problems and will likely remain at the forefront of research in artificial intelligence for the foreseeable future.

3.1.1 RNN

Recurrent neural networks (RNNs) are a type of neural network that is well-suited for modeling sequential data, such as natural language text, time series data, and music. Unlike feedforward neural networks, which only process a fixed-length input, RNNs can handle input of variable length by maintaining a hidden state that represents the context of the previous inputs. This hidden state is updated at each time step based on the current input and the previous hidden state.

The core building block of an RNN is a recurrent layer, which computes the hidden state at each time step based on the current input and the previous hidden state. The output of the recurrent layer can be used to make a prediction at each time step, or it can be fed into another layer for further processing. There are several variants of RNNs, such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which are designed to address the problem of vanishing gradients, which can occur when the gradient signal becomes too small to propagate through many layers.

LSTMs are a type of RNN that is particularly effective at modeling long-term dependencies in sequential data. They have a memory cell that allows them to selectively remember or forget information based on the current input and the previous hidden state. This makes them well-suited for tasks such as language modeling, machine translation, and speech recognition, where the input sequence can be quite long.

3.1.2 CNN

Convolutional Neural Networks (CNNs) are a type of neural network that are particularly well-suited for image classification tasks. They were inspired by the structure and function of the visual cortex in animals, which is known to process visual information in a hierarchical and spatially-local manner. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which are arranged in a hierarchical manner to gradually learn more complex features from the input image.

The core building block of a CNN is the convolutional layer, which applies a set of learnable filters to the input image to extract features. The filters are typically small in size (e.g., 3x3 or 5x5) and slide over the entire image to produce a feature map. By applying multiple filters, the convolutional layer can learn to detect different types of features, such as edges, corners, and textures, at different locations in the image.

After the convolutional layer, a pooling layer is often applied to reduce the spatial dimensions of the feature map and improve computational efficiency. The pooling layer can perform operations such as max pooling or average pooling, which aggregate the output of adjacent neurons in the feature map to produce a smaller output.

The fully connected layers are used to map the learned features to the output classes. They take the flattened output of the previous layer and apply a set of weights to produce a prediction.

One of the key advantages of CNNs is their ability to automatically learn features from the input image, rather than relying on manual feature engineering. This makes them particularly effective for tasks such as object recognition, where the visual appearance of the object can vary widely depending on the viewpoint, lighting, and occlusion.

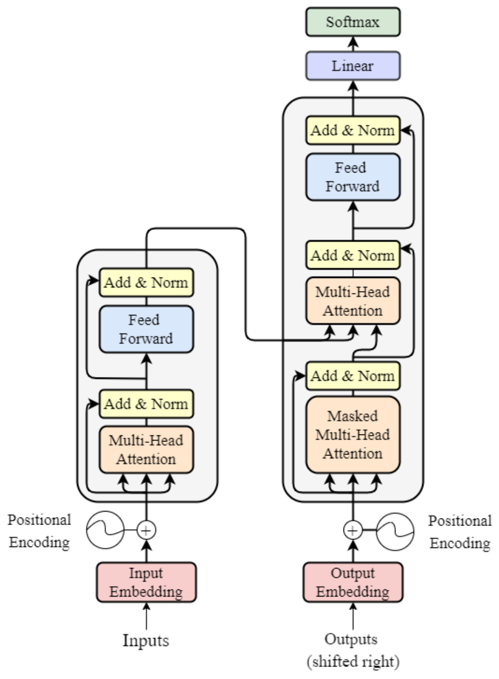


Figure 3.1: Transformer's Architecutre

3.1.3 Transformers

The Transformer architecture is a type of deep learning model that was introduced in 2017 for natural language processing tasks, such as machine translation and language modeling. It uses a self-attention mechanism to process sequential data, such as sentences or documents.

The self-attention mechanism allows the model to focus on different parts of the input sequence at each step, based on the relevance of each part to the current context. This enables the model to capture long-range dependencies and relationships between different parts of the input sequence, which is difficult for traditional recurrent neural networks.

The Transformer architecture consists of an encoder and a decoder, where the encoder processes the input sequence and the decoder generates the output sequence. The encoder consists of multiple layers, each of which has a multi-head self-attention mechanism and a feed-forward neural network. The output of each layer is passed to the next layer as input, allowing the model to capture increasingly complex patterns in the input sequence.

The Transformer architecture has become one of the most popular deep learning models in natural language processing due to its high accuracy and efficiency in processing long sequences.

LayoutLM is a deep learning model that is based on the encoder of the Transformer architecture. It extends the Transformer architecture to include layout analysis, which is the process of identifying the position and structure of text and graphics in a document.

LayoutLM is pre-trained on large amounts of document data to learn the relationships between text and layout features. During pre-training, the model is trained on two objectives simultaneously: predicting the next word in a sentence (language modeling) and predicting the next layout element in a document (layout modeling). This allows the model to learn to predict both the textual content and the layout structure of a document.

The pre-trained LayoutLM model can then be fine-tuned on specific downstream tasks, such as named entity recognition and relation extraction, by adding task-specific layers on top of the pre-trained model.

The use of LayoutLM as a core component in our project allows us to extract structured data from scanned documents, taking into account both the textual content and the layout structure of the document. The pre-trained LayoutLM model is fine-tuned on our specific task of extracting structured data from invoices, which allows it to learn the specific patterns and relationships that are relevant to this task.

Overall, the Transformer architecture is a powerful deep learning model that is widely used in natural language processing tasks. LayoutLM extends the Transformer architecture to include layout analysis, and is pre-trained on large amounts of document data to learn the relationships between text and layout features. Our project uses LayoutLM as a core component to extract structured data from scanned documents, taking advantage of its ability to capture both textual content and layout structure.

3.2 Optical Character Recognition(OCR)

Optical Character Recognition (OCR) is a technique that involves the conversion of scanned images of printed or handwritten text into machine-readable text that can be processed and analyzed by a computer. OCR is an essential component of document digitization as it enables the extraction of information from documents that have been scanned or photographed. OCR is used in a wide range of applications, including data entry, document indexing, and archival storage.

OCR technology has evolved significantly over the years, with the current state-of-the-art OCR systems relying on deep learning algorithms. Deep learning algorithms are neural networks that can learn to recognize patterns and features in images and text data. These algorithms have been shown to outperform traditional OCR techniques based on pattern recognition and rule-based approaches.

OCR involves several stages that include image preprocessing, text localization, text segmentation, character recognition, and post-processing.

Image preprocessing : This stage involves the preparation of the scanned image for OCR processing. It includes operations such as image resizing, normalization, and noise removal. The objective of this stage is to enhance the quality of the scanned image to improve OCR accuracy.

Text localization : This stage involves identifying the regions in the scanned image that contain text. It is accomplished using techniques such as edge detection, contour analysis, and connected component analysis. The output of this stage is a binary image that highlights the regions that contain text.

Text segmentation: This stage involves dividing the text regions into individual characters or words. It is achieved using techniques such as horizontal and vertical projection profiles, connected component analysis, and watershed segmentation. The output of this stage is a set of character or word images that can be recognized by an OCR engine.

Character recognition: This stage involves recognizing the characters in the segmented images. OCR engines use machine learning algorithms, such as convolutional neural networks (CNNs), to classify the characters based on their features. The output of this stage is a sequence of recognized characters.

Post-processing : This stage involves refining the output of the OCR engine to improve the accuracy of the recognized text. It includes operations such as spelling correction, punctuation and symbol recognition, and contextual analysis.

OCR systems are evaluated based on their accuracy, which is typically measured using metrics such as character error rate (CER) and word error rate (WER). The accuracy of OCR systems depends on several factors, including the quality of the scanned image, the complexity of the text, and the language being recognized.

3.2.1 docTR

To get the OCR output for our project we have used OCR based on deep learning approach, it is called docTR(Document Transformer) [8], docTR OCR uses a two-stage approach for end-to-end OCR, involving text detection and text recognition. The text detection stage involves localizing words in an image, while the text recognition stage involves identifying all the characters in the localized words. For each stage, docTR OCR offers a range of deep learning neural network architectures to choose from.

One of the architectures available for text detection is the DBNet [9], which stands for Differentiable Binarization Network. The DBNet architecture is a deep learning network that uses the Differentiable Binarization (DB) technique to detect text in the document image. The DB technique is a thresholding-based approach that converts the input image into a binary image by applying a differentiable thresholding function. The DBNet architecture consists of a feature extraction network and a binary classification network. The feature extraction network extracts a set of features from the input image, while the binary classification network produces a binary mask that indicates the text regions in the image.

The DBNet architecture is optimized for detecting text in various orientations, languages, and font styles, and it can handle text of different sizes and aspect ratios. Moreover, it has the advantage of being able to train end-to-end using backpropagation, which allows it to learn features and thresholding parameters directly from the data.

For text recognition, docTR OCR offers the CRNN [10] (Convolutional Recurrent Neural Network) architecture. The CRNN architecture is a deep learning network that combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The CNNs are used to extract features from the input image, while the RNNs are used to model the sequential nature of text. The output of the CRNN network is a sequence of characters, which can be decoded into text using a language model. The CRNN architecture is particularly effective for recognizing text in complex layouts, such as documents with multiple columns, tables, and images.

Overall, the DBNet and CRNN architectures in docTR OCR provide a robust and accurate solution for text detection and recognition in a wide range of document types and languages. By selecting the appropriate architecture for each stage of the OCR process, docTR OCR can achieve high accuracy and performance in document capture and recognition tasks.

3.3 Information Extraction

Information extraction (IE) is a subfield of natural language processing (NLP) that focuses on identifying and extracting structured information from unstructured or semi-structured sources of data, such as text documents, web pages, and emails. The goal of IE is to automatically extract relevant information from large volumes of textual data and transform it into a structured format, such as a database or a spreadsheet.The process of IE involves several steps, including:

Named entity recognition (NER): This step involves identifying and extracting named entities from the text, such as persons, organizations, locations, dates, and other entities of interest. NER is typically achieved using machine learning models that are trained on annotated datasets.

Relation extraction: Once named entities are identified, the next step is to identify the relationships between them. For example, if a document mentions two people, relation extraction can identify whether they are family members, co-workers, or have some other type of relationship.

Template-based extraction: In some cases, the information to be extracted follows a specific pattern or template. In such cases, template-based extraction can be used to identify and extract the relevant information. For example, in a resume, the name, address, education, and work experience sections follow a specific template that can be easily extracted using regular expressions.

Machine learning-based extraction: In cases where the information to be extracted is not easily defined by a template or pattern, machine learning-based approaches can be used. These approaches involve training machine learning models on annotated datasets to identify relevant information.

IE has a wide range of applications, including:

Business intelligence: IE can be used to extract and analyze data from various sources, such as financial reports, customer feedback, and social media. This can provide valuable insights for business decision-making.

Document classification: IE can be used to automatically categorize documents based on their content, such as legal documents, medical records, or news articles.

Information retrieval: IE can be used to extract relevant information from large volumes of unstructured data, such as web pages, to improve search results.

Fraud detection: IE can be used to detect fraudulent activity, such as credit card fraud or insurance fraud, by analyzing large volumes of data to identify patterns and anomalies.

3.4 LayoutLM

3.4.1 The BERT Model

The BERT model is an attention-based bidirectional language modeling approach. It has been verified that the BERT model shows effective knowledge transfer from the self-supervised task with large-scale training data. The architecture of BERT is basically a multi-layer bidirectional Transformer encoder. It accepts a sequence of tokens and stacks multiple layers to produce final representations. In detail, given a set of tokens processed using WordPiece, the input embeddings are computed by summing the corresponding word embeddings, position embeddings, and segment embeddings. Then, these input embeddings are passed through a multi-layer bidirectional Transformer that can generate contextualized representations with an adaptive attention mechanism. There are two steps in the BERT framework: pre-training and fine-tuning. During the pre-training, the model uses two objectives to learn the language representation: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), where MLM randomly masks some input tokens and the objective is to recover these masked tokens, and NSP is a binary classification task taking a pair of sentences as inputs and classifying whether they are two consecutive sentences. In the fine-tuning, task-specific datasets are used to update all parameters in an end-to-end way. The BERT model has been successfully applied in a set of NLP tasks.

3.4.2 The LayoutLM Model

Although BERT-like models become the state-of-the-art techniques on several challenging NLP tasks, they usually leverage text information only for any kind of inputs. When it comes to visually rich documents, there is much more information that can be encoded into the pre-trained model. Therefore, LayoutLM [11] was proposed to utilize the visually rich information from document layouts and align them with the input texts. Basically, there are two types of features which substantially improve the language representation in a visually rich document, which are: Document Layout Information. It is evident that the relative positions of words in a document contribute a lot to the semantic representation. Taking form understanding as an example, given a key in a form (e.g., “Passport ID:”), its corresponding value is much more likely on its right or below instead of on the left or above. Therefore, we can embed these relative positions information as 2-D position representation. Based on the self-attention mechanism within the Transformer, embedding 2-D position features into the language representation will better align the layout information with the semantic representation. Visual Information. Compared with the text information, the visual information is another significantly important feature in document representations. Typically, documents contain some visual signals to show the importance and priority of document segments. The visual information can be represented by image features and effectively utilized in document representations. For document-level visual features, the whole image can indicate the document layout, which is an essential feature for document image classification. For word-level visual features, styles such as bold, underline, and italic, are also significant hints for the sequence labeling tasks. Therefore, It is believed that combining the image features with traditional text representations can bring richer semantic representations to documents.

3.4.3 LayoutLM Architecture

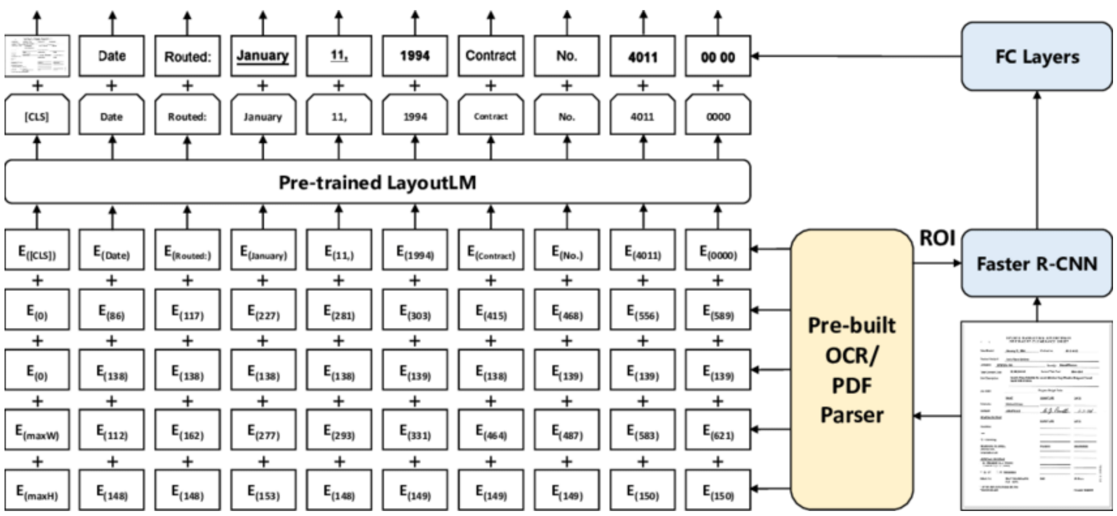

Date
Routed:
January
11,
1994
Contract
No.
4011
00 00
FC Layers
[CLS]
Date
Routed:
January
11,
1994
Contract
No.
4011
0000
Pre-trained LayoutLM
E([CLS]) E(Date) E(Routed:) E(January) E(11,) E(1994) E(Contract) E(No.) E(4011) E(0000)
E(0) E(86) E(117) E(227) E(281) E(303) E(415) E(468) E(556) E(589)
E(0) E(138) E(138) E(138) E(138) E(139) E(138) E(139) E(139) E(139)
E(maxW) E(112) E(162) E(277) E(293) E(331) E(464) E(487) E(583) E(621)
E(maxH) E(148) E(148) E(153) E(148) E(149) E(149) E(149) E(150) E(150)
ROI
Faster R-CNN
Pre-built OCR/ PDF Parser

Figure 3.2: LayoutLM Architecutre

To take advantage of existing pre-trained models and adapt to document image understanding tasks, BERT architecture is used as the backbone and add two new input embeddings: a 2-D position embedding and an image embedding. 2-D Position Embedding. Unlike the position embedding that models the word position in a sequence, 2-D position embedding aims to model the relative spatial position in a document. To repre- sent the spatial position of elements in scanned document images, we consider a document page as a coordinate system with the top- left origin. In this setting, the bounding box can be precisely defined by (x0 , y0, x 1 , y1), where (x0 , y0) corresponds to the position of the upper left in the bounding box, and (x1 , y 1) represents the position of the lower right. We add four position embedding layers with two embedding tables, where the embedding layers representing the same dimension share the same embedding table. This means that we look up the position embedding of x 0 and x1 in the embedding table X and lookup y0 and y1 in table Y . Image Embedding. To utilize the image feature of a document and align the image feature with the text, we add an image embedding layer to represent image features in language representation. In more detail, with the bounding box of each word from OCR results, we split the image into several pieces, and they have a one-to-one correspondence with the words. We generate the image region features with these pieces of images from the Faster R-CNN [12] model as the token image embeddings. For the [CLS] token, we also use the Faster R-CNN model to produce embeddings using the whole scanned document image as the Region of Interest (ROI) to benefit the downstream tasks which need the representation of the [CLS] token.

3.4.4 Embeddings

In deep learning, embeddings are a way of representing data in a lower-dimensional space. This is often used in natural language processing (NLP) tasks to represent words as vectors of a fixed length. The goal of using embeddings is to capture the semantic meaning of the words in the context of the task at hand, such as sentiment analysis or language translation.

Traditionally, NLP models would represent each word as a one-hot vector, where each word in the vocabulary is represented by a vector of all zeros except for a one in the position corresponding to the index of the word in the vocabulary. However, one-hot vectors are not effective for capturing the semantic meaning of words, since each word is treated as an independent entity. By contrast, embeddings represent words as vectors that capture the semantic relationships between words. For example, similar words such as "cat" and "dog" will have similar embeddings, while dissimilar words such as "cat" and "car" will have dissimilar embeddings.

Embeddings are learned during training using techniques such as backpropagation and stochastic gradient descent. During training, the model adjusts the embeddings such that they capture the semantic meaning of the words in the context of the task at hand. Once the embeddings are learned, they can be used to represent words in new inputs and fed into the model for inference.

3.4.5 Pre-training

Pre-training is a technique in deep learning that involves training a model on a large dataset before training it on a specific task. The idea is to use the large dataset to learn general features that can be applied to a wide range of tasks. The pre-training is typically unsupervised, meaning that the model learns to extract features from the input data without any explicit labels or targets.

One popular pre-training approach in NLP is called language modeling. The model is trained to predict the next word in a sequence of words, given the previous words in the sequence. This task requires the model to capture the semantic relationships between words and the overall structure of the language. Once the model is pre-trained on a large corpus of text, the learned features can be applied to a range of downstream tasks, such as sentiment analysis or text classification.

3.4.6 Fine-tuning

Fine-tuning is a technique in deep learning that involves taking a pre-trained model and updating the weights of some of the layers to adapt it to a new task. The idea is to leverage the general knowledge learned during pre-training and transfer learning to quickly adapt the model to the specifics of the new task.

For example, in the sentiment analysis task mentioned earlier, the pre-trained language model can be fine-tuned by adding a new output layer and updating the weights of the existing layers to optimize the model for the sentiment analysis task. The fine-tuning process typically involves training the model on a smaller dataset than the pre-training dataset, which allows for faster training times and better performance on the new task.

3.4.7 Transfer Learning

Transfer learning is a technique in deep learning that involves transferring knowledge from one task to another. The idea is to use a model that has been pre-trained on a large dataset and fine-tuned for a specific task as a starting point for training a new model on a related task. This can speed up the training process and improve the performance of the model on the new task.

For example, in NLP, a model that has been pre-trained on a large corpus of text using language modeling can be used as a starting point for training a sentiment analysis model. The pre-trained model has already learned general features that are useful for understanding the semantic relationships between words, which can be applied to the sentiment analysis task.

3.4.8 Sequence Labeling

Sequence labeling is a type of task in natural language processing (NLP) that involves assigning labels to each element of a sequence. The sequence can be a sequence of words, characters, or other tokens. The goal of sequence labeling is to identify the structure and meaning of the sequence by assigning labels to each element.One common type of sequence labeling task in NLP is named entity recognition (NER), which involves identifying and categorizing entities in a text, such as people, organizations, and locations. In this task, the sequence is a sequence of words, and the labels indicate whether each word is part of an entity and what type of entity it is.Another type of sequence labeling task is part-of-speech (POS) tagging, which involves labeling each word in a sentence with its part of speech, such as noun, verb, or adjective. In this task, the sequence is a sequence of words, and the labels indicate the part of speech of each word.

Sequence labeling can be performed using various machine learning techniques, including rule-based systems, statistical models, and deep learning models. In recent years, deep learning models such as recurrent neural networks (RNNs) and transformers have achieved state-of-the-art performance on many sequence labeling tasks.To train a deep learning model for sequence labeling, a labeled dataset is required. The labeled dataset consists of a sequence of inputs and corresponding labels. The model is trained using backpropagation and stochastic gradient descent to minimize a loss function, which measures the difference between the predicted labels and the true labels.During inference, the trained model takes a sequence of inputs as input and outputs a sequence of predicted labels. The predicted labels can be used to identify the structure and meaning of the sequence. For example, in NER, the predicted labels can be used to identify the entities in the text and their types.

3.4.9 Cross-Entropy Loss

Cross-entropy loss is a commonly used loss function in deep learning, particularly in classification tasks. It measures the difference between the predicted probability distribution and the true probability distribution of the labels. The goal of the loss function is to minimize this difference, which indicates how well the model is predicting the correct labels.

The formula for cross-entropy loss is as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

where L is the cross-entropy loss N is the number of samples in the batch C is the number of classes y_{ij} is the true label for sample i and class j (1 if the sample belongs to the class, 0 otherwise) $\hat{y}_{ij}$ is the predicted probability for sample i and class j The formula can be interpreted as follows: for each sample, the loss is the sum of the negative logarithm of the predicted probability of the correct label. The negative sign is used to ensure that the loss is always positive and that the model is penalized for making incorrect predictions.

In practice, the cross-entropy loss is often combined with other techniques, such as regularization and optimization, to improve the performance of the model. It is also used in conjunction with various deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), to achieve state-of-the-art performance on a wide range of tasks.

3.4.10 Tokenization

Tokenization is the process of breaking down a text into smaller units called tokens. Tokens can be words, phrases, symbols, or other units, depending on the specific task and the chosen tokenizer. The goal of tokenization is to make it easier to analyze and process text data, by breaking it down into smaller, more manageable units.

The formula for tokenization can be written as follows:

$$\text{Tokenization}(text) = [t_1, t_2, ..., t_n]$$

where:

Tokenization$(text)$ is the tokenized representation of the input text $[t_1, t_2, ..., t_n]$ is a sequence of tokens, where n is the number of tokens in the text

Tokenization is a fundamental step in many NLP tasks, such as text classification, sentiment analysis, and machine translation. The choice of tokenizer can have a significant impact on the performance of the model, as different tokenizers may break down the text in different ways, and may capture different levels of granularity in the text data. There are many different tokenization algorithms and tools available in NLP, ranging from simple rule-based tokenizers to complex neural network-based tokenizers. Some popular tokenization algorithms include the Penn Treebank tokenizer, the WordPiece tokenizer, and the Byte Pair Encoding (BPE) tokenizer.

3.4.11 Classification Metrices

When evaluating the performance of a machine learning model for a classification task, it is important to use appropriate metrics to measure its accuracy. In this section, we will discuss some of the most commonly used metrics for classification tasks.

Accuracy

Accuracy is the simplest and most intuitive metric for measuring the performance of a classification model. It measures the proportion of correctly classified samples out of the total number of samples in the dataset. The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{Number of Correctly Classified Samples}}{\text{Total Number of Samples}}$$

While accuracy is a useful metric, it can be misleading in some cases. For example, if the dataset is imbalanced, with many more samples in one class than in another, a model that always predicts the majority class will have high accuracy, but will not be useful for practical purposes.

Precision and Recall

Precision and recall are two complementary metrics that provide a more detailed picture of the model's performance. Precision measures the proportion of true positives (correctly classified samples) out of all positive predictions (samples predicted to be in a certain class):

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall measures the proportion of true positives out of all actual positive samples:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

In general, precision and recall are trade-offs: increasing one metric often results in a decrease in the other. F1 score is a commonly used metric that combines precision and recall into a single score:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion Matrix

A confusion matrix is a table that summarizes the classification results of a model. It shows the number of true positives, true negatives, false positives, and false negatives for each class. The table can be used to calculate other metrics such as precision, recall, and F1 score.

Table 3.1: **Confusion Matrix**

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

4. Methodology

4.1 Data Collection

The data collection process is a critical aspect of any machine learning project, and it requires a careful consideration of the dataset's size, diversity, and quality. In our project, we collected ATM receipts from different ATM booths in the city and tax invoices from various vendors we know. We selected these sources because they represent common types of documents that users might need to extract information from and these were the most easy source we know to collect the required data for our project, These are relevant because user might want to extract information from documents such as transaction records, expense reports, and receipts for tax purposes

To augment our dataset and increase its diversity, we also generated synthetic tax invoices using an Excel-based template that closely resembles real tax invoices. This approach enabled us to add more samples to our dataset without incurring additional costs or relying solely on a limited set of available documents. By generating synthetic data, we can also control the variability and complexity of the data, which can be useful for testing the robustness and generalization capabilities of our system.

In total, we collected a dataset of around 500 documents for training and testing. We ensured that our dataset is representative of real-world documents by including documents with varying layouts, fonts, sizes, and other visual characteristics. To evaluate the performance of our system, we randomly split the dataset into training, validation, and test sets. We also ensured that the annotation process is consistent and accurate by providing clear annotation guidelines and regular quality checks.

To provide more transparency and reproducibility of our results, we have included a sample of each dataset in the appendix of our report. These samples illustrate the diversity and complexity of the data we used for training and testing our system. They also provide a concrete example of the types of documents that our system can handle, such as ATM receipts with different layouts, tax invoices with varying item descriptions, and synthetic invoices with controlled variables. Overall, the data collection process is a crucial component of our project, and we believe that our approach to collecting and augmenting the dataset is a key factor in achieving accurate and robust information extraction from scanned documents.

4.2 Data Annotation

In order to train a deep learning model to accurately extract information from documents, it is necessary to first annotate the data. Annotation involves the process of labeling key fields in the documents, such as dates, merchant names, and total amounts. This is typically done using an annotation tool that allows the annotator to highlight the relevant text and assign it a specific label.

In our software platform, we utilized an annotation tool to annotate the collected documents for training the deep learning model. This involved creating a dataset of documents that had been labeled with the key fields that the model needed to learn to recognize. The annotation process was done by a team of trained annotators who were provided with clear guidelines and instructions for labeling the data.

Ensuring consistency and accuracy in the annotation process was a critical component of this project. To achieve this, we provided our annotators with a detailed set of guidelines and rules for labeling the data. These guidelines covered the specific labeling conventions and the criteria for determining what should be labeled as a date, merchant name, total amount, etc.

Additionally, we implemented regular quality checks to ensure that the annotation was being done correctly and consistently. This involved reviewing a sample of the annotated data on a regular basis to check for any errors or inconsistencies. If issues were identified, we would provide additional training to the annotators to ensure that they were following the guidelines correctly.

4.3 Pretrained LayoutLM Model

4.3.1 Pre-training Dataset

The performance of pre-trained models is largely determined by the scale and quality of datasets. Therefore, large-scale scanned document image dataset was used to pre-train the LayoutLM model. This model pre-trained on the IIT-CDIP Test Collection 1.0, which contains more than 6 million documents, with more than 11 million scanned document images. Moreover, each document has its corresponding text and metadata stored in XML files. The text is the content produced by applying OCR to document images. The metadata describes the properties of the document, such as the unique identity and document labels. Although the metadata contains erroneous and inconsistent tags, the scanned document images in this large-scale dataset was perfectly suitable for pre-training this model.

4.3.2 Pre-training Tasks

Pretraining objective refers to the specific task or objective that a deep learning model is trained on during the pretraining stage. The pretraining objective is typically different from the objective of the downstream task that the model will be used for, and is designed to help the model learn general features and representations that can be transferred to the downstream task.

The LayoutLM model proposes two pre-training tasks to improve document image understanding. The first task is the Masked Visual-language Model (MVLM), which is inspired by the masked language model. The MVLM is designed to learn language representations with the clues of 2-D position embeddings and text embeddings. During pre-training, some of the input tokens are randomly masked, but the corresponding 2-D position embeddings are kept. The model is then trained to predict the masked tokens given the contexts. This approach enables the model to understand language contexts and utilize the corresponding 2-D position information, bridging the gap between visual and language modalities.

The second pre-training task proposed is the Multi-label Document Classification (MDC) loss. This task is used for document image understanding and generates high-quality document-level representations. The IIT-CDIP Test Collection includes multiple tags for each document image, and the MDC loss is used to supervise the pre-training process. The model clusters knowledge from different domains and generates better document-level representations. However, since the MDC loss needs the label for each document image, it may not be used for larger datasets. The performance of MVLM and MVLM+MDC gives the better result. Overall, the pre-training tasks proposed in LayoutLM help to improve document image understanding by utilizing language contexts and position information, as well as generating high-quality document-level representations.

4.3.3 Pre-training Configuration

The LayoutLM model initializes its weights using the pre-trained BERT base model, which has a 12-layer Transformer with 768 hidden sizes and 12 attention heads. The model contains approximately 113 million parameters. For the LARGE setting, the model has a 24-layer Transformer with 1,024 hidden sizes and 16 attention heads, which is initialized using the pre-trained BERT LARGE model and contains around 343 million parameters. The model randomly masks 15% of input tokens during pre-training and replaces them with the [MASK] token 80% of the time, a random token 10% of the time, and an unchanged token 10% of the time. The corresponding token is then predicted with the cross-entropy loss.

Additionally, LayoutLM adds 2-D position embedding layers with four embedding representations (x0, y0, x1, y1), where (x0, y0) represents the position of the upper left in the bounding box, and (x1, y1) represents the position of the lower right. To account for variations in document layout due to different page sizes, the actual coordinates are scaled to a "virtual" coordinate with a value from 0 to 1,000. The Faster R-CNN model uses the ResNet-101 model as its backbone network, which is pre-trained on the Visual Genome dataset.

The LayoutLM model is trained on eight NVIDIA Tesla V100 32GB GPUs with a total batch size of 80. The Adam optimizer is used with an initial learning rate of 5e-5 and a linear decay learning rate schedule. The BASE model takes approximately 80 hours to finish one epoch on 11 million documents, while the LARGE model takes nearly 170 hours to complete one epoch. Overall, the LayoutLM model leverages the pre-trained BERT model, incorporates 2-D position embeddings, and uses the ResNet-101 model as its backbone network to improve document image understanding.

4.4 Finetuning

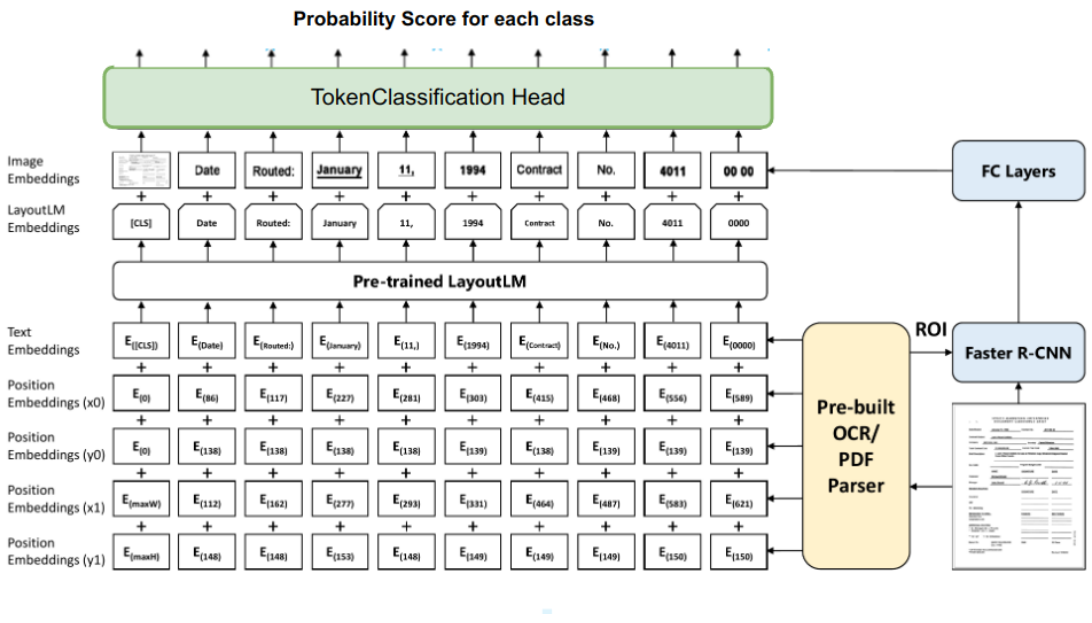


Figure 4.1: Fine-tuning LayoutLM for token classification

Fine-tuning is a process of adapting a pre-trained model on a specific task by training it on a small amount of task-specific data. In the case of LayoutLM, we finetune the model for token classification task, where the objective is to identify the tokens present in the document and classify them into different categories such as text, image, table, or equation.

To fine-tune the model, we add an additional layer called the token classification head to the pre-trained LayoutLM architecture. The token classification head is a fully connected layer that takes the contextual embeddings from LayoutLM as input and produces a probability distribution over the different token categories. The parameters of this layer are initialized using some form of initialization and are optimized during fine-tuning.

During fine-tuning, the model is trained on a small set of task-specific data, where the token labels are provided. The objective is to minimize the cross-entropy loss between the predicted and the actual labels. The fine-tuning process updates the parameters of the token classification head and fine-tunes the pre-trained LayoutLM parameters to adapt to the specific token classification task.

Fine-tuning LayoutLM for token classification enables us to leverage the pre-trained model's knowledge of language and document layout to improve the performance of the token classification task. By fine-tuning on the task-specific data, we can optimize the model's parameters to improve the classification performance on the specific task, and achieve state-of-the-art performance on various document analysis tasks.

Here is the high level observation of steps involved during Fine-tuning of LayoutLM for token classification task.

4.4.1 OCR

In this step, we pass our document through the Optical Character Recognition (OCR) software to convert the scanned images of the document into digital text. The OCR software that we used for this project is docTR, which is a deep learning-based OCR engine that supports a wide range of languages and fonts.

The docTR OCR engine uses a combination of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to recognize and transcribe the text from the input images. The CNNs are used to extract features from the input images, while the RNNs are used to model the sequence of characters in the text.

The output of the OCR step is a digital representation of the text in the document, which can be further processed and analyzed to extract useful information. However, the accuracy of the OCR output depends on the quality of the input images, the complexity of the document layout, and the language and font used in the document. We specifically need *Text* and *Bounding box* information the OCR. Hence the output of this step is Text in the digital form along with its bounding box. Four numbers are used to represent bounding box, representing top left and bottom right of the rectangle.

Overall, the OCR step is a crucial component of the document analysis pipeline, as it enables us to convert the scanned images of the document into digital text that can be analyzed and processed using machine learning techniques.

4.4.2 Labeling

Since LayoutLM finetuning for token classification uses a supervised learning technique, it requires a set of input sequences with corresponding labels that represent the ground truth for the target task.

In our project, we labeled the digital text output of the OCR step using a manual annotation tool. The annotation tool allowed us to label each token in the input text with the corresponding label that represents its semantic category. The labels used in our project included entity types such as Person, Organization, Location, Date, Time, and Money, as well as other types such as Number, Symbol, and Other.

To create the labeled training data, we first divided the digital text output into individual sentences and then labeled each token in each sentence using the annotation tool. We also validated the labeled data by performing inter-annotator agreement (IAA) analysis, which measures the level of agreement between different annotators for the same set of input sequences. The IAA analysis helped us ensure the consistency and accuracy of the labeled data.

4.4.3 Tokenization

Tokenization involves converting the input text into a sequence of tokens that can be processed by the model. Tokenization is the process of breaking down the input text into smaller units, such as words or subwords, that can be represented as discrete units of input for the model.

In our project, we used the WordPiece tokenization algorithm to tokenize the input text. The WordPiece algorithm is a popular subword tokenization algorithm that can handle out-of-vocabulary words by breaking them down into smaller subwords. The WordPiece algorithm works by iteratively merging the most frequent pairs of adjacent subwords until a maximum vocabulary size is reached.

After tokenizing the input text, we also added special tokens to the beginning and end of each sequence to indicate the start and end of the input, as well as padding tokens to ensure that all sequences have the same length. We also created attention masks to indicate which tokens should be attended to by the model and which tokens should be ignored.

The tokenization step is critical in the LayoutLM finetuning process for token classification, as it converts the input text into a format that can be processed by the model. The choice of tokenization algorithm can have a significant impact on the performance of the model, and it is important to choose an algorithm that is appropriate for the target task and the characteristics of the input text. Additionally, it is important to carefully consider the use of special tokens and attention masks to ensure that the model is able to effectively process the input sequences.

4.4.4 Data Pre-Processing

This step involves preparing the labeled data for input into the model. In our project, this step involved several key tasks, including normalizing the document images, converting the labeled data into the appropriate format, and splitting the data into training, validation, and test sets.

First, we normalized the document images to a resolution of 1000 by 1000 pixels. This was done to ensure that all images had the same size, which is a requirement for input into the LayoutLM model. The image normalization process involved resizing and padding the images to ensure that they had a consistent aspect ratio and size.

Next, we converted the labeled data into the appropriate format for input into the model. This involved encoding each token in the input text as a numerical value, using the tokenization algorithm that was applied in the previous step. We also created a corresponding label for each token, representing its semantic category, which was encoded as a numerical value as well.

Finally, we split the labeled data into training, validation, and test sets. The training set was used to train the model, while the validation set was used to tune the hyperparameters of the model and prevent overfitting. The test set was used to evaluate the performance of the model on unseen data.

4.4.5 Calculating Embeddings

Embeddings calculated in this step include:

1D position embeddings: These embeddings represent the position of each token along the 1D sequence of tokens in the input text.

2D position embeddings: These embeddings represent the position of each token within the 2D image space of the document. Four different embeddings are used to represent the Left, Right, Top and Left position of the bounding rectangle.

Height and width embeddings: These embeddings represent the height and width of each token in the document image.

Word embeddings: These embeddings represent the semantic meaning of each token in the input sequence.

All of these embeddings are concatenated together to form a single vector for each token. This concatenated embedding vector captures both the spatial and semantic relationships between each token and is used as input to subsequent layers of the model for further processing and classification.

By incorporating these different types of embedding, the LayoutLM model is able to effectively capture the complex relationships between text and layout information in document images. The Position and Word Embedding Layer is a key step in this process, as it enables the model to represent both the spatial and semantic information in a unified manner. Moreover all these embedding are learnable and model update these embedding while training to get contextual embedding that represent some meaning in the context of document

4.4.6 Multiheaded Attention

After the Position and Word Embedding Layer, the next step in the LayoutLM finetuning process for token classification is the Multihead Attention layer. This layer is used to capture the dependencies between different tokens in the input sequence, taking into account both their spatial and semantic relationships.

Multihead Attention is a variant of the Attention mechanism, which is widely used in deep learning models for natural language processing tasks. Attention allows the model to selectively focus on different parts of the input sequence when making predictions, based on their relevance to the current task.

In the Multihead Attention layer of LayoutLM, the input sequence of token embeddings is transformed into a set of queries, keys, and values. These vectors are then used to compute a set of attention weights for each token, indicating how much importance should be given to each other token in the sequence when making predictions for that token.

The Multihead Attention layer uses multiple attention heads, each of which learns to capture different patterns of dependencies between tokens. The outputs of the different attention heads are concatenated and passed through a feedforward layer to produce the final representation for each token.

Overall, the Multihead Attention layer in LayoutLM allows the model to capture complex dependencies between tokens in the input sequence, taking into account both their spatial layout and semantic content. By using multiple attention heads, the model is able to learn different patterns of dependencies and capture a more diverse range of relationships between tokens.

The output of the Multihead Attention layer is a sequence of vectors that represent each token in the input sequence, enriched with information about the dependencies between tokens. These vectors are used as input to the subsequent layers of the model for further processing and classification.

4.4.7 Feed Forward Layer

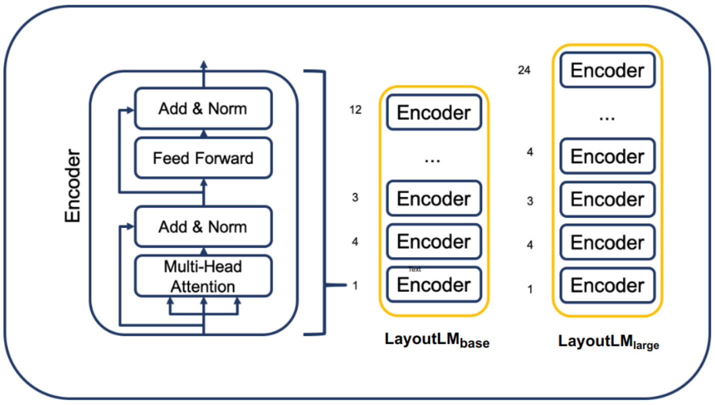


Figure 4.2: Pre-trained LayoutLM's Architecture

After the Multihead Attention layer in LayoutLM, the next step is the Feedforward layer. This layer is used to transform the output of the Multihead Attention layer into a higher-dimensional space, where it can be better separated and classified.

The Feedforward layer consists of two linear transformations, each followed by a nonlinear activation function such as the Rectified Linear Unit (ReLU). The first transformation maps the input vectors to a higher-dimensional space, while the second transformation maps the output of the first transformation back to the original dimensionality.

The purpose of the Feedforward layer is to allow the model to learn more complex and non-linear relationships between tokens in the input sequence. By mapping the input vectors to a higher-dimensional space, the model can capture more intricate interactions between tokens that may not be apparent in the original feature space.

The output of the Feedforward layer is a sequence of vectors that have been transformed to a higher-dimensional space and then back to the original dimensionality. These vectors represent each token in the input sequence and are enriched with additional information about their relationships with other tokens in the sequence.

The output of the Feedforward layer is then used as input to the final layer of the LayoutLM model for token classification, the Classification Head. By incorporating the Feedforward layer in the model, LayoutLM is able to capture more complex and subtle relationships between tokens, leading to improved performance on token classification tasks.

4.4.8 Token Classification

Token classification is the next step in the finetuning process for LayoutLM. It involves using the trained model to classify each token in a given input sequence into one or more predefined label, such as named Company, Address,Date etc..

For each token this layer will output the probability of falling into each labels in the training set. The labels with the highest probability for each token are then assigned as the predicted labels for that token. In cases where multiple labels are assigned to a single token, the most likely label or labels can be selected based on a variety of criteria, such as the highest probability, or the label that is most consistent with the context of the surrounding tokens.

4.4.9 Loss Calculation

During the finetuning process for LayoutLM, the Cross Entropy Loss function is used to measure the difference between the predicted label distribution and the true label distribution for each token in the training set. The Cross Entropy Loss is a commonly used loss function for classification tasks, and is defined as the negative log-likelihood of the true label given the predicted label distribution.

During training, the Cross Entropy Loss is calculated for each token in the training set, and the average loss over all tokens is used as a measure of the model's performance. The goal of the training process is to minimize this average loss by adjusting the parameters of the model.

To update the model parameters, gradient descent is used to compute the gradients of the Cross Entropy Loss with respect to each parameter. These gradients are then used to update the parameters in the direction that reduces the loss. This process is repeated for multiple epochs, until the model has converged to a set of parameters that minimize the loss on the training set.

Overall, the Cross Entropy Loss is a critical component of the fine-tuning process for LayoutLM, as it provides a measure of the model's performance on the task of token classification, and guides the update of the model's parameters during training. By minimizing the Cross Entropy Loss, LayoutLM can be trained to accurately classify tokens into predefined categories, and enable the extraction of structured information from unstructured text.

5. Architecture and Implementation

5.1 Design Implementation

5.1.1 Overview of System Architecture

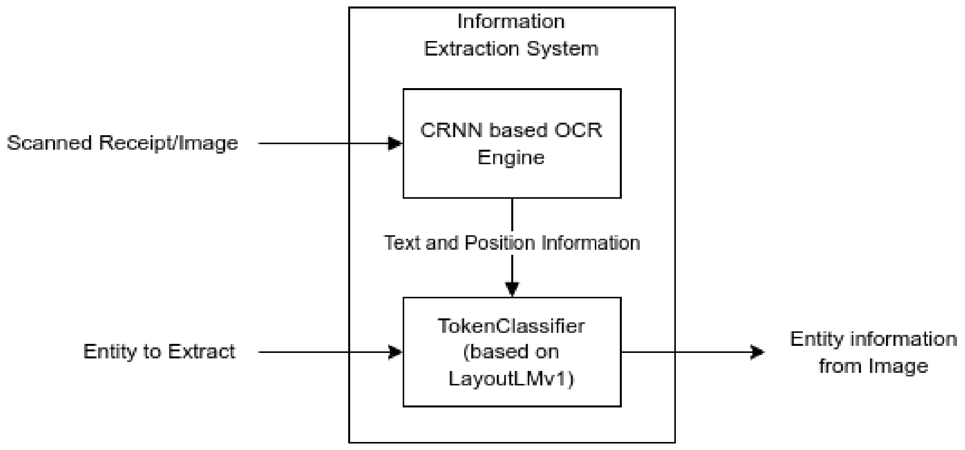
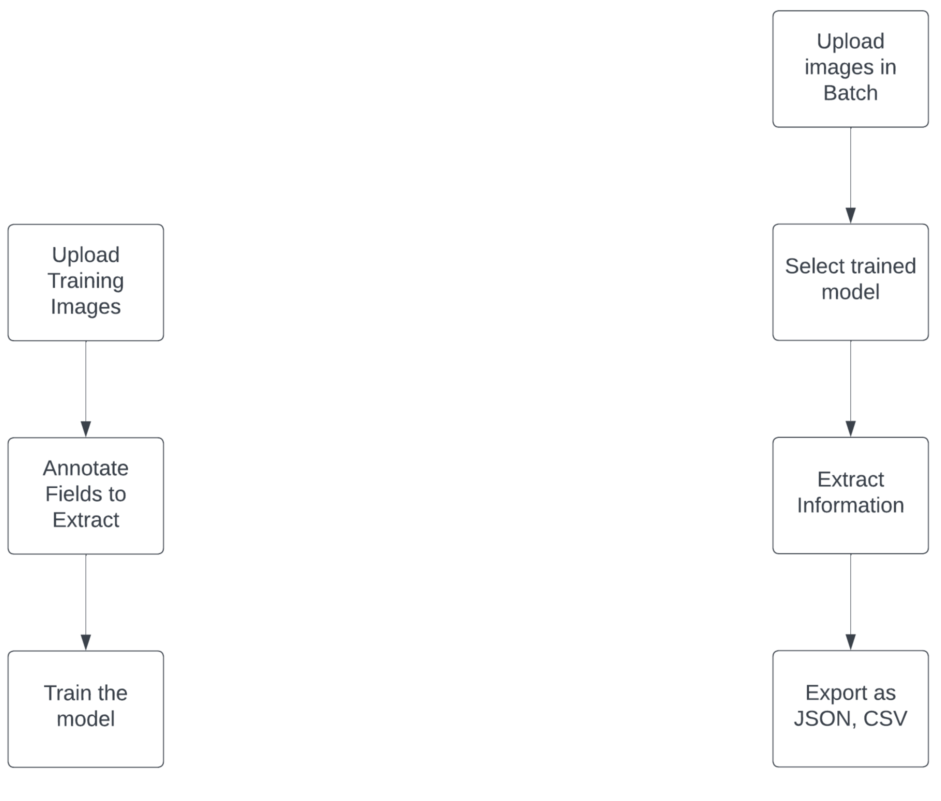

Information Extraction System
Scanned Receipt/Image
CRNN based OCR Engine
Text and Position Information
Entity to Extract
TokenClassifier (based on LayoutLMv1)
Entity information from Image

Figure 5.1: System Block Diagram

The system is composed of mainly two components. A CRNN based OCR engine, and the Token Classifier. The CRNN based OCR Engine takes the invoice images from the user and gives the text and position information to the TokenClassifier. The TokenClassifier then then takes those text and position information and the user input Entity to Extract and gives the output information.



(a) Training process (b) Inference process

Figure 5.2: Process flow

A user can upload images of invoices, annotate them and then train a model on the annotated images. The user can then use the model to extract information from the images.

If a model is already trained and it fits the user's requirements, then the user can upload images and then pick a suitable model. The model then takes the images as input and produces information that the user wants.

The user can register for a new account on the system using an email or a Google account. Once logged in to the system the user can upload receipts/invoices. The user can then annotate the uploaded receipts to label what the user wants to extract from the image. The user can then use the collection of annotated images to fine tune a pretrained model. Once the model is trained the user can then upload images and use the model to extract information from them.

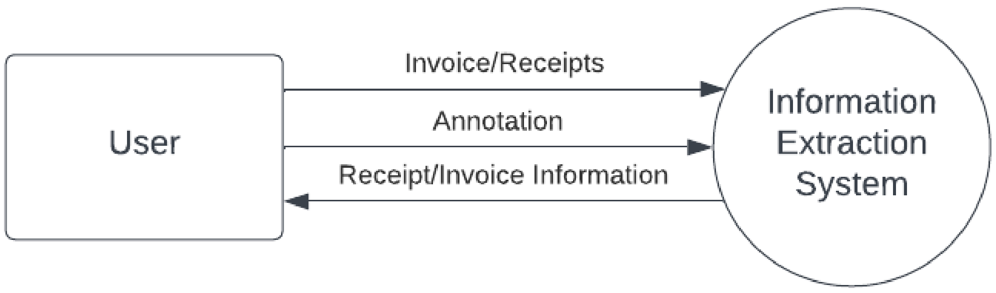


Figure 5.3: System Context Diagram

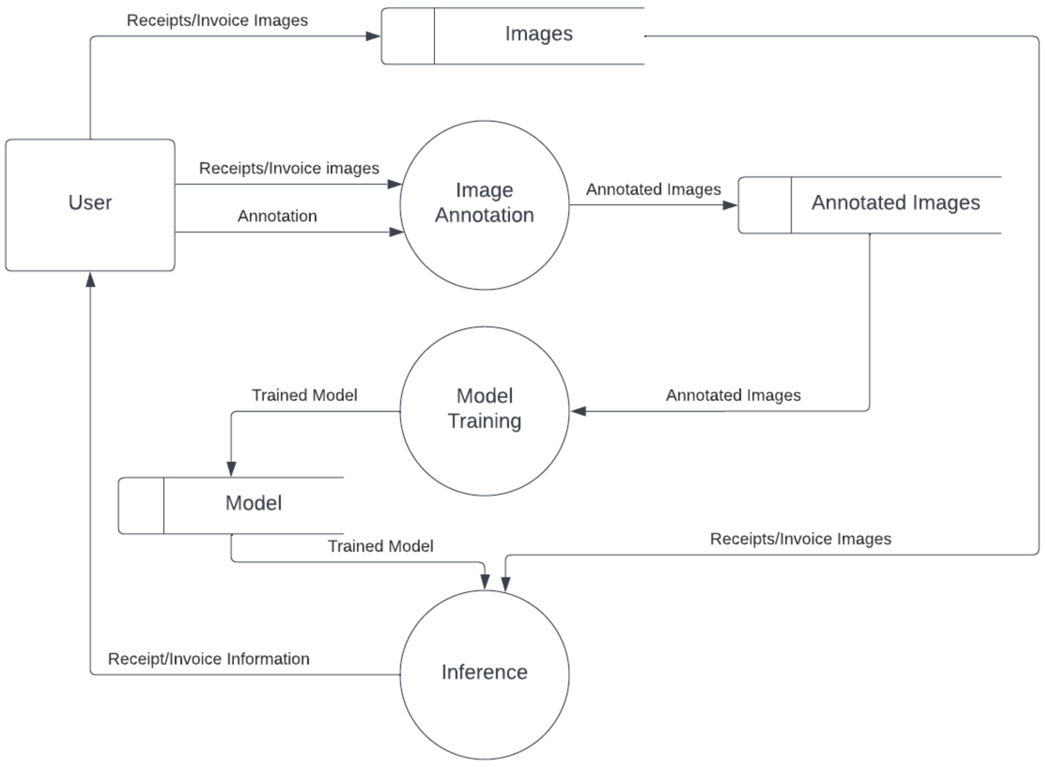


Figure 5.4: Data Flow Diagram

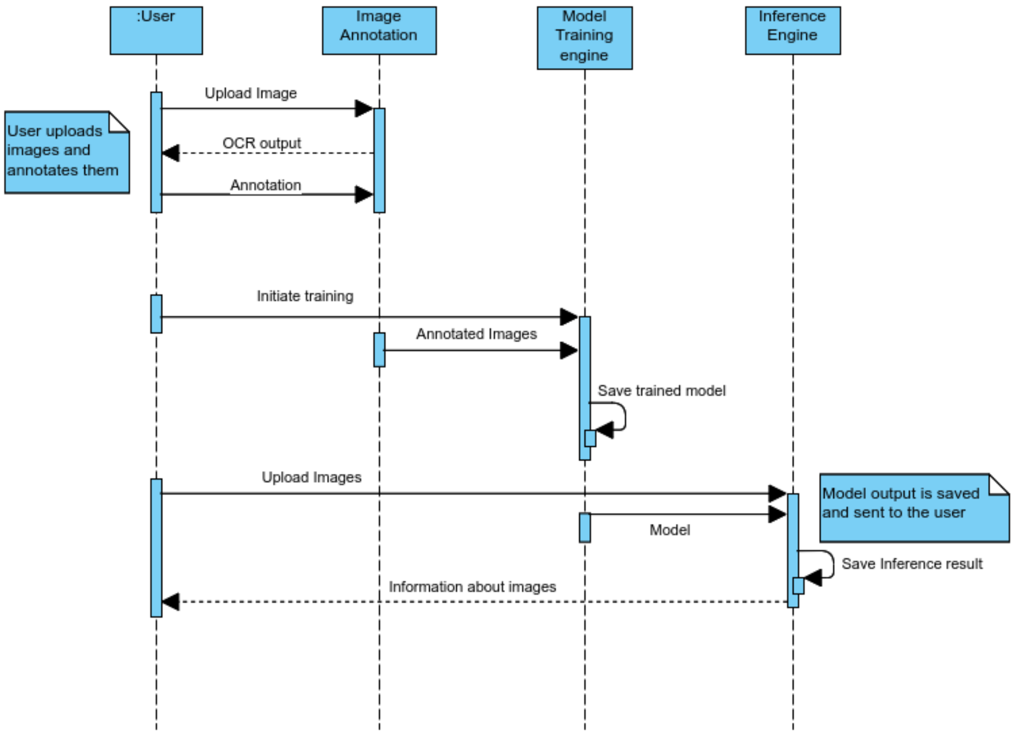


Figure 5.5: Sequence Diagram

5.2 Technologies used

1. **React**: React is a JavaScript framework for building user interfaces that offers reusable components, a virtual DOM for better performance, unidirectional data flow, a large and active community, and flexibility to work with other frameworks and tools. It is popular for building complex web applications and is a good choice for developers looking for a flexible, efficient, and well-supported framework.

2. **FastAPI**: FastAPI is a Python web framework for building high-performance APIs. It is fast, easy to use, Pythonic, automatically generates documentation, and uses the Pydantic library for data validation and serialization. FastAPI is designed to work asynchronously, which allows it to handle a large number of concurrent requests and provide high-performance responses.

3. **SQLAlchemy**: SQLAlchemy is an open-source SQL toolkit and Object-Relational Mapping (ORM) library for Python that provides a powerful SQL Expression Language, supports multiple databases, transactions, connection pooling, and an ORM for interacting with relational databases in an object-oriented way. Its consistent API works with a wide range of relational databases, making it easy to switch between them.

4. **Firebase**: Firebase is a mobile and web application development platform that provides a suite of tools and services to help developers build high-quality apps quickly. It offers real-time database, user authentication, cloud storage, hosting, and messaging services, among others. Firebase's features are designed to make it easy to build scalable and reliable apps with minimal effort. We use Firebase as an Auth server and also to sign in with a Google account directly.

5. **Pytorch**: PyTorch is an open-source machine learning framework that is known for its simplicity, flexibility, and ease of use. It is based on the Torch library and provides an efficient way to perform numerical computing and build deep learning models. This library is used for all machine learning components of the system.

6. **Google Colab**: Google Colab is a cloud-based data science work space which provide a powerful resource to perform machine learning operations. Hence, we have trained our models in this platform.

5.3 Deployment

The API for ReceiptScanner is deployed on Azure cloud platform behind an Nginx proxy, and it is using Docker containers. This setup allows for efficient containerization, scalability, and secure traffic routing to the application. A sidecar container is also present inside the docker environment. It's function is to continuously monitor the container registry (Docker Hub) for updates and pull the newest image and relaunch the container with new image. The container registry is also automatically updated. A Continuous Deployment service (GitHub actions) monitors the code repository for changes and then automatically builds a new image and pushes the image to the container registry.

The frontend of the system is hosted on Vercel which is a free cloud deployment service. Application code is hosted on a GitHub repository and a GitHub workflow is setup that monitors each update on the repository and then triggers a build on Vercel. If the build passes the most recent code change is automatically reflected on the publicly accessible domain.

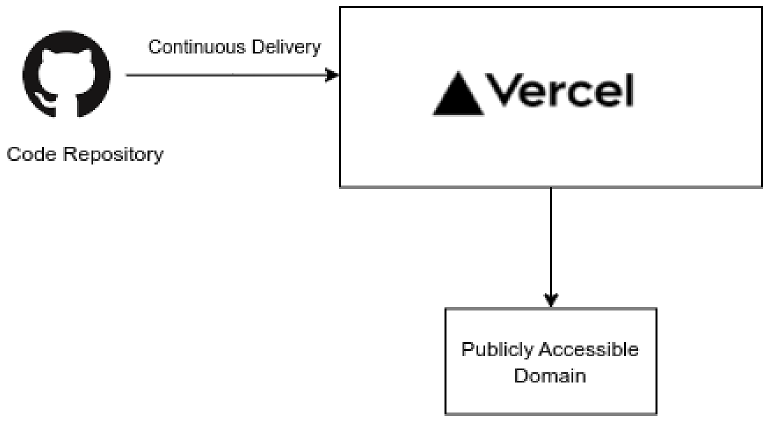


Figure 5.6: System Frontend Deployment Architecture

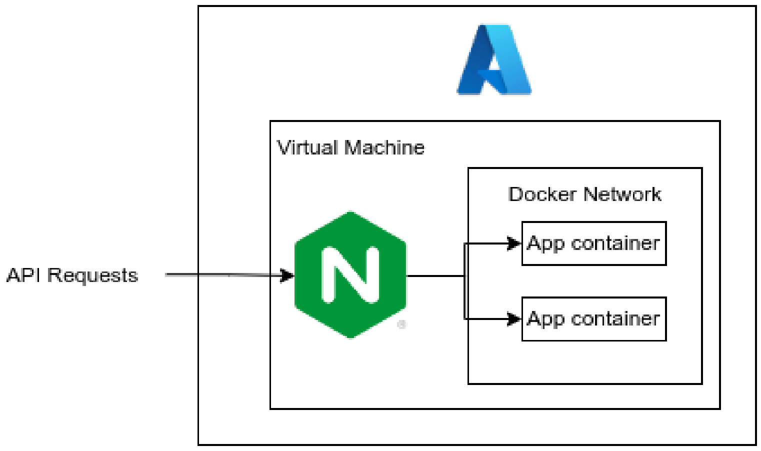


Figure 5.7: System API Deployment Architecture

5.4 Database Design

For interacting with the database, SQLAlchemy ORM is used. This offers the benefit of switching database engines (eg. MySQL to PostgreSQL) without substantial change in application code and also protects from various security threats. Additionally an ORM allows the programs to be written in a Object Oriented Manner.

The database schema for the system is as follows:

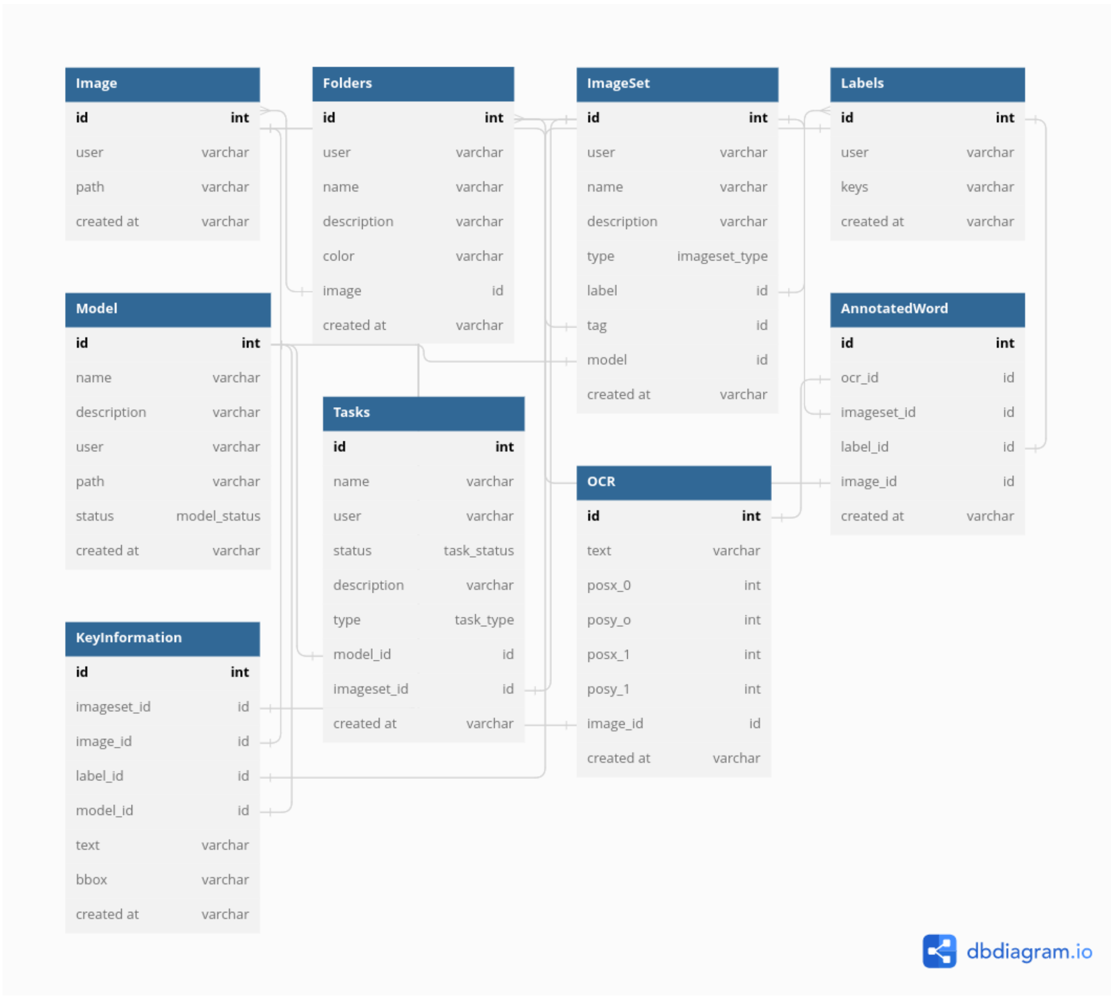


Image	
id	int
user	varchar
path	varchar
created at	varchar

Folders	
id	int
user	varchar
name	varchar
description	varchar
color	varchar
image	id
created at	varchar

ImageSet	
id	int
user	varchar
name	varchar
description	varchar
type	imageset_type
label	id
tag	id
model	id
created at	varchar

Labels	
id	int
user	varchar
keys	varchar
created at	varchar

Model	
id	int
name	varchar
description	varchar
user	varchar
path	varchar
status	model_status
created at	varchar

Tasks	
id	int
name	varchar
user	varchar
status	task_status
description	varchar
type	task_type
model_id	id
imageset_id	id
created at	varchar

AnnotatedWord	
id	int
ocr_id	id
imageset_id	id
label_id	id
image_id	id
created at	varchar

OCR	
id	int
text	varchar
posx_0	int
posy_o	int
posx_1	int
posy_1	int
image_id	id
created at	varchar

KeyInformation	
id	int
imageset_id	id
image_id	id
label_id	id
model_id	id
text	varchar
bbox	varchar
created at	varchar

Figure 5.8: Database Schema

1. **Image**: Image table represents an image that a user has uploaded. The user can upload many images. The records on this table represent an image resource and if the records are deleted the resource is also automatically deleted(via triggers).
2. **Folders**:Each Image must be associated with one and only one folder. One folder can have many images associated with it. Folders are a way of grouping images. Folders can be created by users as needed and deletion of a folder will also cause all associated images to be deleted.
3. **ImageSet**:A record of an ImageSet represents collection of images under multiple folders. The ImageSet can have a labels associated (if it is of type

training). A ImageSet can be annotated if it is of type training. If it is for inference, then it cannot be annotated.

4. **Labels**:When training a model, a the model must know what entities to extract, this is represented by the label record. Multiple labels can be associated with a single ImageSet as we can extract multiple things from a collection of images.

5. **Model**:A model represents record the machine learning model resource. A model is trained on a fixed ImageSet.

6. **OCR**:The OCR table stores the output of the OCR engine for each image. Even if a image is not associated with any ImageSet, if OCR is run on it then the output is stored on this table for faster access later.

7. **AnnotatedWord**:For a Image in an ImageSet for a specific label, a AnnotatedWord represents the data that has been input by the user. It references the OCR table for the actual word. The corrections for the words (in case of errors by OCR enging) are applied directly to the OCR table.

8. **KeyInformation**: KeyInformation is the table designated for storing the output of an inference session. It stores the model output for each image in a ImageSet, for a label.

9. **Task**:Task represents an ongoing task of training a model on a ImageSet, running the model on an ImageSet(inference) or applying OCR on images of an ImageSet(for faster access later). Depending on the type of task, different columns of this table have either Null or reference other tables.

5.5 Web Application

The web application is a complete product with some notable features.

1. When the user needs to annotate an image, the system not only provides the user tools for drawing bounding boxes on images but also a best guess OCR tokens drawn on the corresponding positions in the image. This allows the user to quickly select the correct tokens instead on manually drawing a bounding box and then typing the token on an input field. Instead, the user can quickly select the token related to a label. The user can still draw the bounding boxes manually and even correct the token that are not recognized correctly by the OCR engine.

2. Once an Image Collection has been annotated, the user can export the Image Collection. The exported files follow a simple structure and a simple extraction program can be used to convert it to any desired format. Thus, any user who only wants to use the system as an annotation tool can also get the annotation data in a desired form. The system can also import the Image Collection exported from the system. Thus, a user can send the Image Collection to different accounts if such a need arises.

3. After a model has processed an input Image Collection, the resulting data can be exported in a standardized format such as JSON or CSV. This allows users to easily access and use the model's predictions, regardless of the software or platforms they use. The exported data includes predicted labels and the corresponding system generate image ID and the original image file name.

6. Result and Analysis

In this section, we will present the result of our application and analyze its results. To evaluate the performance of our application for extracting key information scanned documents, we conducted experiments using these datasets: SROIE, ATM receipt, and our custom invoice dataset. The SROIE dataset contains 1000 images of receipts, the ATM receipt dataset contains 155 images, and our custom invoice dataset contains 220 images. Similarly, The SROIE dataset contains five labels: company, address, date, and total, and Other. The ATM receipt dataset contains six labels: bank, location, date, time, and anount and other. The custom invoice dataset contains eleven labels: Sender Company Name, Sender Address, Sender PAN number, Receiver Name, Receiver Address, Receiver PAN Number, Date, Invoice Number, taxable amount, total and Other. We split each dataset into a training set and a test set, with 80% of the data used for training and 20% for testing.

6.1 Training Loss

During the fine-tuning process of our model on all three datasets, we monitored the training loss to ensure that our model was effectively learning from the data. We used learning rate of 5e-5, Adam optimizer with a weight decay of 0.01, and a batch size of 8. We fine-tuned the LayoutLM architecture for 10 epochs, which allowed our model to learn from the data and improve its performance on the given task. We observed that the loss was consistently decreasing on the training and test set as shown in Figure 6.1.

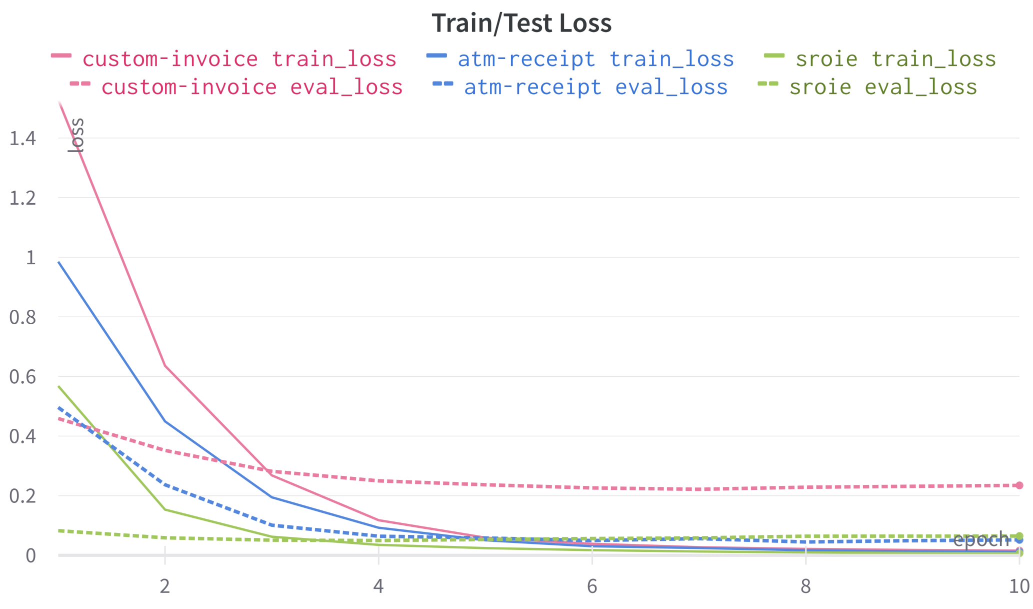


Figure 6.1: Loss during LayoutLM Finetuning (10 epochs)

The decreasing loss during the training phase indicates that our model was able to effectively learn from the data and improve its ability to accurately extract the desired information from the scanned documents. The overall decreasing loss in testing phase suggests that our model was able to effectively generalize to new data and capture the underlying patterns in the data.

Overall, the decreasing loss during the fine-tuning process of our model provides further evidence that our application for digitizing scanned documents using deep learning-based LayoutLM architecture is effective and reliable in accurately extracting important information from scanned receipts like documents in bulk.

6.2 Overall Evaluation Metrics

We used the deep learning based LayoutLM architecture at the core of our project to extract the desired fields from the scanned receipts in each dataset. For evaluation, we used precision, recall, and F1-score as the evaluation metrics.

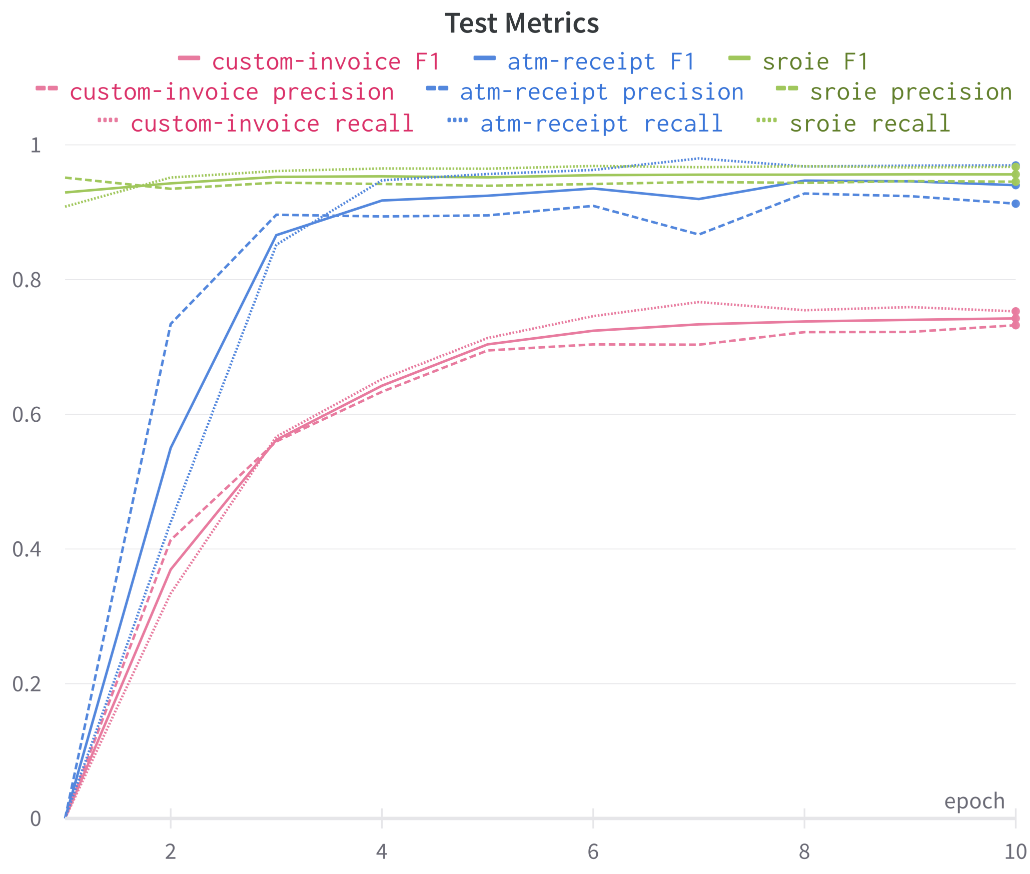


Figure 6.2: Evaluation Metrics after each epoch on test set

The graph 6.2 shows the precision, recall, and F1 score metrics for each epoch on the test set of three different datasets: SROIE, ATM receipt, and custom invoice datasets. The trend of the graph reveals that the model's performance improves for all three datasets as the epochs progress. However, the improvement is more substantial for the SROIE and ATM receipt datasets than for the custom invoice dataset. The final evaluation metrics on three datasets after training the model for 10 epochs is shown in table 6.1

Table 6.1: **Evaluation Metrics after 10 epochs**

Dataset	Total Documents	Labels	Precision	Recall	F1-score
SROIE	1000	5	0.95	0.97	0.96
ATM receipt	155	6	0.94	0.95	0.95
Custom invoice	220	11	0.70	0.80	0.75

Our experiments showed that our application achieved high precision, recall, and F1-score values on SRIOE dataset and ATM Receipt dataset. However, the model's performance on the custom invoice dataset is relatively low. Specifically, on the SROIE dataset, our system achieved an average precision of 0.95, recall of 0.96, and F1-score of 0.95. On the ATM receipt dataset, our system achieved an average precision of 0.94, recall of 0.95 and F1-score of 0.94. Finally, on our custom invoice dataset, our system achieved an average precision of 0.70, recall of 0.80, and F1-score of 0.75. The relatively low metrics on custom invoice dataset may be due to the complexity of the document layout and the relatively large number of different labels present in the dataset. Specifically, the custom invoice dataset contains 11 different labels for different types of entities, compared to only 5 labels in the SROIE dataset and 6 labels in the ATM receipt dataset.

These results demonstrate that our deep learning-based approach is effective in accurately identifying and extracting relevant fields from a variety of scanned documents, including receipts and invoices. The high precision and recall values obtained on each dataset as shown in table 6.1 indicate that our system is able to handle variations in layout and formatting, as well as different types of documents.

6.3 Label-wise Evaluation Metrics

We also calculated the precision, recall, and F1-score metrics for each label in the extracted information. Table 6.2,Table 6.3 and Table 6.4 shows the evaluation metrics for each label on the datasets used in our experiments, SROIE, ATM receipt, and Custom invoice respectively.

Table 6.2: **Label-wise Metrics on SROIE dataset (test) using LayoutLM Model**

Label	Precision	Recall	F1-score
Company	0.93	0.99	0.96
Address	0.98	0.98	0.98
Date	0.95	0.98	0.96
Total	0.67	0.71	0.69
Micro Avg	0.95	0.97	0.96
Macro Avg	0.88	0.91	0.90
Weighted Avg	0.95	0.97	0.96

In the SROIE dataset, the model achieved high precision, recall, and F1-scores for the labels **Company**, **Address**, and **Date**, with a weighted average F1-score of 0.96. However, the model's performance for the **Total** label was not as good as the other labels, with an F1-score of 0.69. This may be due to the fact that the **Total** label is more complex and requires the model to accurately identify and extract numerical values, which can be more challenging than identifying and

Table 6.3: **Label-wise Metrics on ATM receipt dataset (test) using LayoutLM Model**

Label	**Precision**	**Recall**	**F1-score**
Bank	0.97	0.92	0.94
Location	0.90	0.95	0.92
Date	0.91	1.00	0.95
Time	1.00	1.00	1.00
Amount	0.89	0.94	0.92
Micro Avg	0.94	0.95	0.94
Macro Avg	0.94	0.96	0.95
Weighted Avg	0.94	0.95	0.94

Table 6.4: **Label-wise Metrics on Custom Invoice dataset (test) using LayoutLM Model**

Label	**Precision**	**Recall**	**F1-score**
Sender Company Name	0.85	0.98	0.91
Sender Address	0.72	0.80	0.76
Sender PAN number	0.91	0.74	0.82
Receiver Name	0.61	0.65	0.63
Receiver Address	0.52	0.86	0.65
Receiver PAN Number	0.80	0.89	0.84
Invoice Number	0.85	0.83	0.84
Date	0.57	0.75	0.65
Taxable Amount	0.50	0.69	0.58
Total	0.97	0.75	0.85
Micro Avg	0.70	0.80	0.75
Macro Avg	0.73	0.79	0.75
Weighted Avg	0.72	0.80	0.75

extracting text values.

In the ATM receipt dataset, the model achieved high F1-scores for all labels, with the **Time** label having a perfect F1-score of 1.0. The macro and weighted average F1-scores were also high at 0.95. his may be due to the fact that the labels in this dataset are relatively simple and easy to identify, consisting mostly of short text strings and numerical values.

In the Custom Invoice dataset, the model achieved high F1-scores for the **Sender Company Name**, **Sender PAN number**, **Receiver PAN number**, **Invoice Number**, and **Total** labels, with F1-scores ranging from 0.84 to 0.91. However, the performance of the model was relatively poor for the **Receiver Name**, **Receiver Address**, **Date**, and **Taxable Amount** labels, with F1-scores ranging from 0.58 to 0.65. The micro and weighted average F1-scores were 0.75, indicating that the model performed moderately well overall. This may be due to the fact that the labels in this dataset are more varied and complex, and require the model to accurately identify and extract different types of information from the invoice.

In conclusion, the LayoutLM model showed promising performance for the label-wise metrics in all three datasets. However, there is room for improvement in some labels, particularly in the Custom Invoice dataset. Therefore, further optimization and fine-tuning of the model may be necessary to enhance its performance in specific label categories. It is important to consider these factors mentioned above when evaluating the performance of a model and determining its suitability for a particular task.

7. Epilogue

7.1 Conclusion

We have successfully developed a software platform for information extraction from scanned documents that uses deep learning-based LayoutLM architecture. The system allows users to upload and extract important information from scanned receipts and other similar documents in bulk, and it is highly effective in accurately recognizing and extracting relevant information.

Our system offers several benefits, including reducing manual data entry and improving data accuracy, which could significantly benefit businesses and individuals alike. Our project is an important step towards creating more efficient and accurate document processing systems. We believe that our system has the potential to transform the way businesses and individuals handle scanned documents and data extraction, and we recommend further exploration and research in this area to continue improving upon the current capabilities of our system.

7.2 Limitation and Future Enhancements

The system is not enforcing type of field checking. Eg. If the field is date, we are sure of the token is in one of the enumerable standard formats. In such a case, we can give preference to those tokens that conform to one of the standard format and get better accurate.

The system doesn't allow fine tuning a model which has already been fine tuned up to a point. The user might want to train a model further based on the the new data that has been acquired.

A limitation of the system is that all processes OCR, inference, Database, Image Storage, and model training are done on a single machine. These sub-systems are ideal candidates for micro-services. A micro-service architecture would allow the system to be highly scalable and would allow for independent development of micro-services.

Another enhancement could be class of document based rule engine. For eg. If a certain type of document, Department Store bills for example, always have address towards the right top corner then the system could get better results if this information was incorporated in some way.

BIBLIOGRAPHY

[1] S. Sarawagi, "Information extraction. foundations and trends r in databases," pp. 261–377, 2008.

[2] A. M. Falk Brauer Robert Rieger and W. M. Barczynski, "Enabling information extraction by inference of regular expressions from sample entities," pp. 1285–1294, 2011.

[3] F. Deckert, B. Seidler, M. Ebbecke, and M. Gillmann, "Table content understanding in smartfix. in 2011 international conference on document analysis and recognition," *IEEE*, pp. 488–492, 2011.

[4] A. R. D. Bertin Klein, "An adaptive system for document analysis and understanding. in reading and learning," pp. 166–186, 2019.

[5] C. SAGE, "Deep learning for information extraction from business documents," pp. 81–97, 2021.

[6] F. L. Rasmus Berg Palm Dirk Hovy and O. Winther, "End-to-end information extraction without token-level supervision," pp. 48–52, 2017.

[7] F. L. Rasmus Berg Palm and O. Winther, "Attend, copy, parse end-to-end information extraction from documents. in 2019 international conference on document analysis and recognition (icdar)," *IEEE*, pp. 329–336, 2019.

[8] Mindee, *Doctr: Document text recognition*, `https://github.com/mindee/doctr`, 2021.

[9] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 11 474–11 481.

[10] B. Shi, X. Bai, and C. Yao, "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*, vol. 39, no. 11, pp. 2298–2304, 2016.

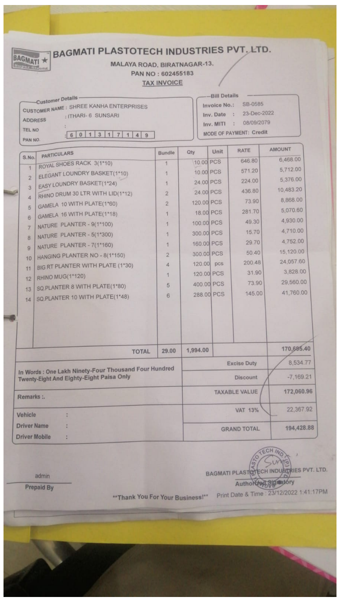
[11] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "Layoutlm: Pre-training of text and layout for document image understanding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1192–1200.

[12] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

8. Appendix

Appendix A

Sample Data



BAGMATI PLASTOTECH INDUSTRIES PVT. LTD.
MALAYA ROAD, BIRATNAGAR-13.
PAN NO : 602455183
TAX INVOICE

Customer Details
CUSTOMER NAME : SHREE KANHA ENTERPRISES
ADDRESS : ITHARI- 6 SUNSARI
TEL NO :
PAN NO. : 6 0 1 3 1 7 1 4 9

Bill Details
Invoice No.: SB-0585
Inv. Date : 23-Dec-2022
Inv. MITI : 08/09/2079
MODE OF PAYMENT: Credit

S.No.	PARTICULARS	Bundle	Qty	Unit	RATE	AMOUNT
1	ROYAL SHOES RACK 3(1*10)	1	10.00	PCS	646.80	6,468.00
2	ELEGANT LOUNDRY BASKET(1*10)	1	10.00	PCS	571.20	5,712.00
3	EASY LOUNDRY BASKET(1*24)	1	24.00	PCS	224.00	5,376.00
4	RHINO DRUM 30 LTR WITH LID(1*12)	2	24.00	PCS	436.80	10,483.20
5	GAMELA 10 WITH PLATE(1*60)	2	120.00	PCS	73.90	8,868.00
6	GAMELA 16 WITH PLATE(1*18)	1	18.00	PCS	281.70	5,070.60
7	NATURE PLANTER - 9(1*100)	1	100.00	PCS	49.30	4,930.00
8	NATURE PLANTER - 5(1*300)	1	300.00	PCS	15.70	4,710.00
9	NATURE PLANTER - 7(1*160)	1	160.00	PCS	29.70	4,752.00
10	HANGING PLANTER NO - 8(1*150)	2	300.00	PCS	50.40	15,120.00
11	BIG RT PLANTER WITH PLATE (1*30)	4	120.00	pcs	200.48	24,057.60
12	RHINO MUG(1*120)	1	120.00	PCS	31.90	3,828.00
13	SQ.PLANTER 8 WITH PLATE(1*80)	5	400.00	PCS	73.90	29,560.00
14	SQ.PLANTER 10 WITH PLATE(1*48)	6	288.00	PCS	145.00	41,760.00
	TOTAL	29.00	1,994.00			170,685.40

In Words : One Lakh Ninety-Four Thousand Four Hundred Twenty-Eight And Eighty-Eight Paisa Only

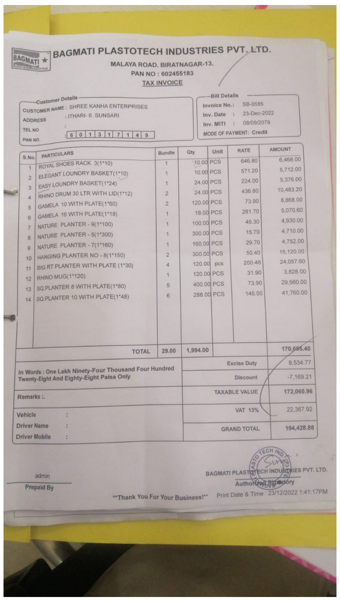
Excise Duty	8,534.77
Discount	-7,169.21
TAXABLE VALUE	172,060.96
VAT 13%	22,367.92
GRAND TOTAL	194,428.88

Remarks :.
Vehicle :
Driver Name :
Driver Mobile :

admin
Prepaid By

BAGMATI PLASTOTECH INDUSTRIES PVT. LTD.
Authorized Signatory
BAGMATI PLASTO TECH IND. (P.) LTD.

Thank You For Your Business! Print Date & Time : 23/12/2022 1:41:17PM



BAGMATI PLASTOTECH INDUSTRIES PVT. LTD.
MALAYA ROAD, BIRATNAGAR-13.
PAN NO : 602455183
TAX INVOICE

Customer Details
CUSTOMER NAME : SHREE KANHA ENTERPRISES
ADDRESS : ITHARI- 6 SUNSARI
TEL NO :
PAN NO. : 6 0 1 3 1 7 1 4 9

Bill Details
Invoice No.: SB-0585
Inv. Date : 23-Dec-2022
Inv. MITI : 08/09/2079
MODE OF PAYMENT: Credit

S.No.	PARTICULARS	Bundle	Qty	Unit	RATE	AMOUNT
1	ROYAL SHOES RACK 3(1*10)	1	10.00	PCS	646.80	6,468.00
2	ELEGANT LOUNDRY BASKET(1*10)	1	10.00	PCS	571.20	5,712.00
3	EASY LOUNDRY BASKET(1*24)	1	24.00	PCS	224.00	5,376.00
4	RHINO DRUM 30 LTR WITH LID(1*12)	2	24.00	PCS	436.80	10,483.20
5	GAMELA 10 WITH PLATE(1*60)	2	120.00	PCS	73.90	8,868.00
6	GAMELA 16 WITH PLATE(1*18)	1	18.00	PCS	281.70	5,070.60
7	NATURE PLANTER - 9(1*100)	1	100.00	PCS	49.30	4,930.00
8	NATURE PLANTER - 5(1*300)	1	300.00	PCS	15.70	4,710.00
9	NATURE PLANTER - 7(1*160)	1	160.00	PCS	29.70	4,752.00
10	HANGING PLANTER NO - 8(1*150)	2	300.00	PCS	50.40	15,120.00
11	BIG RT PLANTER WITH PLATE (1*30)	4	120.00	pcs	200.48	24,057.60
12	RHINO MUG(1*120)	1	120.00	PCS	31.90	3,828.00
13	SQ.PLANTER 8 WITH PLATE(1*80)	5	400.00	PCS	73.90	29,560.00
14	SQ.PLANTER 10 WITH PLATE(1*48)	6	288.00	PCS	145.00	41,760.00
	TOTAL	29.00	1,994.00			170,685.40

In Words : One Lakh Ninety-Four Thousand Four Hundred Twenty-Eight And Eighty-Eight Paisa Only

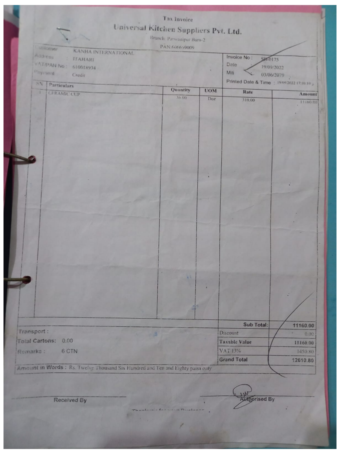
Excise Duty 8,534.77
Discount -7,169.21
TAXABLE VALUE 172,060.96
VAT 13% 22,367.92
GRAND TOTAL 194,428.88

Remarks :.
Vehicle :
Driver Name :
Driver Mobile :

admin
Prepaid By

BAGMATI PLASTOTECH INDUSTRIES PVT. LTD.
Authorized Signatory

Thank You For Your Business! Print Date & Time : 23/12/2022 1:41:17PM



Tax Invoice
Universal Kitchen Suppliers Pvt. Ltd.
Branch: Parwanipur Bara-2
PAN:606669009

Customer : KANHA INTERNATIONAL
Address : ITAHARI
VAT/PAN No : 610018934
Payment : Credit

Invoice No : SB-0175
Date : 19/09/2022
Miti : 03/06/2079
Printed Date & Time : 19/09/2022 17:10:10

SN	Particulars	Quantity	UOM	Rate	Amount
1	CERAMIC CUP	36.00	Doz	310.00	11160.00

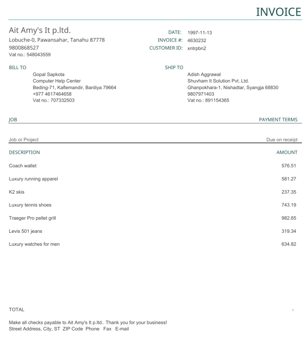
Transport :
Total Cartons: 0.00
Remarks : 6 CTN

Sub Total:	11160.00
Discount	0.00
Taxable Value	11160.00
VAT 13%	1450.80
Grand Total	12610.80

Amount in Words : Rs. Twelve Thousand Six Hundred and Ten and Eighty paisa only

Received By
Authorised By

Figure 8.1: Collected tax invoices



INVOICE

Ait Amy's It p.ltd.
Lobuche-0, Pawansahar, Tanahu 87778
9800868527
Vat no.: 548043559

DATE: 1997-11-13
INVOICE #: 4630232
CUSTOMER ID: xntrpbn2

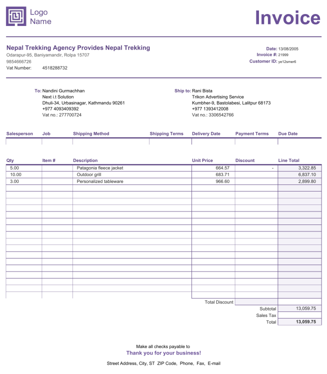
BILL TO
Gopal Sapkota
Computer Help Center
Beding-71, Kaflemandir, Bardiya 79664
+977 4617464658
Vat no.: 707332503

SHIP TO
Adish Aggrawal
Shuvham It Solution Pvt. Ltd.
Ghanpokhara-1, Nishadtar, Syangja 68830
9807971403
Vat no.: 891154365

JOB	PAYMENT TERMS
Job or Project	Due on receipt

DESCRIPTION	AMOUNT
Coach wallet	576.51
Luxury running apparel	581.27
K2 skis	237.35
Luxury tennis shoes	743.19
Traeger Pro pellet grill	982.65
Levis 501 jeans	319.34
Luxury watches for men	634.82
TOTAL	-

Make all checks payable to Ait Amy's It p.ltd.. Thank you for your business!
Street Address, City, ST ZIP Code Phone Fax E-mail



Logo Name

Invoice

Nepal Trekking Agency Provides Nepal Trekking
Odarapur-95, Baniyamandir, Rolpa 15707
9854666726
Vat Number: 4518288732

Date: 13/08/2005
Invoice #: 21999
Customer ID: ye12smer6

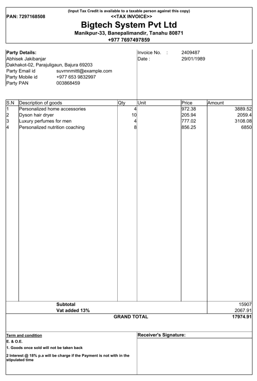
To: Nandini Gurmachhan
Next i.t Solution
Dhuli-34, Urbasinagar, Kathmandu 90261
+977 4093409392
Vat no.: 277700724

Ship to: Rani Bista
Trikon Advertising Service
Kumbher-9, Bastolabesi, Lalitpur 68173
+977 1393412008
Vat no.: 3306542766

Salesperson	Job	Shipping Method	Shipping Terms	Delivery Date	Payment Terms	Due Date

Qty	Item #	Description	Unit Price	Discount	Line Total
5.00		Patagonia fleece jacket	664.57	-	3,322.85
10.00		Outdoor grill	683.71		6,837.10
3.00		Personalized tableware	966.60		2,899.80
				Total Discount	
				Subtotal	13,059.75
				Sales Tax	
				Total	13,059.75

Make all checks payable to
Thank you for your business!
Street Address, City, ST ZIP Code, Phone, Fax, E-mail



PAN: 7297168508
(Input Tax Credit is available to a taxable person against this copy)
<<TAX INVOICE>>
Bigtech System Pvt Ltd
Manikpur-33, Banepalimandir, Tanahu 80871
+977 7697497859

Party Details:
Abhisek Jakibanjar
Dakhakot-02, Parajuligaun, Bajura 69203
Party Email id suvmnmittl@example.com
Party Mobile id +977 653 9832997
Party PAN 003868459

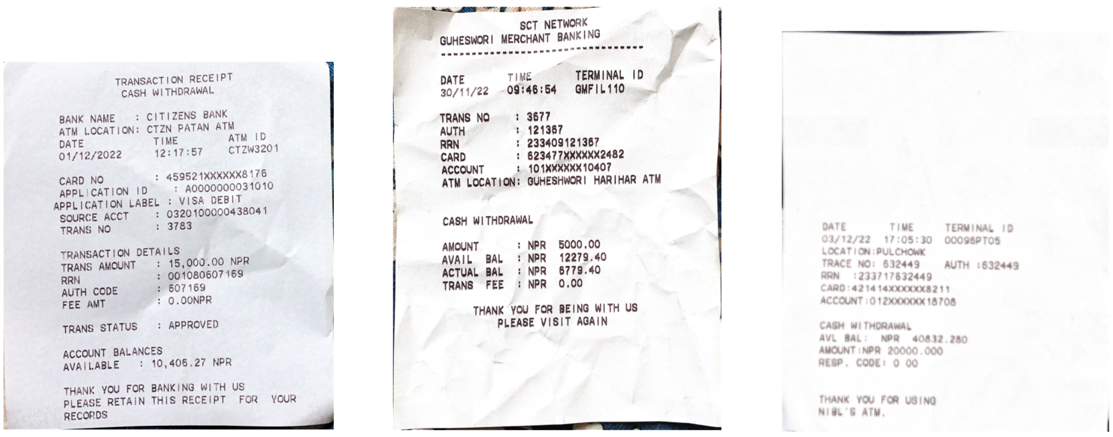
Invoice No. : 2409487
Date : 29/01/1989

S.N	Description of goods	Qty	Unit	Price	Amount
1	Personalized home accessories	4		972.38	3889.52
2	Dyson hair dryer	10		205.94	2059.4
3	Luxury perfumes for men	4		777.02	3108.08
4	Personalized nutrition coaching	8		856.25	6850
	Subtotal				15907
	Vat added 13%				2067.91
	GRAND TOTAL				17974.91

Term and condition
E. & O.E.
1. Goods once sold will not be taken back
2 Interest @ 18% p.a will be charge if the Payment is not with in the stipulated time

Receiver's Signature:

Figure 8.2: Generated invoices



```
TRANSACTION RECEIPT
CASH WITHDRAWAL

BANK NAME    : CITIZENS BANK
ATM LOCATION: CTZN PATAN ATM
DATE              TIME          ATM ID
01/12/2022        12:17:57      CTZW3201

CARD NO              : 459521XXXXXX8176
APPLICATION ID          : A0000000031010
APPLICATION LABEL : VISA DEBIT
SOURCE ACCT          : 0320100000438041
TRANS NO             : 3783

TRANSACTION DETAILS
TRANS AMOUNT         : 15,000.00 NPR
RRN                  : 001080607169
AUTH CODE            : 607169
FEE AMT              : 0.00NPR

TRANS STATUS         : APPROVED

ACCOUNT BALANCES
AVAILABLE     : 10,406.27 NPR

THANK YOU FOR BANKING WITH US
PLEASE RETAIN THIS RECEIPT  FOR  YOUR
RECORDS
```

```
              SCT NETWORK
GUHESWORI MERCHANT BANKING
------------------------------------

DATE       TIME       TERMINAL ID
30/11/22   09:46:54   GMFIL110

TRANS NO        : 3677
AUTH            : 121367
RRN             : 233409121367
CARD            : 623477XXXXXX2482
ACCOUNT         : 101XXXXXX10407
ATM LOCATION: GUHESHWORI HARIHAR ATM

CASH WITHDRAWAL

AMOUNT          : NPR  5000.00
AVAIL  BAL      : NPR  12279.40
ACTUAL BAL      : NPR  6779.40
TRANS  FEE      : NPR  0.00

     THANK YOU FOR BEING WITH US
         PLEASE VISIT AGAIN
```

```
DATE        TIME       TERMINAL ID
03/12/22  17:05:30  00096PT05
LOCATION:PULCHOWK
TRACE NO: 632449     AUTH :632449
RRN  :233717632449
CARD:421414XXXXXX8211
ACCOUNT:012XXXXXX18708

CASH WITHDRAWAL
AVL BAL:  NPR  40832.280
AMOUNT:NPR 20000.000
RESP. CODE: 0 00

THANK YOU FOR USING
NIBL'S ATM.
```

Figure 8.3: Collected ATM receipts

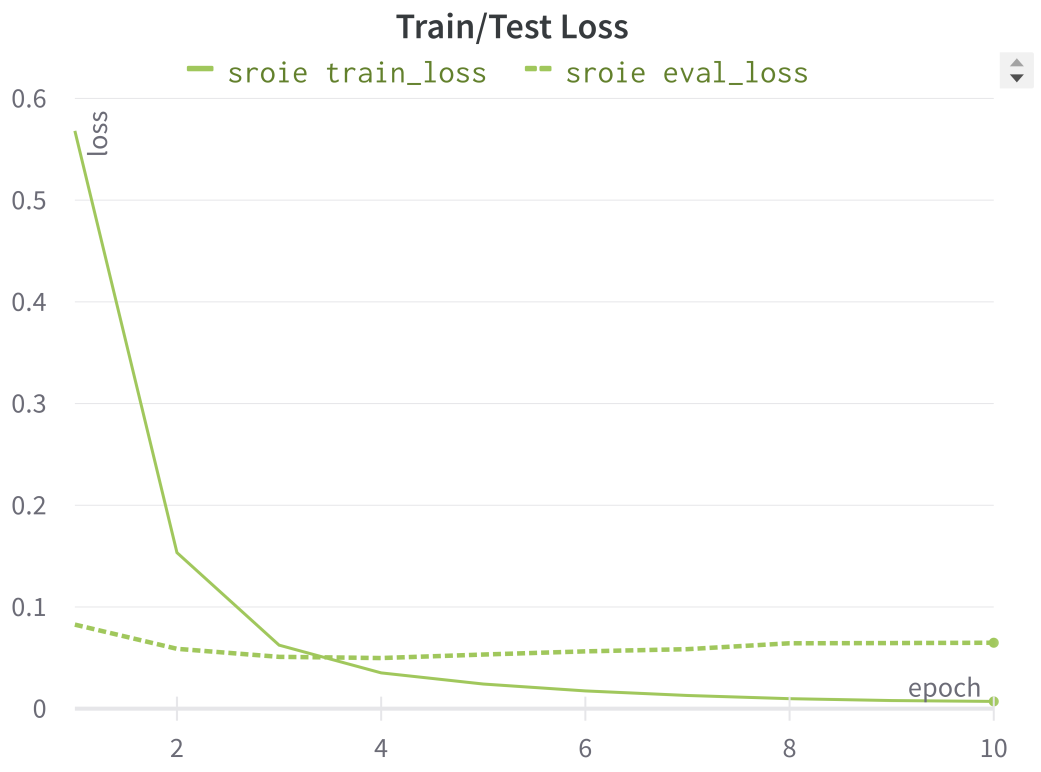


Figure 8.4: Train/Test loss for SRIOE test dataset

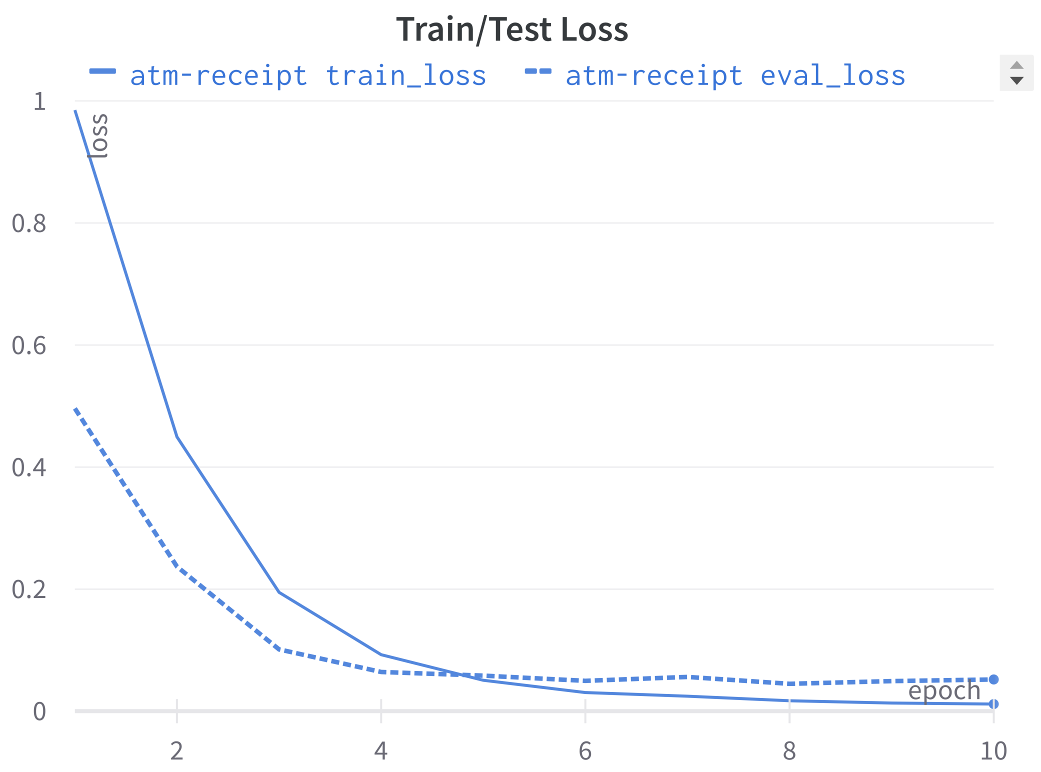


Figure 8.5: Train/Test loss for ATM Receipt test dataset

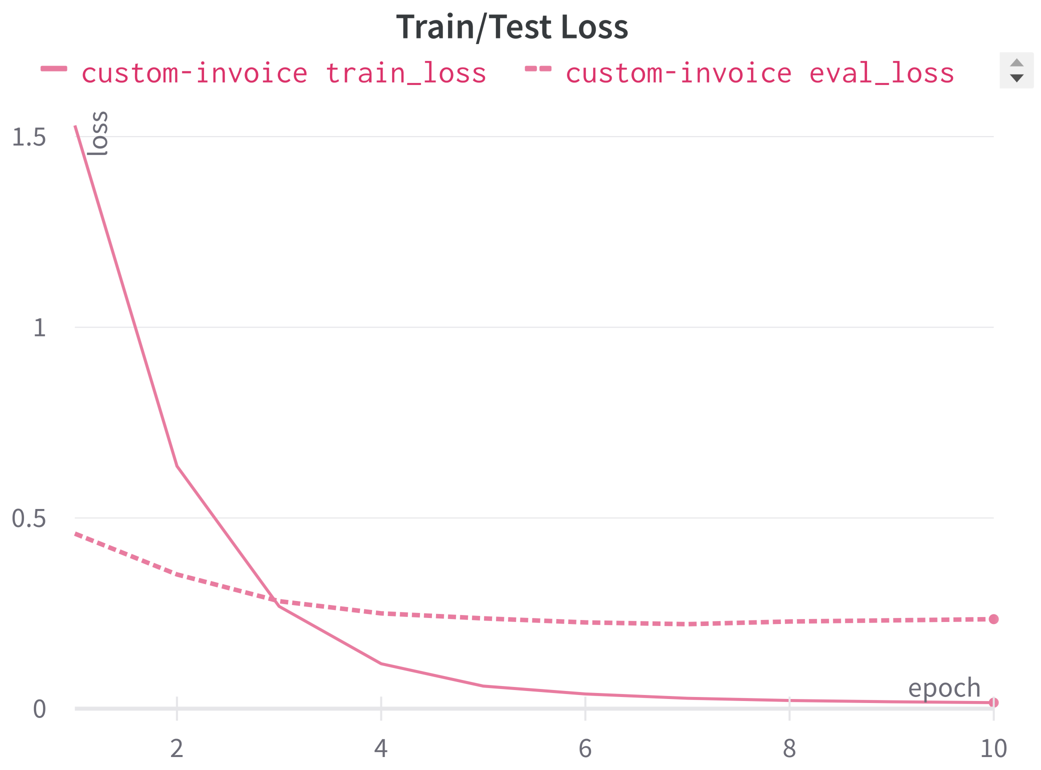


Figure 8.6: Train/Test loss for Custom invoice test dataset

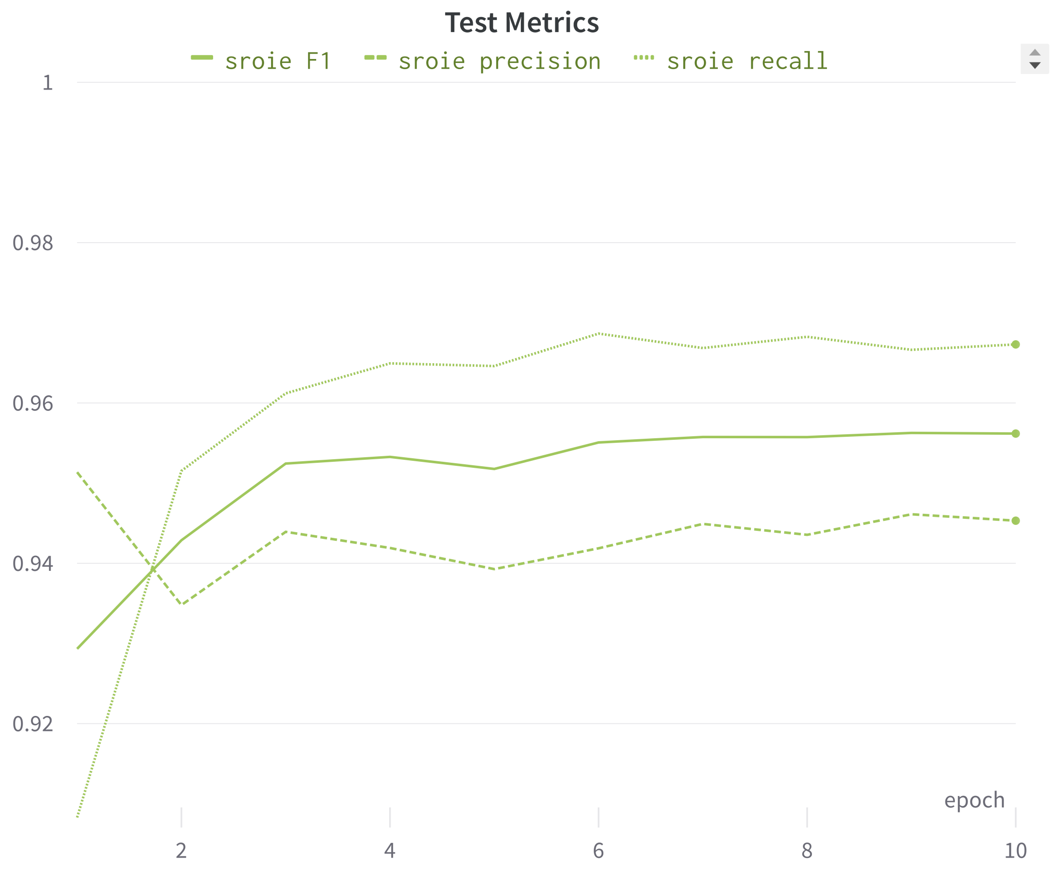


Figure 8.7: Evaluation Metrics for SRIOE dataset

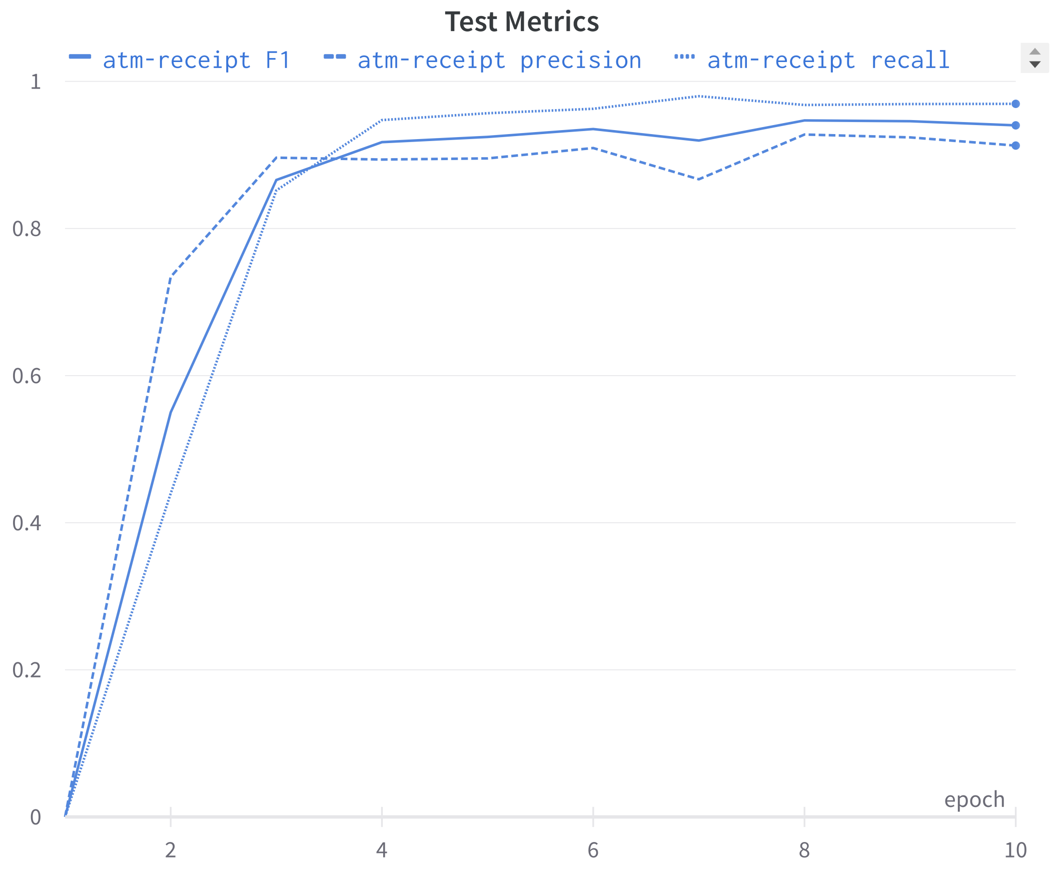


Figure 8.8: Evaluation Metrics for ATM Receipt dataset

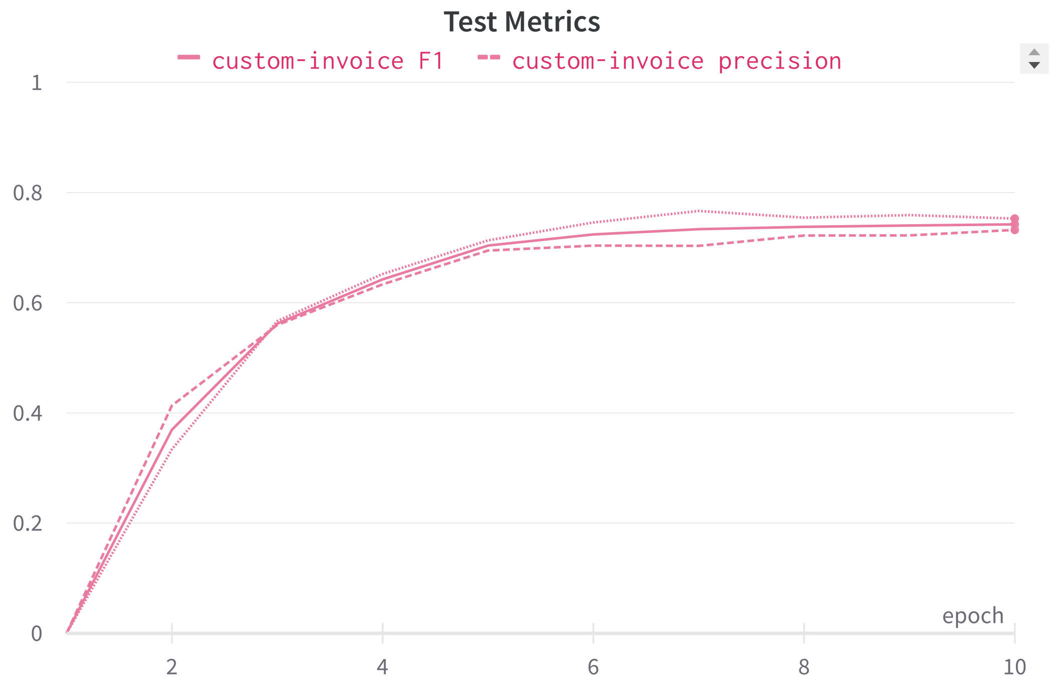
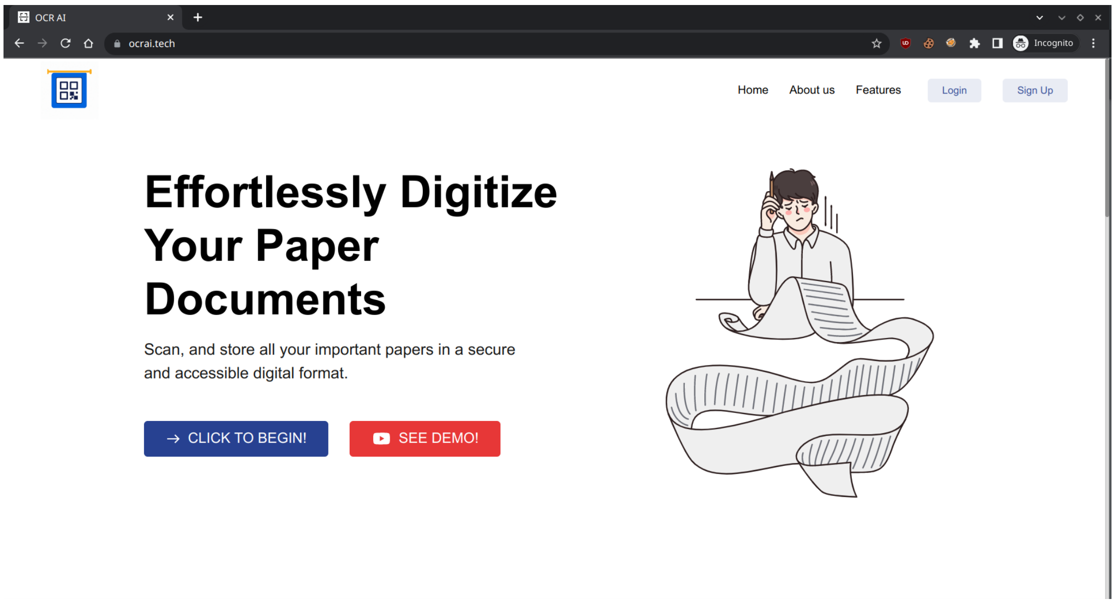


Figure 8.9: Evaluation Metrics for Custom invoice dataset

Appendix B



Effortlessly Digitize Your Paper Documents

Scan, and store all your important papers in a secure and accessible digital format.

Figure 8.10: Web App Homepage

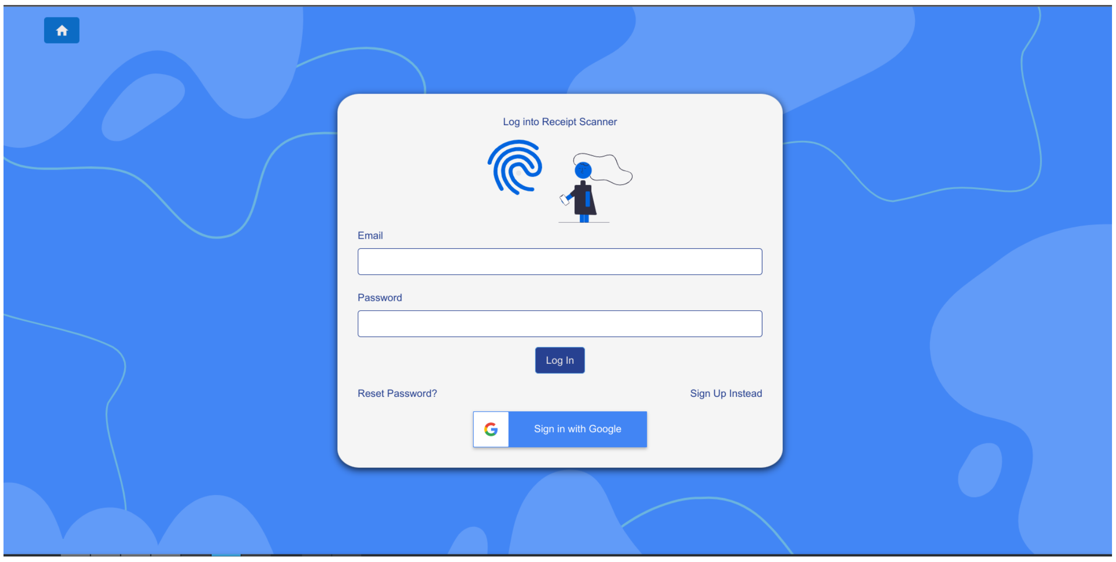
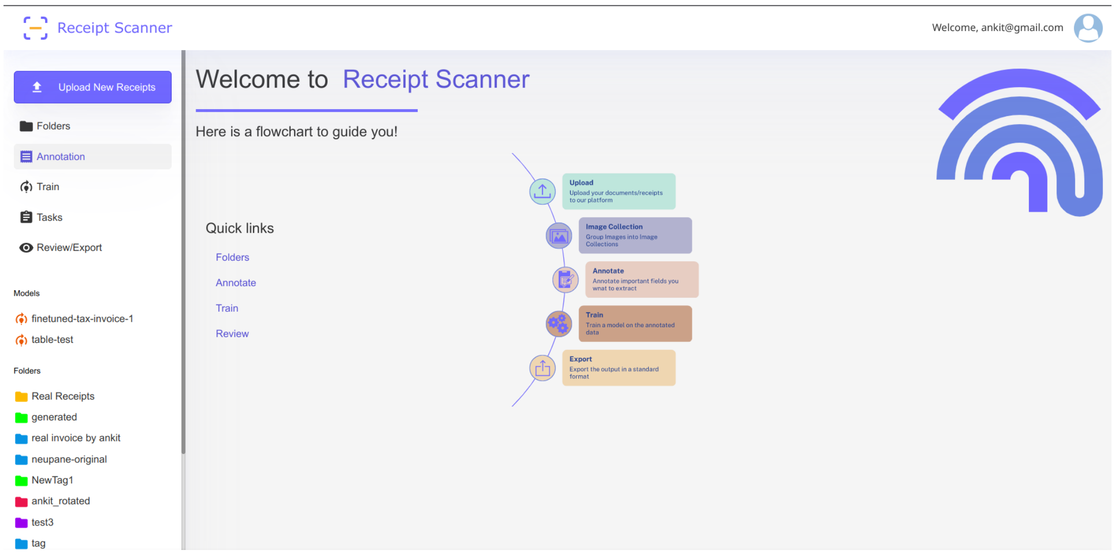


Figure 8.11: Web App Login Page



Welcome to Receipt Scanner

Here is a flowchart to guide you!

Quick links

Folders

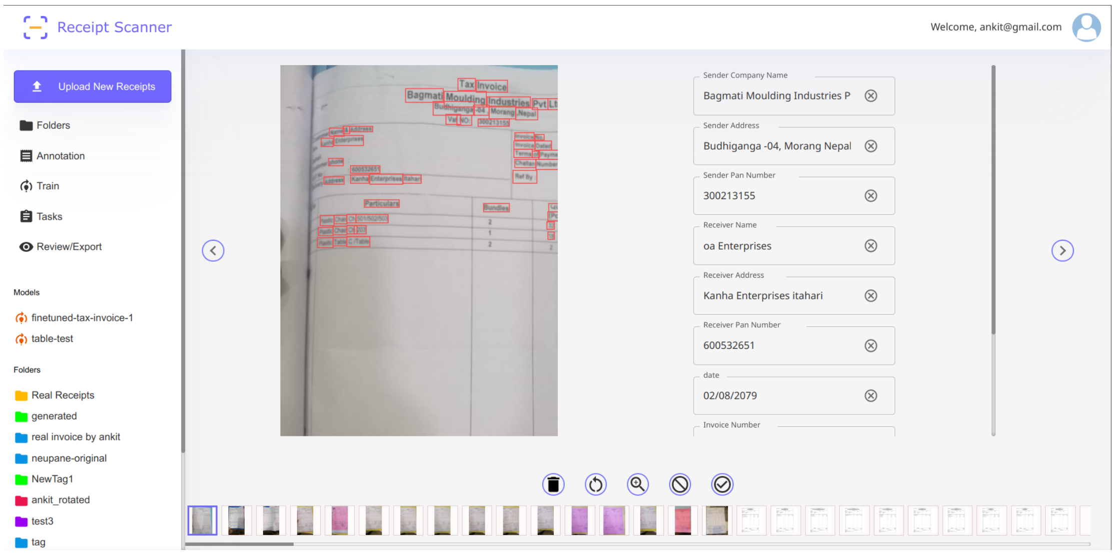
Annotate

Train

Review

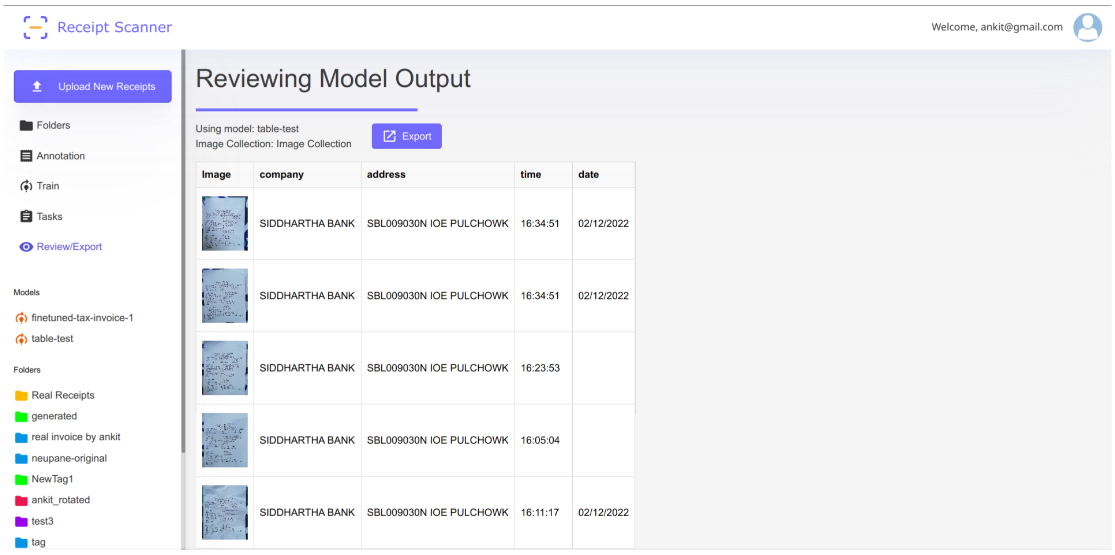
Upload
Upload your documents/receipts to our platform
Image Collection
Group Images into Image Collections
Annotate
Annotate important fields you wnat to extract
Train
Train a model on the annotated data
Export
Export the output in a standard format

Figure 8.12: Web App Dashboard Page



Sender Company Name	Bagmati Moulding Industries P
Sender Address	Budhiganga -04, Morang Nepal
Sender Pan Number	300213155
Receiver Name	oa Enterprises
Receiver Address	Kanha Enterprises itahari
Receiver Pan Number	600532651
date	02/08/2079
Invoice Number	

Figure 8.13: Web App Annotation Page

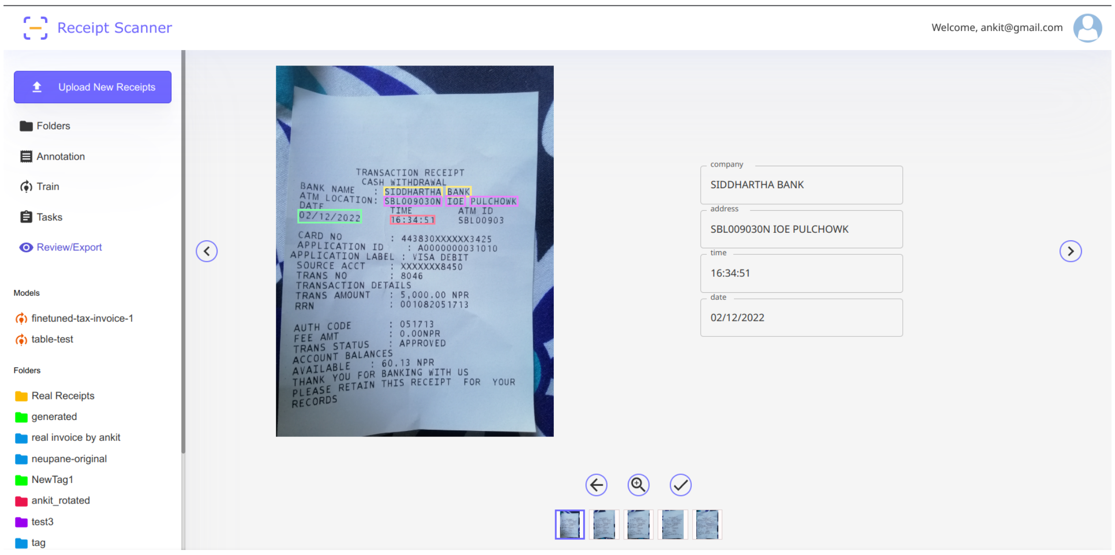


Reviewing Model Output

Using model: table-test
Image Collection: Image Collection

Image	company	address	time	date
	SIDDHARTHA BANK	SBL009030N IOE PULCHOWK	16:34:51	02/12/2022
	SIDDHARTHA BANK	SBL009030N IOE PULCHOWK	16:34:51	02/12/2022
	SIDDHARTHA BANK	SBL009030N IOE PULCHOWK	16:23:53	
	SIDDHARTHA BANK	SBL009030N IOE PULCHOWK	16:05:04	
	SIDDHARTHA BANK	SBL009030N IOE PULCHOWK	16:11:17	02/12/2022

Figure 8.14: Web App Results Page



```
TRANSACTION RECEIPT
 CASH WITHDRAWAL
BANK NAME    : SIDDHARTHA BANK
ATM LOCATION: SBL009030N IOE PULCHOWK
DATE
02/12/2022        TIME          ATM ID
                  16:34:51      SBL00903

CARD NO            : 443830XXXXXX3425
APPLICATION ID        : A0000000031010
APPLICATION LABEL : VISA DEBIT
SOURCE ACCT        : XXXXXXX8450
TRANS NO           : 8046
TRANSACTION DETAILS
TRANS AMOUNT       : 5,000.00 NPR
RRN                : 001082051713

AUTH CODE          : 051713
FEE AMT            : 0.00NPR
TRANS STATUS       : APPROVED
ACCOUNT BALANCES
AVAILABLE     : 60.13 NPR
THANK YOU FOR BANKING WITH US
PLEASE RETAIN THIS RECEIPT  FOR  YOUR
RECORDS
```

company	SIDDHARTHA BANK
address	SBL009030N IOE PULCHOWK
time	16:34:51
date	02/12/2022

Figure 8.15: Web App Single Output Page