Final project: Real-time Receipt Recognition

Author: Jiaen LIU
Part 1: 08/01/2023
Updated Part 2: 20/03/2023

Co-worker: Ivan STEPANIAN

Repository Link: https://github.com/JiaenLiu123/final_project
Fine-tuned donut model link: https://huggingface.co/JiaenLiu/donut_sroie

Introduction

This repository features JiaenLiu's final project for their Bachelor's degree in Computer Science at the Beijing Institute of Petrochemical Technology and International Master Project for EFREI Paris. The project is a web app designed to recognize receipts in real-time and extract essential information from them. Utilizing two CNNs for semantic segmentation of receipts, Tesseract for OCR, and Regex and LayoutLM model families for key information extraction, the project aims to create a robust and accurate system. To further enhance performance and address the limitations of OCR, the Donut model, an OCR-free Visual Document Understanding transformer, is integrated into the system. However, the Donut model presents its own challenges, such as handling different date formats and potential inaccuracies in predicting years due to limited training data. These components and their respective challenges are discussed in detail in the following sections.

Background

With the background of digital transformation, people want to digitalize existing data such as invoices, and recipts. These files are semi-structured. They have a general structure like the receipts, they have the address, telephone number, name of the shop in the top of the receipt, details of items in the middle of the receipt and total amount, details of tax and date in the end of the receipt. Obviously, there is some important structure information inside of the receipt. Instead of just getting the text from receipt, we want a step more, getting all key information inside of one receipt. In this part, I will discuss all the steps before Key Information Extractions of receipts and organize them in sub-tasks format. I will discuss the model I choosed in next section.

- Receipt Localization

  Receipt localization, or more generally, document localization, is defined as the problem of finding a Tetragon contour from an image that can be classified as target document. Generally speaking, this problem is very complex. If you just consider a very simple case, only a whole flat receipt inside of one image with decent contrast with backgroud, traditional machine learning can handle that case. But if you consider more cases such as some parts of the receipt are missing and the receipts are twisted, folded and so on, which all of these are very common in our daily life. Traditional methods are not robust enough to solve these cases. Deep learning based solutions are much more flexible and reliable in these cases. Also, document unwarping can be also applied in this task in order to remove wrinkles and other artifacts.

- Optical Character Recognition

Optical Character Recognition refers to convert any images with textual content to only text inside of this image, such as scanned printings, images of handwritings and photoed documents. There are two main steps in OCR, text detection and text recognition. Text detection's job is to localize all text in the image with bounding box, then text recognition will try to convert all text content within the bounding boxs into machine readable text. For this project,

My approach

- Part 1: **Detect and Segment the receipt in the image**

  In this part, I followed a tutorial offered by LearnOpenCv. It shows how to use convolutional neural networks to detect one receipt inside of one image which the receipt can be twisted, unclosed, and even some parts are missing. Two models are used in this part, mobileNetv3 and ResNet-50, due to the limitation of machines, I do not manage to fine-tune the resnet-50 futher, instead, I fine-tune mobileNetv3 on on our new dataset generated from Findit T1 (Number of Training set: 5391/ Number of Test set: 1079). The best Intersection over Union(IoU) is 0.973 and Loss is 0.072 on test set. The reason why I use this model to detect the receipt is robustness. At the beginning, I try to use traditional machine learning algorithms to find the edge of the receipts and based on that detected edges to locate the receipt. But this approach failed in many cases, such as the receipt is unclosed, twisted, low constract with the background and so on. With the help of CNNs, these problems are solved. But CNNs also have their own limitation and I will discuss it in Need to be improved section.

- Part 2: **Resize the receipt and remove the shadow**

  To be honest, I think this part is very important for the next part. Because image quality is very vital for the Optical Character Recognition (OCR) and Name Entity Recognition. There is no one size fits all solution for that. Due to the time limitation, I use some traditional methods for this part, image resizing and shadows removing. Also, there are a lot of works can be done in this part such as image quality assessment and image quality improvement. I will discuss this in detail in Need to be improvede section.

- Part 3: **OCR, Donut Model, and Key Information Extraction**

  This part is the most important part of my project, and it is also the most time-consuming part. There are three tasks: OCR, Donut Model integration, and Key Information Extraction. The OCR part is relatively easy, and I choose to use Tesseract OCRv5 in my project with the configuration OCR engine modes 2 and Page segmentation modes 12. To mitigate the limitations of OCR, we also integrate the Donut model, an OCR-free VDU transformer that directly maps raw input images to desired outputs without relying on OCR. While this approach can help improve speed and accuracy while reducing dependency on OCR engines, it has its own limitations, such as difficulty handling different date formats and potential inaccuracies in predicting years due to limited training data.

  Ideally, I want to deploy a transformer-based model LayoutLMv3 and Donut to extract all key information inside of one receipt such as date, total amount, address, company name, name of items, price of items, and tax information. But to achieve that, the tasks are too much to be done within two months. Instead, I select date and total amount as my goals in this time.

With my co-worker, we create three approaches for this task. First approach is based on regular expression, depending on format, location, currency sign and so on. A navie weighted mechanism is applied in this approach to check every word in the OCR output whether it is the date or the total amount of the receipt. I use this regex script to label findit Task 1 dataset (500 images). The accuracy of this script is 0.72.

For the second approach, I partly managed to use a transformer-based model layoutLMv3 to extract the date and total amount from the receipt. In this approach, I use a pre-trained model from Theivaprakasham which is fine-tuned on SOIRE dataset. I try to extend his work and fine tune his model on Findit dataset. But I failed to achieve that. I put a lot work to label the findit dataset and convert it into the format that layoutLMv3 required and it is not finished yet. Also, I add a function to get JSON format output from the model. In order to do that, I fully restructure my code and I will explain this in later parts. But I faced a problem of inaccurate OCR output, too much misspelling and unrecognized characters in there. Due to that, I can not test the accuracy of this model based on it's text output.

Despite the limitations of the Donut model, its integration can still provide valuable improvements to the overall performance of the system. To address the issues with handling different date formats and potential inaccuracies in predicting years, a combination of approaches, including Regex, LayoutLM, and the Donut model, can be employed to enhance the robustness and efficiency of receipt recognition and information extraction. Further fine-tuning and expanding the training data for the Donut model could help mitigate these limitations and contribute to a more accurate and versatile system.

My own understanding of LayoutLMv2 model:

At the beginning, I start to use layoutLMv2 as the main algorithms for the key information extraction. This model will leverage the information from 3 aspects, token embedding, visual embedding and layout embedding. For the token embedding, there are three parts in it, word embedding and segment embedding from WordPiece and 1D positional embedding to represent the token index. Visual embedding is generated from a CNN-based visual encoder, however, since CNN can not extract the positional information, a 1D positional embedding and segment embedding are added. For layout embedding, the spatial layout information is respented by axis-aligned token bounding boxes from the OCR results.

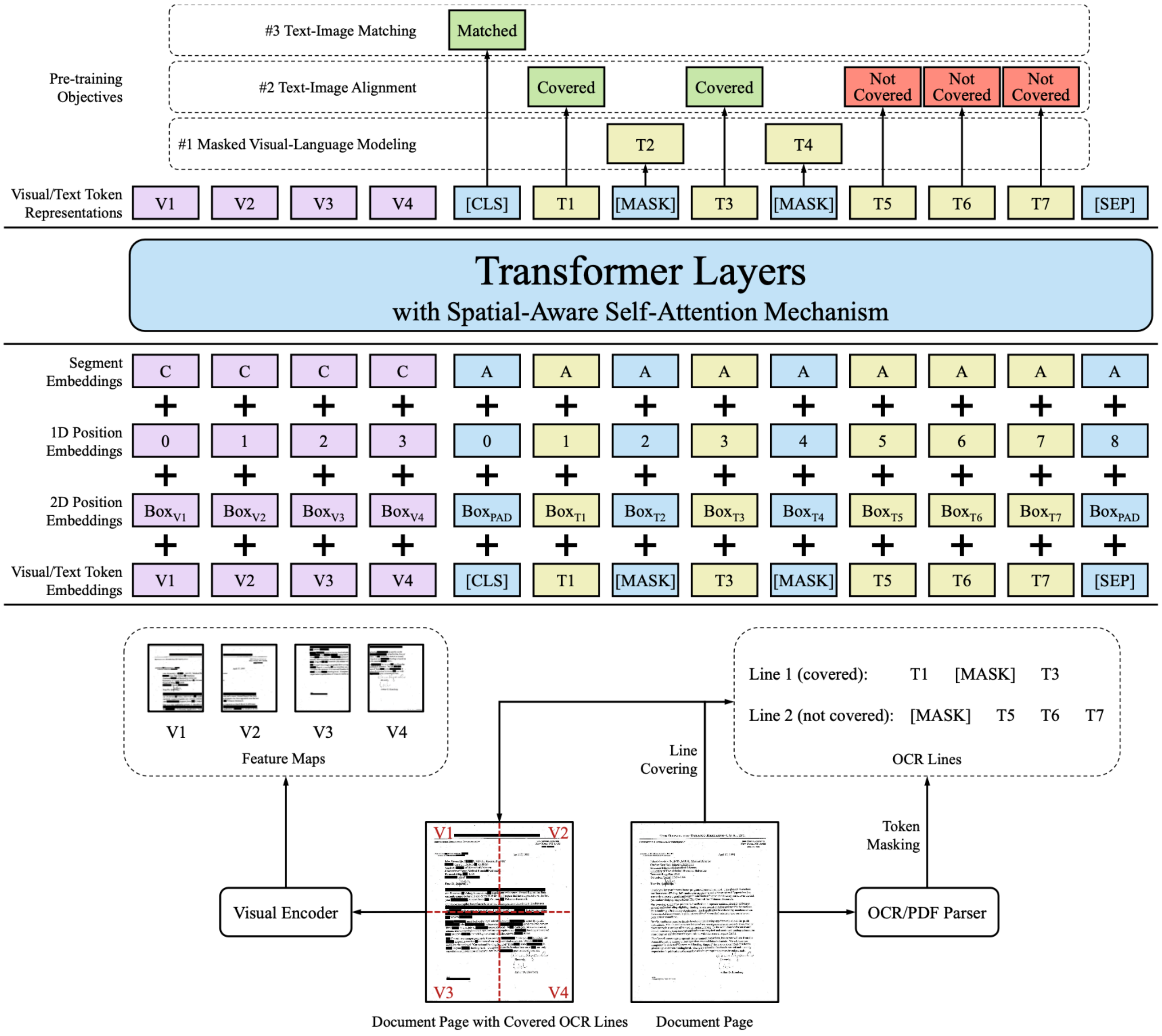

#3 Text-Image Matching
Matched
Pre-training Objectives
#2 Text-Image Alignment
Covered
Covered
Not Covered
Not Covered
Not Covered
#1 Masked Visual-Language Modeling
T2
T4
Visual/Text Token Representations
V1 V2 V3 V4 [CLS] T1 [MASK] T3 [MASK] T5 T6 T7 [SEP]
Transformer Layers
with Spatial-Aware Self-Attention Mechanism
Segment Embeddings
C C C C A A A A A A A A A
1D Position Embeddings
0 1 2 3 0 1 2 3 4 5 6 7 8
2D Position Embeddings
Box_V1 Box_V2 Box_V3 Box_V4 Box_PAD Box_T1 Box_T2 Box_T3 Box_T4 Box_T5 Box_T6 Box_T7 Box_PAD
Visual/Text Token Embeddings
V1 V2 V3 V4 [CLS] T1 [MASK] T3 [MASK] T5 T6 T7 [SEP]
V1 V2 V3 V4
Feature Maps
Line 1 (covered): T1 [MASK] T3
Line 2 (not covered): [MASK] T5 T6 T7
OCR Lines
Line Covering
Token Masking
Visual Encoder
OCR/PDF Parser
Document Page with Covered OCR Lines
Document Page

Figure 1: An illustration of the model architecture and pre-training strategies for LayoutLMv2

Then I find that this model has a new version, layoutLMv3 and then I switch to the new model. There are two main modifications in the new model. First, layoutLMv3 change the ways of layout embedding, instead of using word-level layout positions, new model uses segment-level layout positions. The authors think that words in a segmnet share the same 2D position since the words usually express the shame semantic meaning. Second, for the image embedding, inspired by ViT and ViLT, layoutLMv3 uses linear projection features of image patches to represent the document images and then fitting these embeddings into multimodal Transformer. To be honest, I do not have a deep understanding about that, I am not familiar with Vision Transformers. But I think this will reduce the computation and region supervision is not required anymore. And from the numbers given in the paper, new model beats the old one and the parameters are much less than previous model.

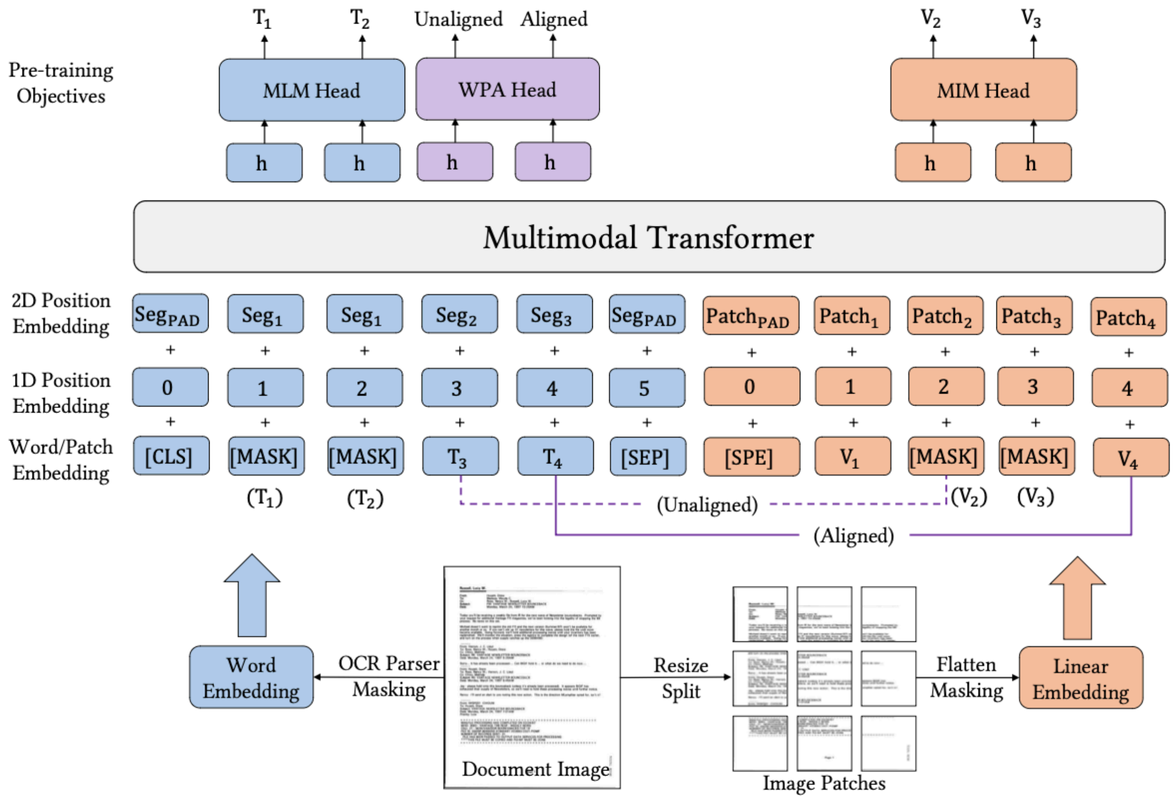


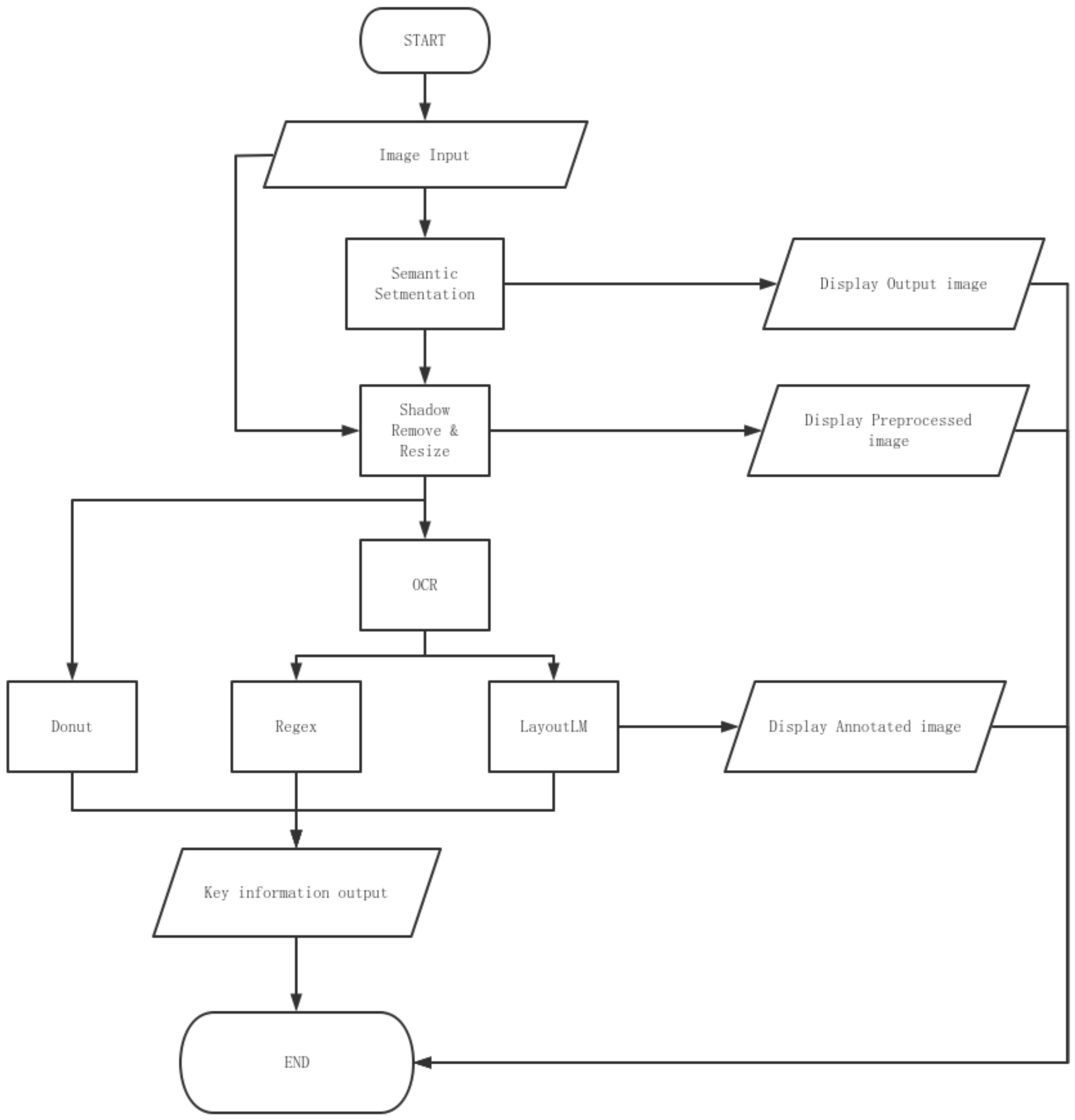
Figure 3: The architecture and pre-training objectives of LayoutLMv3. LayoutLMv3 is a pre-trained multimodal Transformer for Document AI with unified text and image masking objectives. Given an input document image and its corresponding text and layout position information, the model takes the linear projection of patches and word tokens as inputs and encodes them into contextualized vector representations. LayoutLMv3 is pre-trained with discrete token reconstructive objectives of Masked Language Modeling (MLM) and Masked Image Modeling (MIM). Additionally, LayoutLMv3 is pre-trained with a Word-Patch Alignment (WPA) objective to learn cross-modal alignment by predicting whether the corresponding image patch of a text word is masked. "Seg" denotes segment-level positions. "[CLS]", "[MASK]", "[SEP]" and "[SPE]" are special tokens.

The reason why LayoutLM model family will both take the token embedding and layout embedding into consideration is that they have different purpose. The CNN or ViT as a visual encoder to extract visual features from the document image. These visual features capture information related to the appearance of the text and other visual elements in the document. The visual embeddings are created by combining the extracted visual features with 1D positional embeddings and segment embeddings.
So, to clarify, LayoutLMv2 uses:

- Bounding box coordinates from the OCR output to create layout embeddings.
- CNN-based or Transformer-based visual encoder to extract visual features from the document image, which are then used to create visual embeddings.

The combination of token, visual, and layout embeddings allows the model to capture the rich contextual, visual, and spatial information present in the document, which is crucial for tasks involving semi-structured documents.

Flowchat of whole application:



New things in International Master Project Part 2

1. Totally restructed code of the project

In the first part of the project, I put all the code in one file. It is very hard to read and understand. So I restructed the code and put them in different files. I also add some comments to make the code more readable. Then, in order to solve the problem of the model's output, I create a new way to call the model with a more stable way. OCR and KIE are seperated for the cleanning of OCR ouput. After that, I add a new post-processing in order to get the output of the model properly.

For the new post-processing method, this method will process the output from the model inference and organizes it into a readable JSON format. Basicly speaking, the function will interates through the output data from the model and assigns them labels based on the model's predictions. Words with a non-'other' label are added to a list called 'doc_list'. Next, it groups adjacent words with the same label into spans. The output includes the text, label, and constituent words of each span, along with their respective IDs, bounding boxes, and text. Finally, the method organizes the output into a JSON string that represents a list of dictionaries, each containing an 'output' key with a list of the identified spans. In the end, the method returns the JSON string which provides a structured representation of the labeled spans, making it easy to parse and visualize the extracted information from the input document.

2. Add a new Donut model to extract date and total amount from receipt

Donut is an end-to-end, self-contained Visual Document Understanding (VDU) model designed for general understanding of document images. The architecture is composed of a Transformer-based visual encoder and textual decoder modules. Unlike other models, Donut does not rely on OCR functionality but uses a visual encoder to extract features from document images. The textual decoder then maps these features into a sequence of subword tokens to construct a desired structured format, like JSON.

The encoder can use CNN-based models or Transformer-based models, with Swin Transformer being utilized in this model due to its superior performance in document parsing. The decoder employs the BART architecture, initialized with weights from a pre-trained multi-lingual BART model. This can explain why Donut has a multi-language ability. It will take the visual embedding from encoder and prompt as input and generate a token sequence given a prompt, similar to GPT-3.

During training, the model is trained to read all texts in an image in reading order, minimizing cross-entropy loss of next token prediction. In other words, in the training phrase, the model's job is very similar to OCR. Fine-tuning teaches the model to understand document images by interpreting downstream tasks as JSON prediction problems. The decoder generates a token sequence convertible into a JSON that represents the desired output information, allowing the model to handle various tasks such as document classification, named entity recognition, and more.

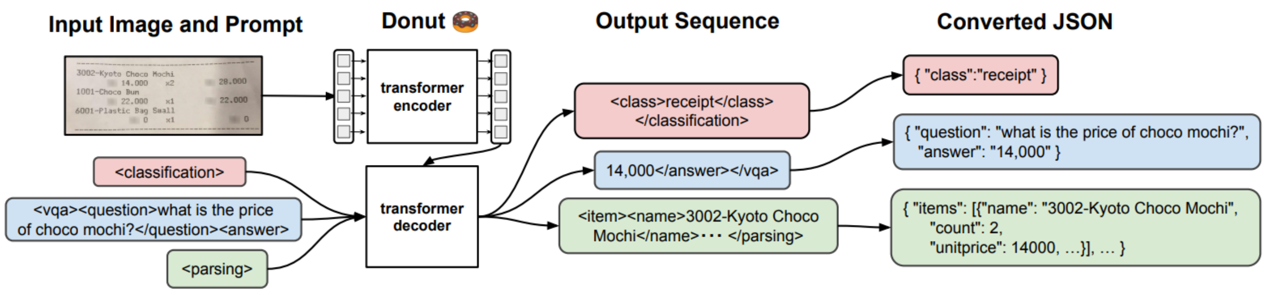

Input Image and Prompt
Donut
Output Sequence
Converted JSON
transformer encoder
transformer decoder
<classification>
<vqa><question>what is the price of choco mochi?</question><answer>
<parsing>
<class>receipt</class></classification>
14,000</answer></vqa>
<item><name>3002-Kyoto Choco Mochi</name>··· </parsing>
{ "class":"receipt" }
{ "question": "what is the price of choco mochi?", "answer": "14,000" }
{ "items": [{"name": "3002-Kyoto Choco Mochi", "count": 2, "unitprice": 14000, …}], … }

Fig. 3. The pipeline of Donut. The encoder maps a given document image into embeddings. With the encoded embeddings, the decoder generates a sequence of tokens that can be converted into a target type of information in a structured form

Due to the time constraints of this project, I only fine-tuned the model on the SROIE dataset for date and total amount. I chose 400 images for the training set and 100 images each for the test set and validation set. The model was trained for 30 epochs with a learning rate of 3e-5, the training script can be found in Donut_fine_tune_script_sroie.ipynb and train_sroie.yaml. For the ground truth, date, address, company, and total amount were provided, which means Donut cannot detect the location of the text. This problem can be solved by providing such information during the training phase. The result for the fine-tuned model is good, in the test set from SROIE dataset fine-tuned model got accuracy at 0.9519 and f1-accuracy is 0.85. All the details of testing donut on SROIE dataset can be found in sroie_output.json. But in real use case. the model's result is not as satisfactory compared to the performance in test set. Upon analyzing the dataset, I found that the date's data variety is not good enough; all the training set is from 2018, which explains why the model cannot predict the year correctly. Also, when the date format differs significantly from the training set, the model fails to predict the date. I believe that if I can find a better dataset with a more diverse range of dates, the model will perform better. Additionally, fine-tuning the model for more tasks could also improve performance. I find this model very promising and will continue to work on it in the future.

Compared to the results of the LayoutLM family, the text output of Donut is much better. However, as mentioned earlier, the model cannot detect the location of the text. So, I think combining the results of Donut and LayoutLM could yield better outcomes. I will try to do that in the future. Also, I plan to add more data for future fine-tuning to make the model more robust and capable of extracting more information from receipts.

3. Reorganized web interface

In part 2, I optimise the web layout and add more options for users to choose. Which allows users to compare the performance between different models and processing methods.

Future Work

1. The improvement of image preprocessing (Unwarp the image)

As I mentioned before, CNNs can detect and segment the receipt when the receipt is twisted and unclosed. But when they cannot unwarp the image back to flat. This can be done by DocUNet and DocTr. These models are desigened to unwarp the image back to its original state. I want to apply these methods in my project. But the cost of computation is too much to accept it. In future, I want to intergrate image unwarping into this project with a efficient way.

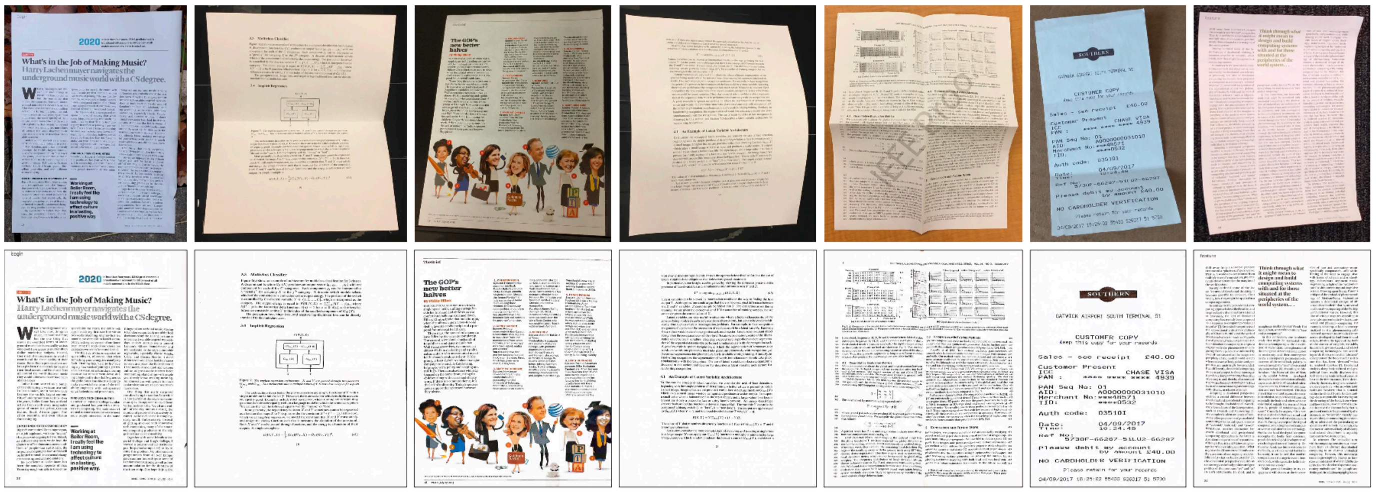


Figure 1: Qualitative rectified results of our Document Image Transformer (DocTr). The top row shows the distorted document images. The second row shows the rectified results after geometric unwarping and illumination correction.

2. Image Quality Assessment & Image Quality Improvement

Image Quality is very important for the OCR and KIE. So I think if I can improve the image quality of input images, the accuracy of the regex and layoutLM will be improved a lot due to a better OCR output. Also, this can be one important factor of our model. If the input quality is bad and model does not extract the information from receipt, that is OK. But if the input quality is good and model also does not extract the information, that is unacceptable. But the image quality is a very subject matrix, that's why I need a stable IQA algorithm to do it instead of doing it manually. I hope I can add this in future to make the project be more accurate and convincing.

3. Full information extraction

As I discribed before, my goal is to create a tool that can extract all useful information from one receipt. I think if I manage to fine tune the model on my new dataset of date and total amount. I can apply this procedure to other information. I need more time to do that and I need to fully understand about this model, how it works and why it works. There are still a lot of jobs to do in this part.

In the future, our work will focus on enhancing the performance and capabilities of our models. We aim to combine the strengths of the Donut model and LayoutLM family to achieve more accurate text recognition and location detection. To accomplish this, we will seek better and more diverse datasets for fine-tuning, particularly with a wider range of dates and formats. Additionally, we plan to expand the tasks for which the model is fine-tuned, enabling it to extract a broader set of information from receipts and other documents. Finally, we will continue refining the web interface to offer users a more seamless experience, allowing them to easily compare and evaluate the performance of various models and processing methods. By addressing these areas, we hope to develop a more robust and versatile solution for document understanding and information extraction.

Installation & Usage Example

Ideally, you need a machine in Ubuntu 18.04 to run this project. There is no GPU requirement. You can also run

Ideally, you need a machine in Ubuntu 18.04 to run this project. There is no GPU requirement. You can also run it in Windows by Windows Subsystem For Linux, that is due to detectron2 is not officially supported on Windows. You need to install tesseract OCR engine manually. The following packages are required to run this project:

```
# First download all codes from this repository
git clone https://github.com/JiaenLiu123/final_project
cd final_project

# Make sure you have install gcc ≥ 5.4 and g++ ≥ 5.4, detectron2 requires them to compile
the C++ code.
# If you don't have them, you can install them by:
sudo apt update
sudo apt install gcc g++

# optional, create a virtual environment by conda
conda create -n final_project python=3.8
conda activate final_project
# if you need gpu acceleration, you can install pytorch and torchvision by:
conda install pytorch torchvision torchaudio pytorch-cuda=11.6 -c pytorch -c nvidia
# if you don't need gpu acceleration, you can just uncomment torch and torchvision in
requirements.txt and install them.

# Install the required packages
python -m pip install -r requirements.txt
python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'
python -m pip install torchvision tesseract

# For Ubuntu to install Tesseract 5
sudo apt update
sudo add-apt-repository ppa:alex-p/tesseract-ocr-devel
sudo apt install -y tesseract-ocr
sudo apt update
# Check the version is correct (Should be the latest version)
tesseract --version

# Add the language data for Tesseract 5
# Use the following command to check the available languages and path to the language data
tesseract --list-langs
# Download the language data
wget https://github.com/tesseract-ocr/tessdata/blob/main/fra.traineddata
# if wget cannot work, you can download it from the link and put it in the correct path
# Be careful to make sure the fra.traineddata and eng.traineddata are correct



# After successfully install all dependencies, you can just run following command to use
# After successfully install all dependencies, you can just run following command to use
the streamlit web application. For the first run, it takes some time to download models
from Internet.

# Updated in Part 2 of the project

# In order to use fine-tuned Donut model, you need to download it from the huggingface
website.
# Make sure you have installed git lfs check the following link to install it
https://docs.github.com/en/repositories/working-with-files/managing-large-
files/installing-git-large-file-storage?platform=linux

# download the model
git clone https://huggingface.co/JiaenLiu/donut_sroie

# run the streamlit web application
streamlit run cleaned_app.py

# Run to test the layoutLMv3 model
python test_handler.py --image_path {YOUR_IMAGE_PATH}
```

Thanks

This project is based on the following projects:
https://learnopencv.com/deep-learning-based-document-segmentation-using-semantic-segmentation-deeplabv3-on-custom-dataset/
https://huggingface.co/spaces/Theivaprakasham/layoutlmv2_sroie
https://github.com/clovaai/donut
https://towardsdatascience.com/ocr-free-document-understanding-with-donut-1acfbdf099be

Datasets:

Findit:

https://drive.google.com/file/d/1fPOTGQS_wX0xCABrmk87OnWCr0rEE-Cx/view?usp=share_link
https://drive.google.com/file/d/1Zog-6cY35vuJcJ24I6ZCKR57zEXqyrNB/view?usp=share_link
https://drive.google.com/file/d/1GzCYqEqOTk2YRLUQemyOBV0TURB2RyPD/view?usp=share_link

SORIE:

https://drive.google.com/drive/folders/1xvZ6jpZQfpMHJCUD0L_5KHQy0T9QAepx?usp=share_link
https://drive.google.com/drive/folders/11lDpn-rO3AmmyIMCYxYizZnP6BPgETsy?usp=share_link

Important papers:

Ylisiurunen, Markus. "Extracting Semi-Structured Information from Receipts." (2022).

Feng, Hao, et al. "Doctr: Document image transformer for geometric unwarping and illumination correction."

Feng, Hao, et al. "DocTr: Document image transformer for geometric unwarping and illumination correction." *arXiv preprint arXiv:2110.12942* (2021).

Huang, Yupan, et al. "LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking." *arXiv preprint arXiv:2204.08387* (2022).

Xu, Yang, et al. "Layoutlmv2: Multi-modal pre-training for visually-rich document understanding." arXiv preprint arXiv:2012.14740 (2020).

Artaud, Chloé, et al. "Receipt Dataset for Fraud Detection." *First International Workshop on Computational Document Forensics*. 2017.

Howard, Andrew, et al. "Searching for mobilenetv3." *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.

Ma, Ke, et al. "Docunet: Document image unwarping via a stacked u-net." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Kim, Geewook, et al. "Ocr-free document understanding transformer." Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII. Cham: Springer Nature Switzerland, 2022.