NOVA IMS
Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

# FINE-TUNING A TRANSFORMERS-BASED MODEL TO EXTRACT RELEVANT FIELDS FROM INVOICES

Rui Francisco Pereira Moital Loureiro da Cruz

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

**NOVA Information Management School**
**Instituto Superior de Estatística e Gestão de Informação**
Universidade Nova de Lisboa

# FINE-TUNING A TRANSFORMERS-BASED MODEL TO EXTRACT RELEVANT FIELDS FROM INVOICES

by

Rui Francisco Pereira Moital Loureiro da Cruz

Dissertation presented as partial requirement for obtaining the Master's degree in Advanced Analytics

**Advisor**: Prof. Mauro Castelli

November 2021

# ACKNOWLEDGEMENTS

I must express my deepest acknowledgement to those who so largely contributed for the completion of this work and all the fun I had doing it. To my supervisor, Prof. Mauro Castelli, I must thank all the precious contributions and openness for discussion. Furthermore, for his motivation and financial support on publishing my work. To António and Miguel, for letting me be part of Feels Like Home since the beginning. This thesis would not be possible without their financial support and the freedom they gave me to conjugate work and study.

To my mother and father, for being the best references I could have and their unconditional support. A special thanks to my mother for photographing more than a thousand invoices with so much love and care. To José Afonso and Manuel, life feels great with you next to me and coding feels special with you on my lap. To Inês, who gave so much of her life to turn mine easier.

# ABSTRACT

Extraction of relevant fields from documents has been a relevant matter for decades. Although there are well-established algorithms to perform this task since the late XX century, this field of study has again gathered more attention with the fast growth of deep learning models and transfer learning. One of these models is LayoutLM, which is a Transformer-based architecture pre-trained with additional features that represent the 2D position of the words.

In this dissertation, LayoutLM is fine-tuned on a set of invoices to extract some of its relevant fields, such as company name, address, document date, among others. Given the objective of deploying the model in a company's internal accountant software, an end-to-end machine learning pipeline is presented. The training layer receives batches with images of documents and their corresponding annotations and fine-tunes the model for a sequence labeling task. The production layer inputs images and predicts the relevant fields.

The images are pre-processed extracting the whole document text and bounding boxes using OCR. To automatically label the samples using Transformers-based input format, the text is labeled using an algorithm that searches parts of the text equal or highly similar to the annotations.

Also, a new dataset to support this work is created and made publicly available. The dataset consists of 813 pictures and the annotation text for every relevant field, which include company name, company address, document date, document number, buyer tax number, seller tax number, total amount and tax amount.

The models are fine-tuned and compared with two baseline models, showing a performance very close to the presented by the model authors. A sensitivity analysis is made to understand the impact of two datasets with different characteristics. In addition, the learning curves for different datasets define empirically that 100 to 200 samples are enough to fine-tune the model and achieve top performance. Based on the results, a strategy for model deployment is defined. Empirical results show that the already fine-tuned model is enough to guarantee top performance in production without the need of using online learning algorithms.

# KEYWORDS

Document data extraction; Deep Learning; Transformers; Invoice dataset

# INDEX

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS AND ACRONYMS

**BERT** Bidirectional Encoder Representations from Transformers

**CER** Character Error Rate

**CNN** Convolutional Neural Network

**CV** Computer Vision

**DL** Deep Learning

**FLH** Feels Like Home

**MDC** Multi-label Document Classification

**MVLM** Masked Visual-Language Model

**NLP** Natural Language Processing

**OCR** Optical Character Recognition

**R-CNN** Recurrent Convolutional Neural Network

**RNN** Recurrent Neural Network

**SROIE** Scanned Receipts OCR and Information Extraction

**WER** Word Error Rate

# 1. INTRODUCTION

## 1.1. EXTRACT RELEVANT FIELDS FROM DOCUMENTS USING NATURAL LANGUAGE PROCESSING

Processing and storing receipt and invoice data are common tasks in every business or organization. Both financial balance and tax credits request strongly rely on processing such documents. Also, saving digitalized document data can serve many applications and services, such as efficient archiving, fast indexing, and document analytics.

Generally, invoices or receipts do not have standard layout rules. As it is not straightforward to automatize the process, many organizations base their document data extraction tasks on human effort. Nevertheless, developments were made since the 1980s to implement algorithms for such tasks, not only in commercial software but also in academia.

The literature published in the past decades about this subject reveals two periods with different perspectives on solving the problem. First, following the development of Optical Character Recognition (OCR) techniques, in the late decades of the XX century, most of the algorithms extract data based on layouts previously defined by users. Such a need to pre-define layouts highlights the lack of flexibility of those algorithms.

The second period occurs from the last ten years until today, with the resurgence of deep learning techniques, which dramatically improve the learning of tasks in the domains of CV and NLP. Also, models with complex architectures pre-trained for millions of samples are made available and the concept of transfer learning is generalized. More focus should be given to the last few years, in which the advances in NLP model architectures, such as Transformers, allow interpretation of language models with a strong context relationship between words.

Considering the need for a tool with the flexibility to extract data from documents with different layouts, Deep Learning appears as the best option for the task. Interpreting and extracting data from a document can be either a problem of CV or NLP. On the one hand, it requires analysis of images in which relevant fields may be differentiated by visual features (ex: total amount has usually a bigger font size). Likewise, the position of relevant fields has a spatial context within the document (ex: the total amount of an invoice is usually printed on the right side in the bottom half of the document). It may also be stated that the text present in invoices and receipts has a completely different, or almost none, grammatical structure when compared to more generalist texts.

On the other hand, despite having a different grammatical structure, there are still patterns in the structure of the text (ex: the word “Total” is usually followed by a number which represents the total amount). This leads to the idea, together with recent bibliography on the subject, that NLP models are suitable for the task. Chosen out of NLP models, LayoutLM is a Transformers-based model which distinguishes from other models on document understanding tasks because it uses not only textual context but also the spatial position of the text within the document.

## 1.2. RESEARCH MOTIVATION

*Feels Like Home, Mediação Imobiliária* (FLH) is a Portuguese company in the tourism industry that manages apartments for short renting. The accountability in the company requires that every expense

associated with an invoice is registered and communicated to tax authorities. Also, such expenses are an essential part of the company's financial balance. Nowadays, the task of registering invoice data is performed manually. The operational teams responsible for purchases keep the paper tickets and deliver them to the accountant team in batches. The documents are then processed one by one by registering in the company's accountant software the image of the document and some relevant fields, such as seller tax number, buyer tax number, invoice date, invoice number, total and tax amount.

As the volume of documents can easily reach a few thousand in certain months, this intensive task requires human effort, which naturally increases company costs and demotivates workers. FLH is constantly open to investing in process optimization, so this work focuses on filling in the automatization gap in this subject by creating an end-to-end pipeline that receives images and returns strings with relevant fields.

Another motivation exists, which is the interest in developing knowledge on the whole process of deploying a machine learning model, from the creation of a dataset and pre-processing, training of a model, analysis of results, and definition of a strategy for production. On the modeling level, there is also a very specific interest in working with state-of-the-art NLP models and transfer learning. Once gaining a more consolidated knowledge on these subjects, there is a considerable number of processes that can be optimized in FLH, such as automatic extraction of data from identification documents or customer email reply suggestions.

To accomplish such optimization, a group of questions and problems arise:

- The usage of transfer learning requires data and annotations for fine-tuning.
- The input of the pipeline is an image, while NLP models require specific input formats.
- LayoutLM is a recent model with few application examples published. It is not guaranteed that it will have good performance with FLH data.
- Documents used in production may have different characteristics from training ones.

## 1.3. RESEARCH OBJECTIVES

Based on the existing literature, the current situation at FLH, and the presented motivations, this thesis has four main objectives.

Firstly, to create an FLH invoices and receipts dataset with images and annotations. It should have enough samples to build a model with good performance.

Secondly, to create a proof of concept of a production pipeline. This must include:

- Pre-processing stage, with image treatment and OCR, matching between OCR text and annotation, and output in model format.
- Modeling stage, including fine-tuning and prediction methods.
- Post-processing stage with model scoring, including different types of metrics.

The third goal is to work with a state-of-the-art NLP model, perform a sensitivity analysis to test its suitability, not only for the created dataset but also for another already existing dataset.

Finally, the fourth goal consists in defining a strategy for model deployment.

## 1.4. THESIS STRUCTURE

This thesis is organized into seven sections. The first section introduces the research, its motivations, and objectives. The second section presents the most relevant research works that give support to this work, namely legacy document field extraction algorithms and deep learning techniques used for such a task.

The third section presents the created dataset, its characteristics, and the methodology of annotation. This section is structured to be submitted as a data paper in an academic journal.

Section four gives insights into the fundamentals of the model used in this work. Both Transformers and BERT models' architectures are analyzed. Following, a more detailed description is made on LayoutLM specificities.

The fifth section describes the methodology used to develop and implement the pipeline, as well as details on the modeling and evaluation phases.

Section six shows the results of the fine-tuning tasks for various situations. It compares the tuned model with two baseline algorithms: BERT and a commercial software. Following, it presents the results of fine-tuning for each of the used datasets, as well as for mixed datasets. Finally, it analyses the impact of dataset size on model performance and discusses the best strategy for production based on the obtained results.

In the last section, the main conclusions are summarized. Proposals to further improve the implemented pipeline are also presented and discussed.

# 2. LITERATURE REVIEW

## 2.1. INTRODUCTION

Although this thesis is focused on recent techniques, it was considered essential to do an overview of legacy systems. Consequently, the first subsection of this literature review presents a brief description of those systems and algorithms used on document information extraction. For a deeper study of such algorithms, which were developed mostly until the early 2000's, a very complete literature review may be found on Nagy (2000) and Mao et al. (2003).

Considering that this research is centered on using deep learning techniques, the second sub-section is dedicated to such methods. While the model used in this thesis does not focus on text extraction or document segmentation, the review of such techniques was part of the roadmap to define the model that would fit better in this kind of analysis. Therefore, the second sub-section is separated into three parts. The first contains a brief review of scene text recognition. A richer review about scene text recognition using deep learning techniques may be read in Liu et al. (X. Liu et al., 2019). The second part describes works focused on segmenting regions of documents. Finally, the third part focuses on techniques to label the text of documents. Out of every model and technique reviewed, LayoutLM is part of the third group and was chosen as the most suitable for the task.

## 2.2. EARLY DAYS OF DOCUMENT INFORMATION EXTRACTION TECHNOLOGY

Document information extraction from scanned files is a subset of the Document Image Analysis field. Research and commercial projects on this subject date back from the early 1980s (Wahl et al., 1982), following the development of Optical Character Recognition algorithms in the 1960s and 1970s.

Nagy (2000) reviewed 99 articles published over 20 years in the journal *IEEE Transactions on Pattern Analysis and Machine Intelligence*. The author states that some of the main advances in the 1980s and 1990s resulted in fiable and robust solutions in areas related to scanning quality, character recognition (OCR), and image pre-processing – including binarization, skew estimation, and text location. Also, many document-based models were developed in the same period.

Mao et al. (2003) highlight that applications are mostly based on algorithms that interpret both the physical layout and the logical structure of the document. The authors state that layouts and structures vary greatly in complexity (thus reducing the possibility of using formal models) being usually represented by trees that are derived from a set of rules, such as in the example of Figure 2.1.

One example of such algorithms is INFORMys (Cesarini et al., 1998), which is based on graphs. Nodes describe objects or parts of objects, such as lines, instructions, information fields, or logos. Arcs describe mutual relationships between nodes by means of numerical attributes, like the mutual position of the items in nodes, distances, or vectors which connect the center of objects. Figure 2.2 illustrates how the software works. It is noticeable in Figure 2.2a that the user must define the forms models. The algorithm checks if pre-existing graphs, such as the one presented in Figure 2.2b, fit into the invoice.

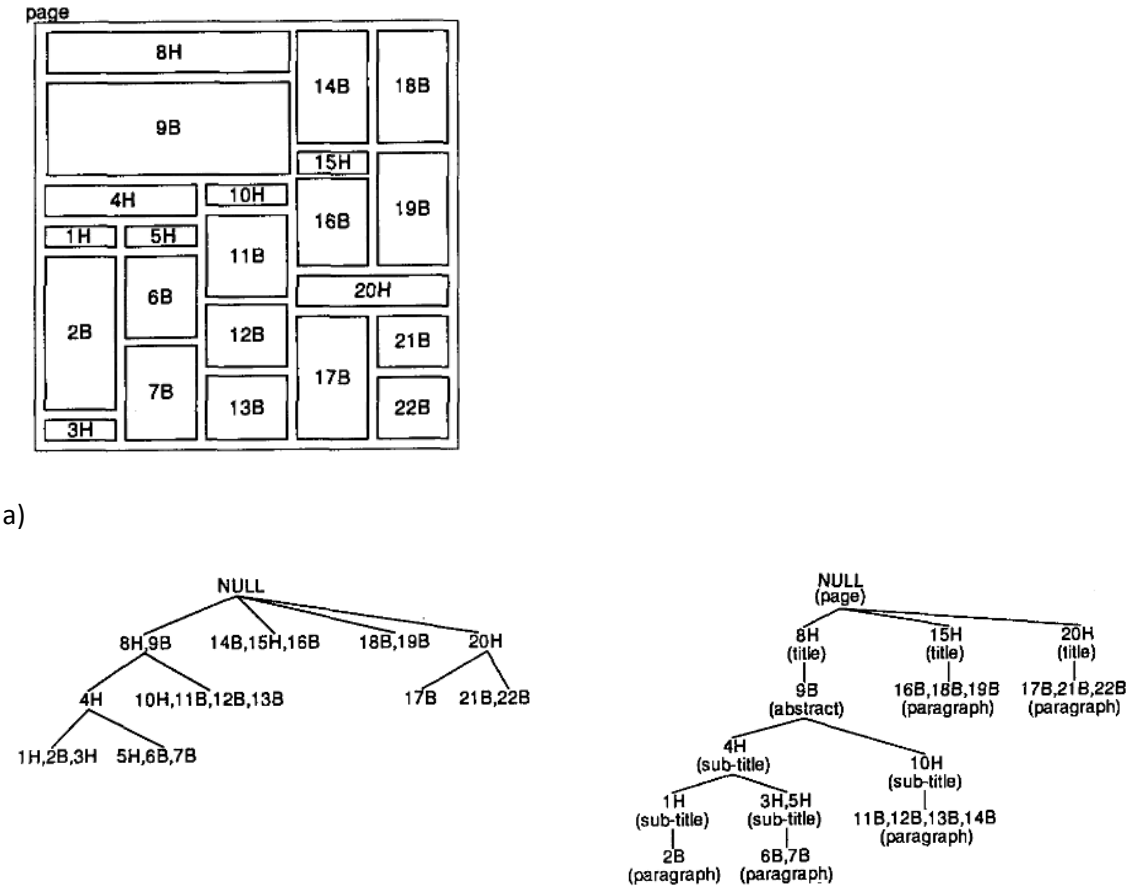


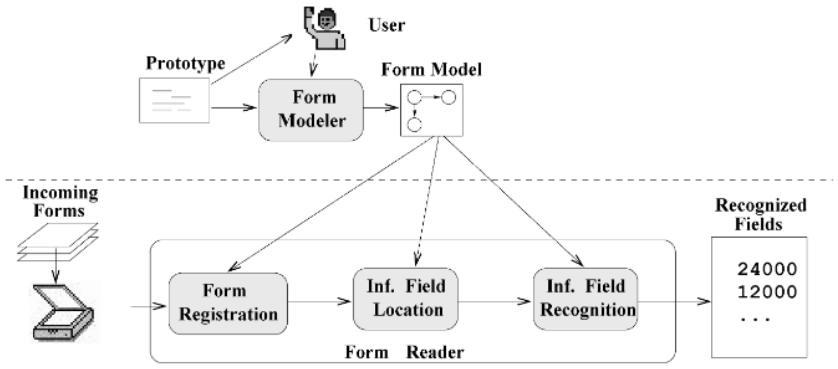

Figure 2.1 – Tree representation of a document (Tsujimoto & Asada, 1990). a) Document divided into blocks. b) Geometric structure. c) Logical structure

This kind of algorithms shows a limitation by resorting to document deterministic models, leading to failure in the presence of noise in the image, which is frequent in scanned documents. Also, the models are too specific for the use cases they are created for, which means that new document models must be input by users.

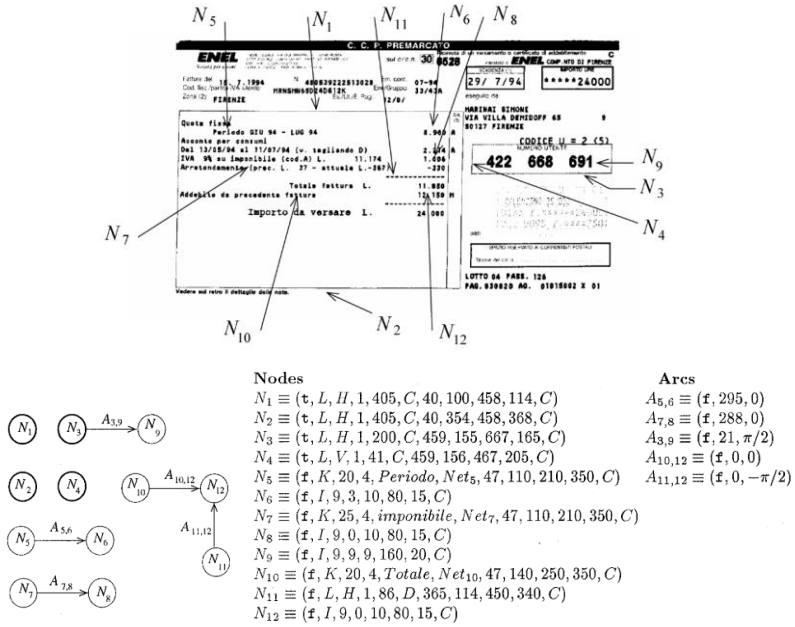
## 2.3. DEEP LEARNING

Deep learning (DL) allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains (Lecun et al., 2015). As scene or document text extraction may be considered a task that mixes both Natural Language Processing (NLP) and Computer Vision (CV), it has obviously benefited from the advances that occurred in DL in the past few years.

Much of the recent development on DL is related to the transfer learning concept. Given the improvements in computer process capabilities and the availability of very large volumes of data, some research groups invested effort on intensively training models that could be used for downstream tasks.




a)




b)

Figure 2.2 – INFORMys (Cesarini et al., 1998). a) Basic structure of the software. b) Example of a graph representation of a document

Transfer learning was also made possible using self-supervised learning (Collobert et al., 2011; Vincent et al., 2008), which drastically reduces the need for annotation. Within self-supervised strategy, the training task is based on unannotated data, and the training is done by removing random known parts of that data and feeding it as a solution to the problem. Self-supervised tasks are not only more scalable, only depending on unlabeled data, but they are designed to force the model to predict parts of the inputs, making them richer and potentially more useful than models trained on a more limited label space (Bommasani et al., 2021).

Transformers architecture founded largely used models, such as BERT (Devlin et al., 2019), GPT-2 (Radford et al., n.d.), RoBERTa (Y. Liu et al., 2019), T5 (Raffel et al., 2019) and BART (Lewis et al., 2019). Firstly used for NLP tasks, Transformers architecture has been showing impressive generalization capabilities, having recent application in images, speech, tabular data, protein sequences, organic

molecules and reinforcement learning. These examples point to a possible future where a unified set of tools may be used in a wide range of modalities (Bommasani et al., 2021). A detailed explanation on Transformers architecture is given in Section 4.

### 2.3.1. Scene or document text extraction

Despite all achievements in OCR technology during the 20th century, new challenges arose in the past few decades. One of the primary ways through which people share and consume information on social networks is through visual media such as photos and videos. In the last several years, the volume of photos being uploaded to social media platforms has grown exponentially to the order of hundreds of millions everyday presenting technological challenges for the retrieval of textual information from such media (Borisyuk et al., 2018).

Rosetta (Borisyuk et al., 2018) is a large-scale system used by Facebook for text detection and recognition in images. Such a system stores in a database the text from images, allowing users to search that text. The algorithm splits the text extraction into two tasks: detection and recognition, as presented in Figure 2.3. The former uses a Faster-RCNN to identify regions of the image that contain text and the latter a fully convolutional character-based recognition model which recognizes the text contained in the regions.

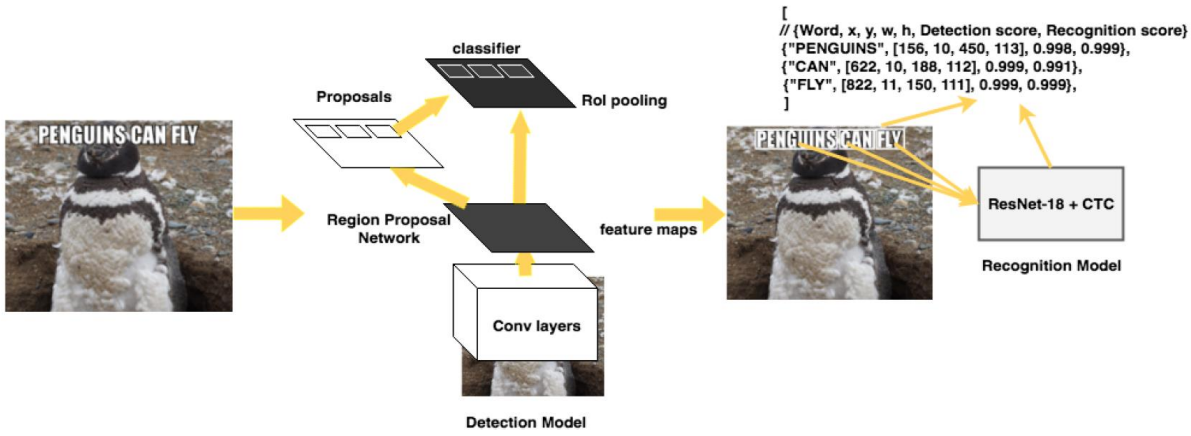



Figure 2.3 – Two-step model architecture of Rosetta (Borisyuk et al., 2018)

Zhang et al. (2016) have presented an approach to detect text, which was very successful in important computer vision competitions, such as ICDAR. It is able of detecting text in multiple orientations, languages, and fonts. It uses firstly a fully Convolutional Neural Network (CNN) inherited from a VGG 16-layer model to predict a "salient map", which is a detection of text blocks at a coarse level. Secondly, text line hypotheses are estimated by combining the salient map and character components. Finally, another fully CNN predicts the centroid of each character. Figure 2.4 presents an example of the whole procedure of text localization.

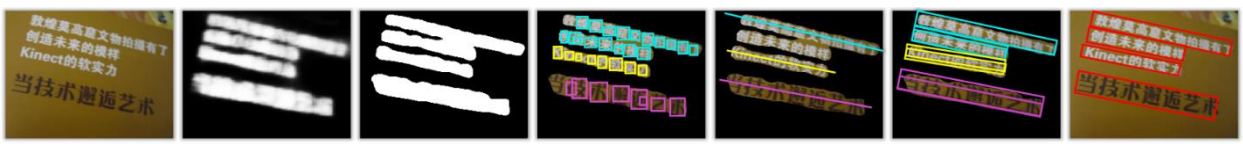


Figure 2.4 – Example of a procedure of the proposed method (Zhang et al., 2016)

W. Liu et al. (2020) have developed an algorithm to extract text from invoices (see Figure 2.5), although not labeling the relevant fields of the text. The authors used an SDD network to determine regions followed by a cropping step of smaller sub-images. Finally, a Convolutional Neural Network – Gated Recurrent Unit combined to extract the text.


Figure 2.5 – Example of the output of the system (W. Liu et al., 2020)

### 2.3.2. Document segmentation

Stahl et al. (2018) have presented a system that outputs different regions of text, as shown in Figure 2.6. The algorithm uses a model with a U-NET architecture to perform semantic segmentation, labeling every pixel of the image. DeeDeSRT (Schreiber et al., 2017) is a system that not only locates tables within images but also detects the structure, rows and columns, of the tables (see Figure 2.7). The model for table detection consists of two parts. The first generates region proposals by a Region Proposal Network, while the second uses a Faster R-CNN for region classification. The model for table structure recognition was based on an FCN semantic segmentation model. The authors obtained state of the art results in ICDAR 2013 competition.

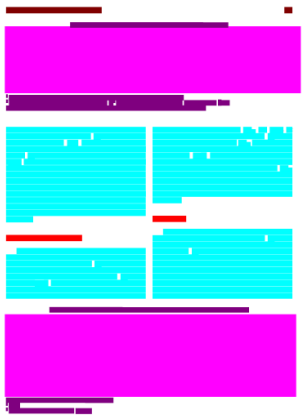
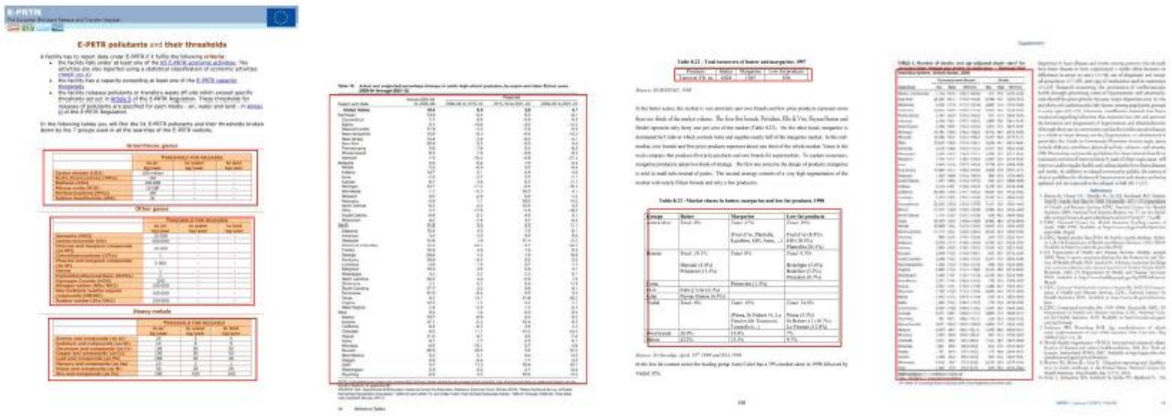


Figure 2.6 – Example of the output with colored regions (Stahl et al., 2018)



a)

b)

Figure 2.7 – Example of DeepDeSRT table detection results (Schreiber et al., 2017). a) Table detection. b) Table structure – rows and columns - detection

Kavasidis et al. (2019) also use semantic segmentation, aligning it with the concept of saliency – the perceptual quality of certain objects that possess distinctive features with respect to the surroundings – to detect tables and charts in image documents. A fully CNN is used to define and label the regions, while a fully connected Conditional Random Field is applied for smoothing the segmentation maps, reducing some of their noisy properties. Figure 2.8 shows the structure of the used algorithm.

Lee et al. (2019) segmented documents using CNN's enhanced by trainable multiplication layers (TML). Apart from common features in CNN, such as pixel-wise information, the algorithm considers features related with the co-occurrence of objects with similar textures and periodicities. Enendu et al. (2019) condense both OCR text and layout information into features, as seen in Figure 2.9. The former is performed using Tesseract, while the latter is obtained by segmentation using Linear-Chain CRF. Then, the different regions of the document are modeled as a sequence in a Recurrent Neural Network.

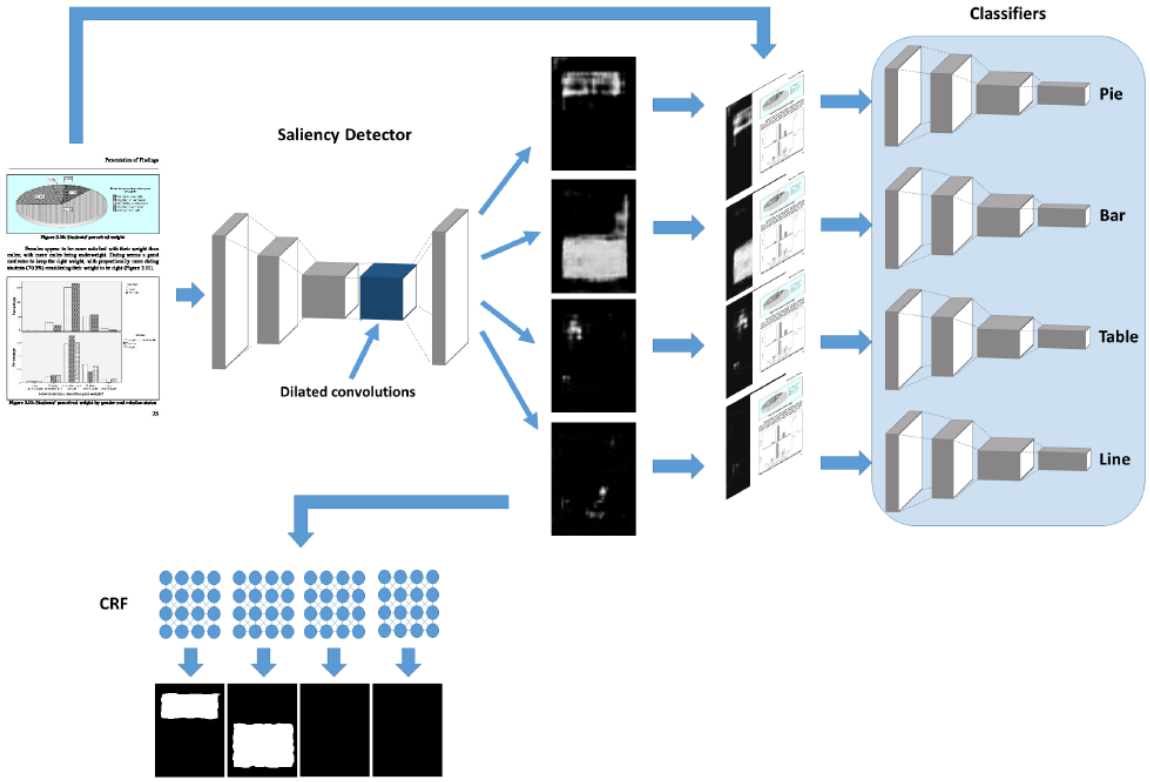



Figure 2.8 –A CNN extracts class-specific saliency maps which are enhanced by CRF models (Kavasidis et al., 2019)

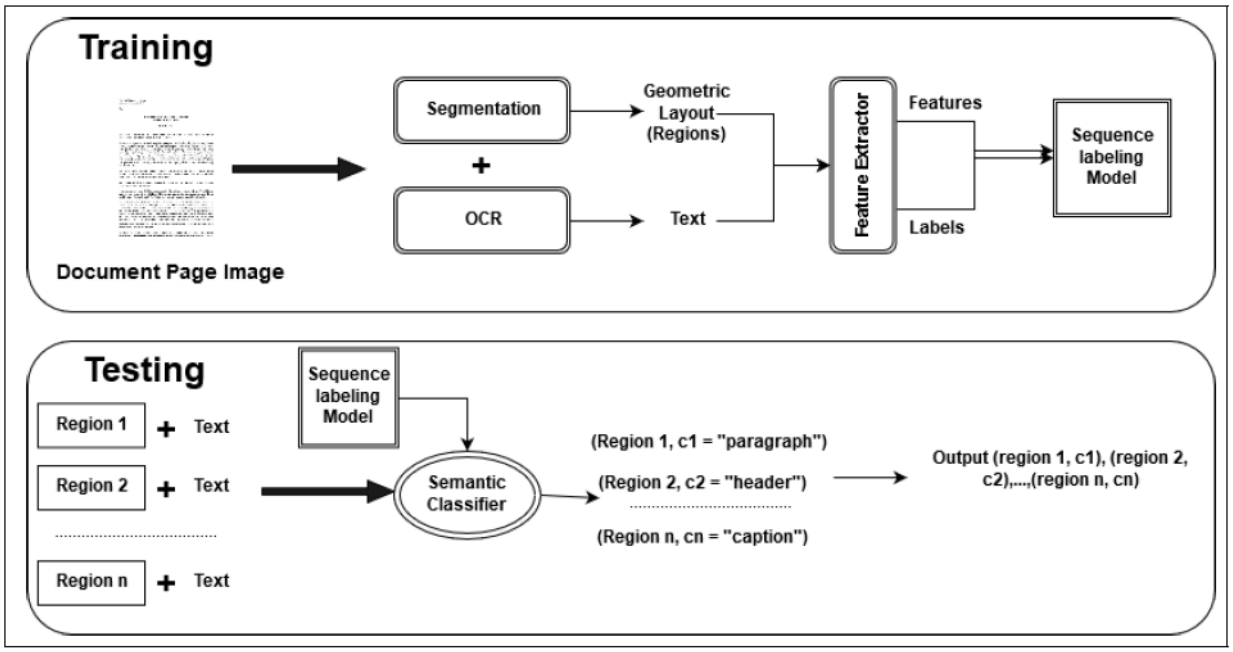



Figure 2.9 – Implementation architecture, including training and testing phases (Enendu et al., 2019)

### 2.3.3. Document text labeling

Holecek et al. (2019) admit that invoices should be treated as tables without outlines. The research work uses words and their positions as basic units of information, defining 5 inputs for the model:

picture of the whole document, coordinates of each word bounding box, sequential position of each word, one-hot chars (40 per word) and neighbor bounding boxes ids. As shown in Figure 2.10, each type of feature has its own pre-processing stage, being concatenated before the final layers which include a self-attention mechanism. The output is given by a sigmoidal layer that labels 35 classes. By comparing with a baseline in which a logistic regression was used, the authors present F1-score results usually over 90%. It should also be highlighted that the authors stated that a similar algorithm using a YOLO architecture has failed to achieve positive results.

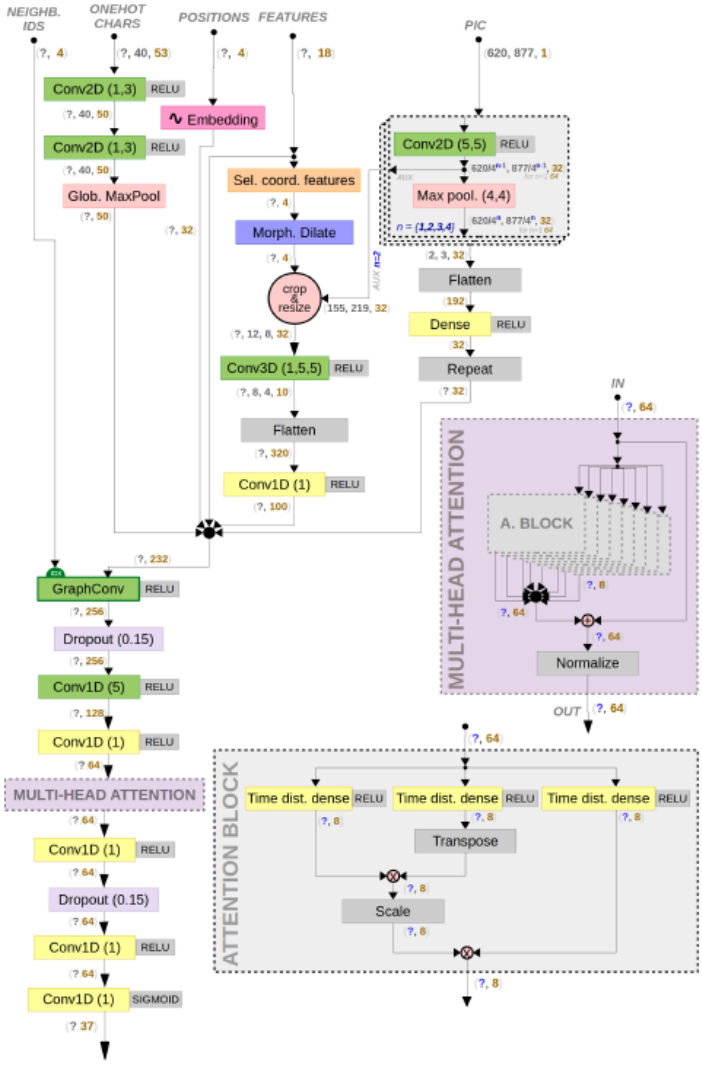



Figure 2.10 – Model architecture presented by Holecek et al. (2019)

Nguyen et al. (2021) presented two cases of applications used in Japanese companies to extract up to 24 types of entities from digital documents. Different combinations of models were experimented, consisting mostly in the use of pre-trained transformers models, BERT and ALBERT, with additional stacked layers that could be CNN, LSTM or BiLSTM, and finally a softmax or a CRF for output. The model is trained as a question-and-answer task, by feeding tags (name of the relevant fields) and their respective values. The results show that the combination BERT + CNN + softmax provides the best results, indicating that CNNs are the best option to capture the specific distribution of the information over the documents.

LayoutLM (Yiheng Xu et al., 2020) was developed based on the assumption that information extraction from documents should be treated as an NLP task with additional information which is specific to form-type or structured documents. That specific information is the bi-dimensional position (geometric embeddings) and style (visual embeddings) of the document elements. The main contribution of this work is the pre-trained model which is available for whoever wants to use it in the context of transfer learning. As LayoutLM is the model used in this research, a more detailed description of the model may be read in Section 4.

The second version of LayoutLM (Yang Xu et al., 2021) outperforms the first version due to the consideration of the visual embeddings in the pre-trained phase, instead of the fine tuning phase. Moreover, based on 1D sequential self-attention mechanism logic, a spatial aware self-attention mechanism is incorporated in the Transformer architecture so that the model understands the relative positional relationship between tokens.

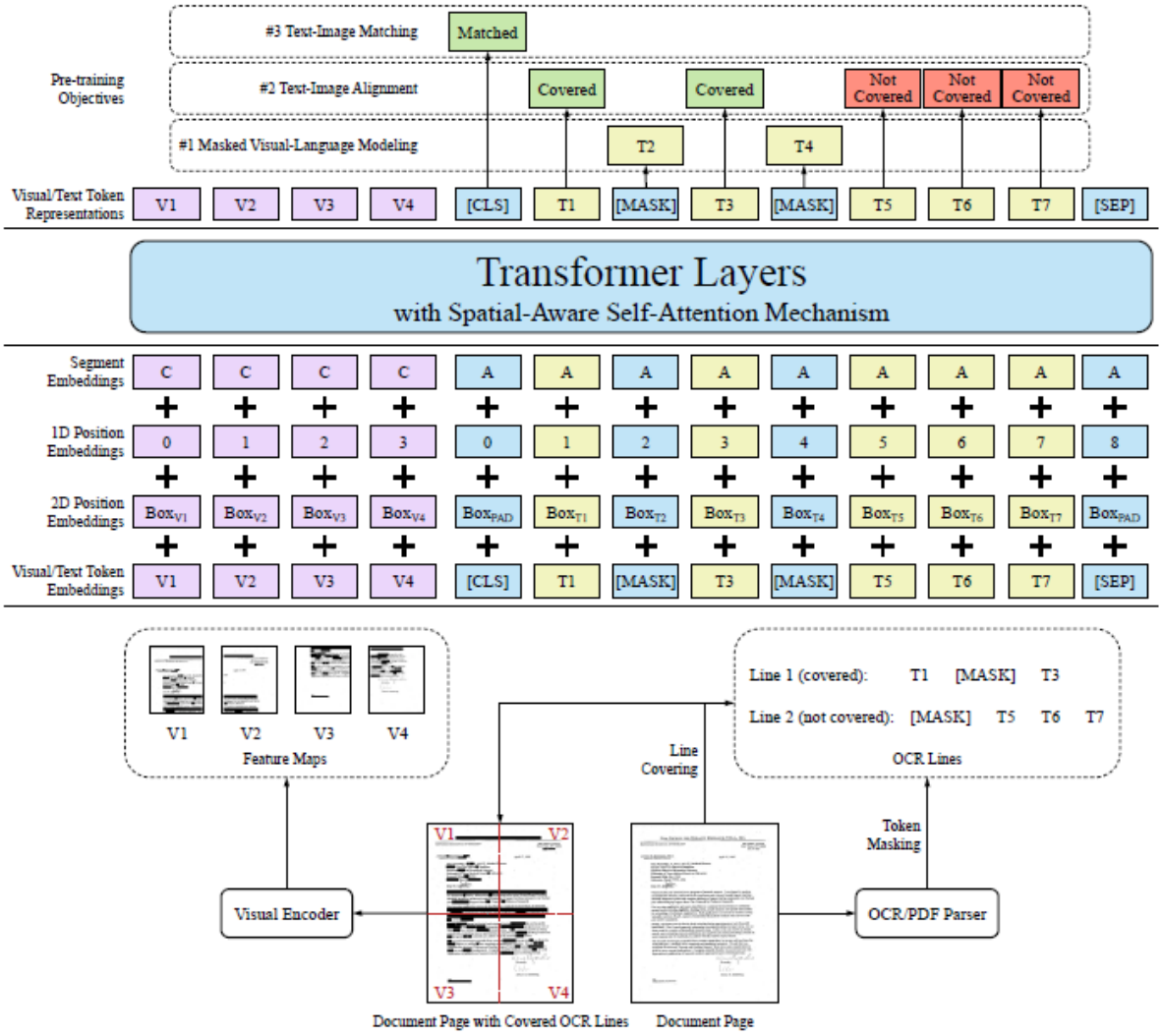



Figure 2.11 – Illustration of pre-training strategy and for LayoutLMv2

As presented in Figure 2.11, LayoutLM2 also incorporates two new pre-training tasks:

- new text-image alignment: consists of hiding random lines from the image while setting a classification for the tokens in those lines as “Covered”. Given a token and an image where the token is covered, the pre-trained model must classify it as “Covered”.
- text-image matching: consists of creating samples in which the text and the image do not match. Those samples are classified as negative, while regular samples (where text and image match) are classified as positive.

Finally, LayoutXLM (Yiheng Xu et al., 2021) has a very similar architecture to the one presented in LayoutLMv2. However, the authors pre-trained it in 30 million documents in 53 languages. A specific multi-lingual dataset was also created, and it was shown that the model results after fine-tuning outperform the state-of-the-art multi-lingual models.

# 3. CREATED DATASET

## 3.1. INTRODUCTION

Datasets with images of documents and text transcription are essential not only to benchmark document processing algorithms but also to train supervised models that focus on sequence labelling or named entity recognition tasks. This section presents a dataset comprising images of invoices and receipts, as well as text files with the transcription of relevant fields. Also, a detail of the annotated fields and a quantitative data analysis is provided.

## 3.2. VALUE OF THE DATA

- The presented dataset may be used for training models for document information extraction
- The dataset may be used to evaluate performance of an OCR processing model
- The dataset contains pictures of invoices and receipts and corresponding annotated relevant fields
- The documents are property of a private company. Such financial documents are usually not made available by private entities
- Some documents have low reading quality, resembling realistic cases

## 3.3. DATA DESCRIPTION

The dataset presented in this section is composed of images and annotation files regarding invoices and receipts. For each of the 813 documents, it contains one photography of the document and an annotation file with the text transcription of specific relevant fields.

The presented dataset was created to be consumed by an end-to-end pipeline, which includes converting the image into bounding boxes, text transcription using OCR, and the train of a transformer-based deep learning model. The model can be subsequently used to predict relevant fields from new invoices or receipts. Despite having been created to be used in a specific application and also the existence of datasets of invoices (Goldmann, 2019; Harley et al., 2015; Holecek et al., 2019; ICDAR, 2019; Jaume et al., 2019; Park et al., 2019), this data is made available and may be extremely useful to other researchers focused in similar tasks.

All the documents in the dataset belong to *Feels Like Home, Mediação Imobiliária* (FLH), a Portuguese company in the tourism industry which manages apartments for short renting. Every document represents a purchase made by FLH. There are two main groups of receipts and invoices in the company database. One represents scheduled and frequent purchases, for which the invoices sent by the provider are in a digital format (usually pdf files). The other group includes purchases made on street shops on the day-by-day by maintenance or check in teams. This dataset focuses on the second group, being characterized by invoices or receipts received in paper tickets. All the invoices were printed in the period between 2017 and 2019 and their language is Portuguese.

It should be stated that a considerable number of pictures have not only a low quality but also noise, blunts or are partially obstructed. Despite turning the document recognition more complex and less

accurate, this factor increases the level of reality of the dataset. Additionally, this factor gains more importance as more reliable hardware such as scanners are having less usage, while being replaced by smartphones, usually used in low light conditions.

### 3.4. DATA DETAILS AND DISTRIBUTIONS

The dataset is organized in two folders. Folder "1_Images" contains 813 image files in jpg format, with all the images correctly oriented. The annotations folder "2_Annotations_Json" contains 813 json files with names that match the same file names of the images folder. Each json object has the text of eight relevant fields. The description for each field is detailed in Table 3.1.

| Field Name | Description |
|---|---|
| company | Name of the provider/company that is selling the product |
| date | Date the document was issued |
| address | Address of the provider/company that is selling the product |
| total | Total amount of the document |
| invoice_number | Reference number of the invoice or receipt |
| NIF_buyer | Tax identification number of the buyer |
| iva_amount | VAT amount |
| NIF_seller | Tax identification number of the selling provider/company |

Table 3.1 – Description of each annotated field

Invoices or receipts usually follow specific rules, such as presenting a total value or an issue date. However, it is expected that some fields do not exist in some documents. Also, as explained in the next section, the annotated values are the ground truth of what is visible in the digital document. Thus, there are some annotations whenever the value was not visible or blurred in the image. Table 3.2 presents the number of non-empty values for each annotated field. As expected, the *total* and *date* fields are the most frequent.

| Field Name | Number of non-empty values |
|---|---|
| company | 755 |
| date | 802 |
| address | 771 |
| total | 812 |
| invoice_number | 793 |
| NIF_Buyer | 696 |
| iva_amount | 703 |
| NIF_seller | 763 |

Table 3.2 – Number of non-empty values for each field

Figure 3.1a shows the histograms for variable *total*. The low amounts of this variable confirm that the documents were related to small day-by-day maintenance or operation tasks. The variable *company*

has 235 unique values, indicating a high variety of document layouts in the dataset. Figure 3.1b shows the number of documents for the top 10 companies. Such samples represent 37% of the whole dataset, which denotes a high variation of layouts in the dataset. As seen in Figure 3.1c, all the documents were issued between 2017 and 2019, being the majority from 2019. The tax identification number for the buyer (*nif_buyer*) is highly concentrated in only two numbers, as seen in Figure 3.1d. This is explained by the fact that all the documents were obtained from one single company (Feels Like Home).

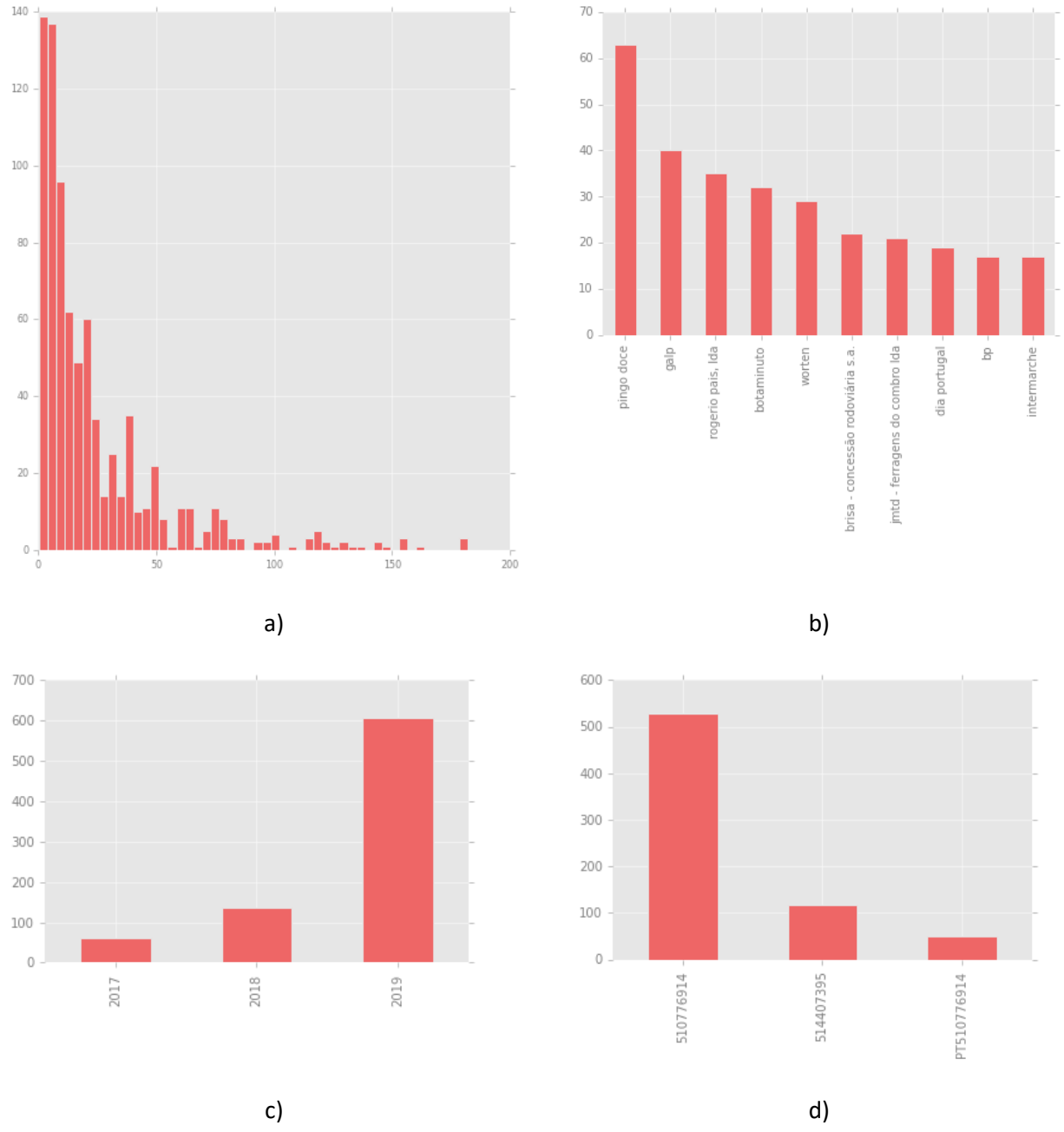
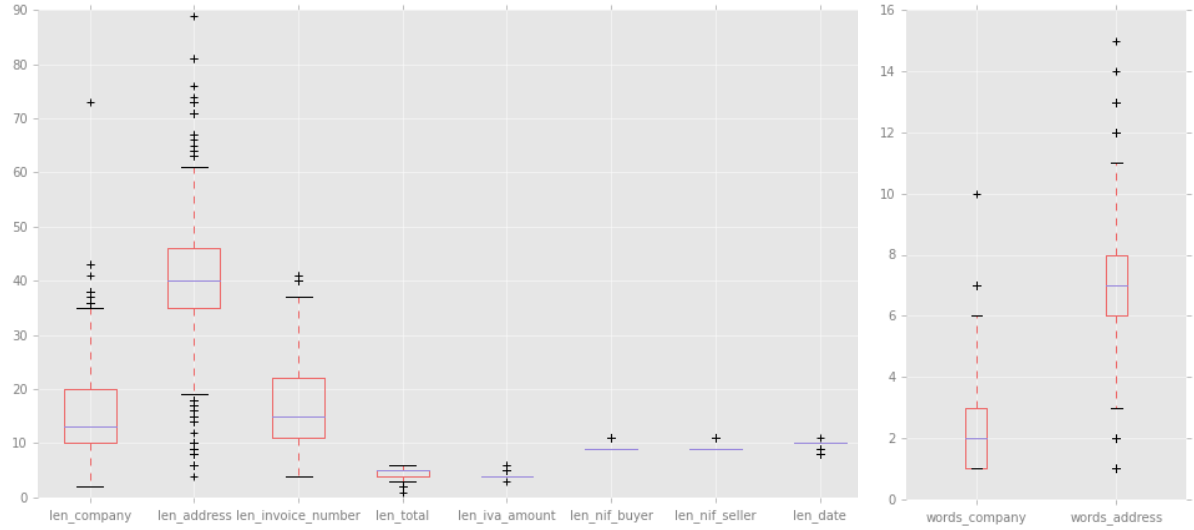



Figure 3.1 – Count of documents for values of variable: a) total (€) (not showing outliers). b) company (top 10). c) date (year issued). d) nif_buyer

Considering the number of characters – including white spaces – in each field, the box plots shown in Figure 3.2a indicate that, as expected, the *address* is the field with longer strings and high variances. As expected, the *company* name field has also a wide range of characters, as well as *invoice_number*. The numerical variables, *total* and *iva_amount*, present a small string size with medians of five and

four characters, respectively. The length of the variables *nif_buyer* and *nif_seller* confirms that Portuguese tax identification numbers have always nine digits or 11 digits (in the case the prefix "PT" is added to the number). Focusing on the number of words, Figure 3.2b shows a similar wide distribution for variable *address*, while the field *company* rarely exceeds six words.




a) b)

Figure 3.2 – Box plots representing for each field the distribution of: a) number of characters. b) number of words

Finally, Figure 3.3 presents the location of the annotated addresses. There are documents issued across the whole country, including the islands – Azores and Madeira.


Figure 3.3 – Location of every address annotated in the dataset

## 3.5. Experimental Design and Methods

As previously stated, this dataset was created to train a model for an end-to-end application. One of the preparation steps for the application is the transcription of text using OCR, so it was considered as

ground truth the text parts that could be read in the digitalized document and not the text that is typed in the physical document.

Figure 3.4a illustrates one example of such difference between values in the digital and physical document. In this example, while the ground truth for the company name (first line) is “Parque do Chão do Loureiro”, the annotated text is “ao do Loureiro”. The reasoning behind this decision is that the type of models that can be trained in the pipeline is based not only on text features but also on the 2D position of the words and image features of each bounding box. Consequently, it would be important to have the transcript text matching what is exactly in the digitalized image.

Finally, a specific limitation of the annotation should be highlighted. A few samples in the dataset do not show the total VAT value (*iva_amount*), but a table detailing the value per VAT rank, which is usually 0%, 6%, 13%, or 23%. In cases where VAT is detailed and the total value is not presented, the value corresponding to the highest rank was annotated. Figure 3.4b shows an example of a case where the total VAT value is not presented.


a) b)

Figure 3.4 – Examples of limitations in images. a) Digitalized document with part of the text covered. b) Green rectangle highlighting the table of VAT values detailed by VAT rank. No total VAT value is presented in the document

# 4. MODEL FUNDAMENTALS

## 4.1. TRANSFORMERS

Most of the state-of-the-art pre-trained models available currently use a Transformers architecture. The fundamental difference of this architecture is that is based on the concept of self-attention, as presented by Vaswani et al. (2017). This concept demonstrates the importance of capturing long-range dependencies and pairwise or higher-order interactions between elements. It also enables shorter computation paths and provides direct means to compare elements far across the input data (such as a pronoun and its antecedent in a sentence, or two sentences that refer to the same topic) (Bommasani et al., 2021).

The use of self-attention also reduces the total computational complexity per layer and increases the amount of computation that can be parallelized. According to the original paper, when compared to RNN's the complexity per layer decreases from $O\ (n.d^2)$ to $O\ (n^2.d)$ $n$ is the sequence length and $d$ is the representation dimension of embeddings. For tasks involving very long sequences, self-attention can be restricted to a neighborhood of size $r$ for the respective output position, decreasing the complexity to $O\ (r.n.d)$.

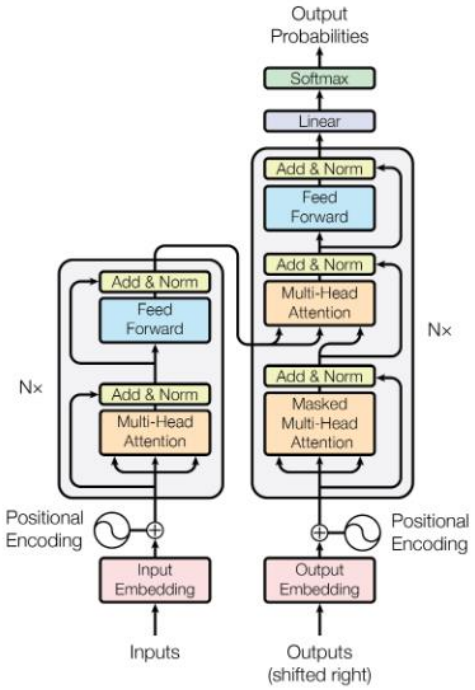



Figure 4.1 – Transformer model architecture (Vaswani et al., 2017)

Figure 4.1 shows the Transformer model architecture, which may be split into the parts described next.

**Input**

Words are transformed into tokens, which may be words, parts of words, or characters of the vocabulary list. Such tokens are represented by id's, which are then transformed in embedding vectors of fixed size (768 in this work). Similarly to word2vec embeddings (Mikolov et al., 2013), such vector represents a token regardless of its different meanings or senses.

Apart from general tokens, special tokens are defined for these models: [BOS] and [EOS] for beginning and end of a sentence, respectively; [UNK] for out-of-vocabulary tokens; [PAD] when a sequence size is extended to have general sequence size; [CLS] to initialize the sequence, and [MASK] to mask tokens (for pre-training purposes).

Each token position in the text sequence, which is given by an integer, is also considered. The position number is transformed into a vector following a sinusoidal function. Finally, both vectors (token embedding and position embedding) are summed and feed the first encoder.

**Encoders**

In the original paper, the encoder stack is composed of six layers. Each encoder layer comprises a multi-head attention layer and a feed-forward neural network.

Each head of the multi-head attention layer involves the computation of the so-called relations between tokens. Each attention head's output is given by Equation 4.1, where Q, K, and V are the Query, Key, and Value matrices, respectively. Those are obtained by multiplying the attention layer input with each of the Query, Key, and Value weight matrices. Such weight matrices are initially set randomly and trained with the rest of the model. By multiplying a softmax to matrix V, the layer guarantees that, for each token, the most important tokens are kept relevant while the less important are drown-out. Finally, the results of each attention head are concatenated and multiplied by another weights matrix. The output of the multi-head attention will be an embedding for each token.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right).V \qquad \text{(Eq. 4.1)}$$

The feed-forward neural network is applied to each position separately and identically, being a good candidate for parallelization. It consists of two linear transformations with a ReLU activation in between. It must be noted that both outputs of multi-head and feed-forward layers are normalized before being passed to the next layer.

**Decoders**

The decoders behave similarly to the encoders, outputting the tokens by sequence considering the previously predicted tokens. The main difference for encoders resides in the masking of the tokens which come after the predicted token, so they do not affect the decoder output. Consequently, each decoder includes a first layer that performs multi-head attention on the already predicted output tokens and a second layer that performs multi-head attention both on the previous layer and on the encoder output.

**Linear layer and softmax**

The output of the decoder layer is then passed in a feed-forward neural network adapted to the type of problem being solved. By using the softmax function, the possible outputs are scored and the output with higher probability is chosen.

## 4.2. BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is conceptually simple and empirically powerful obtaining state-of-the-art results in many NLP tasks (Devlin et al., 2019). It was mainly created to overcome the limitation of existing Transformers models that were only incorporating context from one direction of the text, being focused on a sequence-to-sequence task, limiting the possibility of other fine-tuning jobs.

BERT's architecture is similar to the one presented for Transformer models, while dropping the decoding part. The specific implementations on each part of the model are described as follows.

**Input**

Like Transformers, words are tokenized and translated to ids. Then, it defines the embedding vectors by means of WordPiece embeddings (Wu et al., 2016) with a vocabulary of approximately 30 000 tokens. To account for non-single sentence tasks such as question-answering, the segmentation embedding defines each sentence, separating the sentences with a specific tag [SEP]. Then, token and segment embeddings are summed with position embeddings, as seen in Figure 4.2.

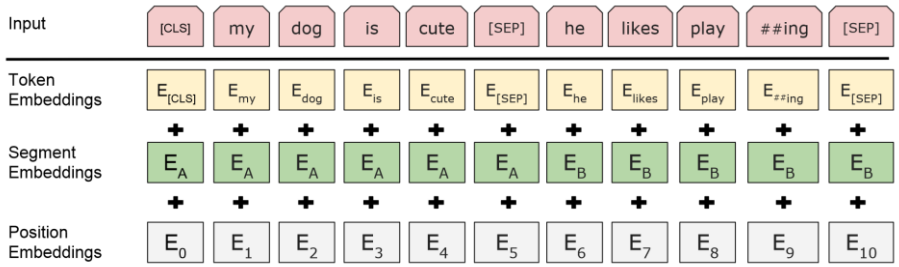



Figure 4.2 – BERT input representation (Devlin et al., 2019)

**Encoders**

BERT encoding part is equal to Transformers one, where a stack of encoders with multi-head attention, feed forward-network, and normalization is used. Nevertheless, two encoding stack sizes are presented. The base model uses 12 encoding layers, representations with 768 dimensions, and 12 attention-heads, resulting in 110 million parameters. The large model uses 24 encoding layers, representations with 1024 dimensions, and 16 attention-heads, resulting in 340 million parameters.

**Output**

When using BERT to apply fine-tuning, task-specific datasets are used to update all parameters in an end-to-end way. Consequently, such as in Transformers' architecture, another feed-forward network with a softmax may be plugged in the output.

On the other hand, BERT may be also used for feature extraction, given that the embeddings obtained by using the pre-trained parameters are already strongly influenced by the relation between words. In this case, the embeddings are directly used as features for other models to be trained for specific tasks.

## 4.3. LAYOUTLM ARCHITECTURE

LayoutLM uses BERT's architecture as a backbone. Nevertheless, the authors have created it assuming that when it comes to visually rich documents, there is much more information that can be encoded into the pre-trained model. In addition to token, segment, and position embeddings, LayoutLM adds two extra input embeddings described below.

**2D position embedding**

The spatial position within the document may be given by the coordinates of each token bounding boxes. Using a coordinate system with the origin in the top-left corner of the image, a bounding box can be defined by ($x_0$, $y_0$, $x_1$, $y_1$), where ($x_0$, $y_0$) corresponds to the position of the upper left and ($x_1$, $y_1$) represents the position of the lower right.

The coordinates are normalized in both directions to fit in a [0, 1000] range and then transformed into vectors, following a similar logic to the one used for sequential position embeddings.

**Image embedding**

Using the coordinates of the bounding boxes, the image of the document is split into each token's image. Then, features for each token image are obtained by running a Faster R-CNN. The features for the whole image are also obtained, so the embedding of the [CLS] tag is also enriched.

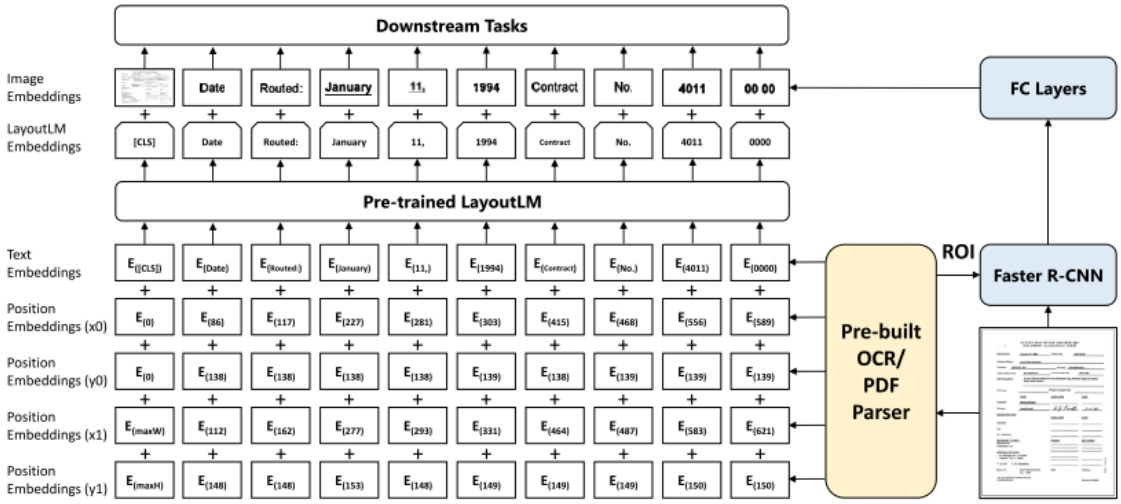



Figure 4.3 – LayoutLM input and downstream representation (Yiheng Xu et al., 2020)

Figure 4.3 shows the input and a suggestion of downstream representation of the model. It should be noted that the first version of LayoutLM (Yiheng Xu et al., 2020) does not include the image embeddings in the pre-trained model. Such a feature was only considered in the second version of the model (Yang Xu et al., 2021). Consequently, as this work was developed while only the first version was available, the image embeddings were left out of scope.

### 4.4. LAYOUTLM PRE-TRAINING

The model is pre-trained on the IIT-CDIP test collection with 11 million scanned document images. The training is based on two tasks presented next.

**Masked Visual-Language Model (MVLM)**

> Self-supervised learning that consists in randomly choosing 15% of tokens and requesting the model to predict them. Of this group of tokens, 80% are masked using a special tag [MASK], 10% are replaced by a random token and 10% are left unchanged. As the 2D position embedding is used, the model learns not only the language context but also its relationship with space context.

**Multi-label Document Classification (MDC)**

> The documents provided in the pre-training dataset are labeled with multiple categories. Therefore, such categories are used to pre-train the model to predict such tags and therefore improve document-level representation.

Following the BERT backbone, LayoutLM also presents a base and a large model. Both models are initialized for pre-training with the corresponding pre-trained BERT weights. Base and large model take, respectively, 80h and 170h to pre-train.

It should also be noted that the first version of LayoutLM was pre-trained only in an English corpus. In opposite, a multi-lingual model was released after the development of this work, being therefore out-of-scope.

### 4.5. LAYOUTLM FINE-TUNING

Compared to pre-training, fine-tuning is a relatively inexpensive process (Devlin et al., 2019). Similar to BERT, LayoutLM is fine-tuned by initializing with the pre-trained weights, plugging in a specific model for the downstream task, and training the parameters end-to-end using labeled data. This way, weights are obtained both from unlabeled and labeled data, in a semi-supervised (Dai & Le, 2015) manner.

The original paper (Yiheng Xu et al., 2020) presents three document image understanding for which LayoutLM may be fine-tuned: form understanding, receipt understanding, and document image classification. The first two tasks require the classification of every token, while the former classifies the document based on the document-representative [CLS] token.

This work focuses on the receipt understanding task, which requires the training samples to provide the label that matches each token. The number of possible labels is fixed, as they are defined with the problem. The output of the encoder stack is down streamed to a linear layer which will result in a vector with the number of possible labels of the problem. Following softmax, the label with the highest probability is chosen to classify the corresponding token.

# 5. METHODOLOGY

## 5.1. PIPELINE

Having the objective of building an end-to-end system that fits either training or predicting purposes, a flexible pipeline is implemented following the schema presented in Figure 5.1.

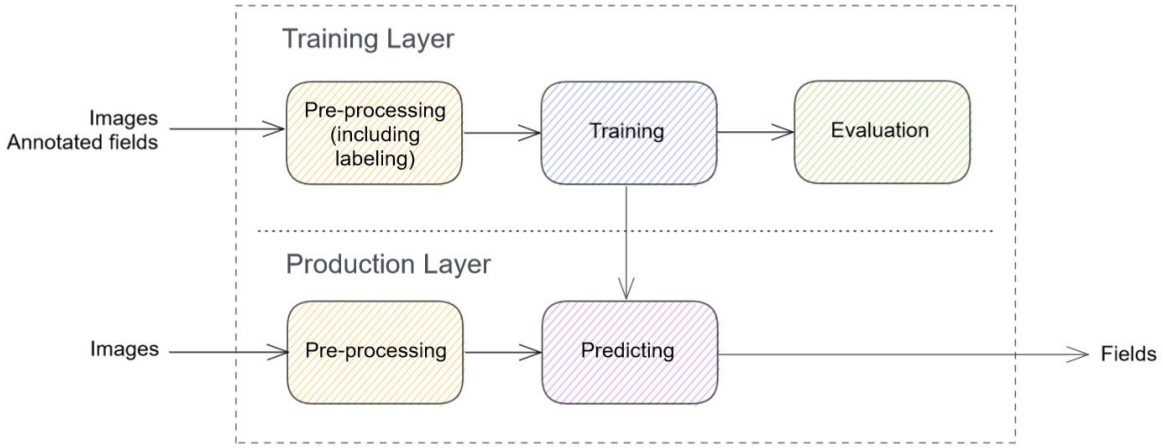



Figure 5.1 – Simplified schema of the implemented pipeline

On the training layer, the algorithm inputs images of documents and annotations of the relevant fields (see examples in Figure 5.2). Then, it preprocesses the images, obtains the text fields, automatically labels the text based on the annotations, and prepares both train and evaluation files in the format expected by the model. Following, it trains the model with the provided samples and finally, evaluates the model.



```
{
    "company": "jmtd - ferragens do combro lda",
    "date": "23/04/2019",
    "address": "calçada do combro nº3 1200-110 lisboa",
    "total": "6,50",
    "invoice_number": "A/30692",
    "nif_buyer": "510776914",
    "iva_amount": "1,22",
    "nif_seller": "513 350 535"
}
```


Figure 5.2 – Image of a document in the dataset and annotation of its relevant fields in JSON format

On the production layer, it inputs images, preprocesses, obtains text from images, and prepares files for the model. In sequence, the pipeline predicts using the trained model and outputs the relevant fields in the same format as the annotation example presented in Figure 5.2.

## 5.2. PRE-PROCESSING

The pre-processing, represented in Figure 5.3, consists of three stages, each of which uses a specific python package. The first stage, *PicImprover*, consists in preparing and improving the image, having the following features:

- Image rotation using the PIL package (*Python-Pillow/Pillow: The Friendly PIL Fork (Python Imaging Library)*, n.d.), as the OCR process needs to receive the image in the correct orientation.
- Image cropping and skewness reduction, using OpenCV package (*Opencv/Opencv-Python: Automated CI Toolchain to Produce Precompiled Opencv-Python, Opencv-Python-Headless, Opencv-Contrib-Python and Opencv-Contrib-Python-Headless Packages.*, n.d.) to find the image contours and convert to a rectangular shape.
- Image adaptive thresholding, which increases contrast, using the Skimage package (Van Der Walt et al., 2014).

In the second stage, *PicOCR*, both text and bounding boxes are extracted from the image using an OCR processor. Two OCR tools are available in this stage: Tesseract, through a python package (Hoffstaetter, n.d.), and Google Vision (GV), through an API request gateway. Although both tools show a good performance, only GV is used in this work mostly because it requires less parametrization. A drawback of GV is that it involves costs. Therefore, to avoid the repetition of OCR processing of images, a cache feature is implemented.

Finally, the third stage of the pre-processing phase (*PreModel*) converts the received text and bounding box data to the format accepted by the model. Moreover, it has an essential role when run in training mode, as it is also responsible for labeling the text extracted with OCR based on the provided annotations.

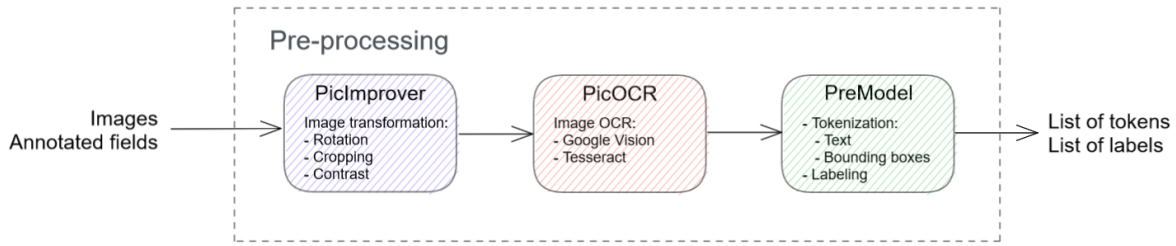
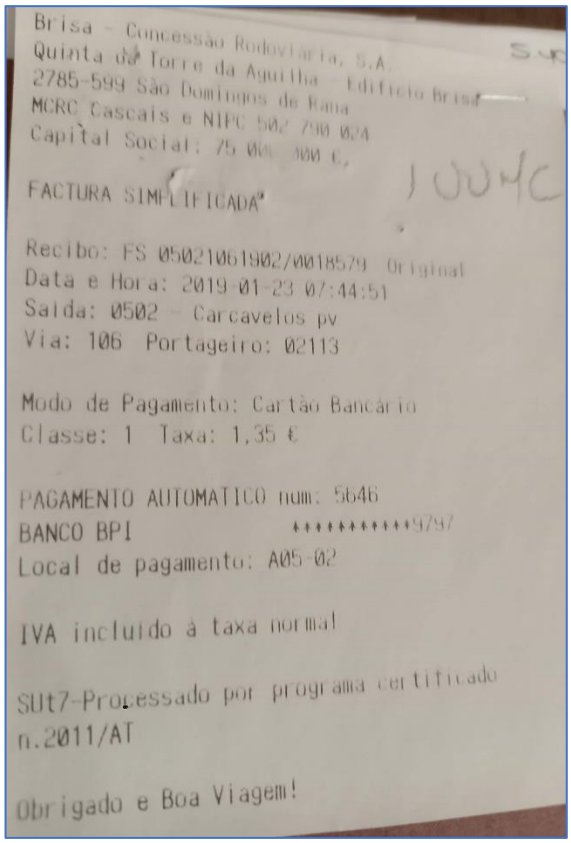



Figure 5.3 – Simplified schema of the pre-processing stage

The main difficulty of labeling the samples resides in the fact that the OCR processor efficiency depends on image quality. For blurred, skewed, or noisy images, the text obtained by OCR may not match the annotated text, as seen in the example of Figure 5.4.




a)

```
Brisa - Concessdu Rodovilla, SIA
Quinta da Torre da Aguilha Edificio Brise
2785-599 São Domingos
MC RC Cascais e NIPC 502 780 524
Capital Social: 75 mm
FACTURA SIMPLIFICADA
Recibo: FS 05021061902/60 18579 Original
Data e Hora: 2019-01-23 07:44:51
```

b)

```
{
    "company": "brisa - concessão rodoviária, S.A.",
    "date": "2019-01-23",
    "address": "quinta da torre da aguilha - edificio brisa",
    "invoice_number": "FS 05021061902/0018579",
    "total": "1,35",
    "nif_buyer": "",
    "nif_seller": "502790024",
    "iva_amount": ""
}
```

c)

Figure 5.4 – Example of a document with poor quality. a) Image of the document. b) Part of the text obtained by OCR with highlighted errors. c) Annotated fields, which do not match exactly the content of the OCR text

Being impossible to label the text based on exact matches, an algorithm is implemented to search in the OCR text for n-grams similar to the ones on the annotated fields. The way of measuring similarity is based on the type of field.

For numeric fields – total amount and tax amount – it searches for n-grams that are convertible to a number and then searches the annotated number, as seen in the example of Figure 5.5. If there is a match between compared numbers, then the text is labeled.

| OCR Text | Search n-grams convertible to number | Search number equal to annotation (1.35) | Label original n-gram |
|---|---|---|---|
| ... | ... | ... | ... |
| Classe: | | | Classe: |
| 1 | 1 | 1 | 1 |
| Taxa: | | | Taxa: |
| 1, | 1 | 1 | 1, TOTAL |
| 35 | 35 | 35 | 35 TOTAL |
| PAGAMENTO | 1.35 | 1.35 | PAGAMENTO |
| AUTOMATICO | | | AUTOMATICO |
| num: | | | num: |
| 5646 | 5646 | 5646 | 5646 |
| ... | ... | ... | ... |

Figure 5.5 – Example of steps of the algorithm that labels numeric fields

For the remaining fields, it searches for n-grams and compares them with the annotated text, based on *SequenceMatcher* method of *difflib* python package (*Difflib — Helpers for Computing Deltas —*

*Python 3.10.0 Documentation*, n.d.). This method computes a similarity score between two strings (Ratcliff & Metzener, 1988). One of its main advantages resides in outputting a score between 0 and 1. On the other hand, this is a computationally expensive algorithm, a factor that was not considered problematic in this work, given that it only runs on the training layer.

Figure 5.6 shows an example of how the algorithm works. For an annotation n-gram with N words, it searches all n-grams with size in the range [N/2, N*2], so it accounts for cases of having extra spaces or concatenated words. For each n-gram found, it computes the similarity score and chooses the n-gram with the highest score. To consider the n-gram with the best similar score as a match, the score must be higher than a certain threshold, which is 0,75 in this work. As a match occurs, the sample is labeled with the corresponding class.

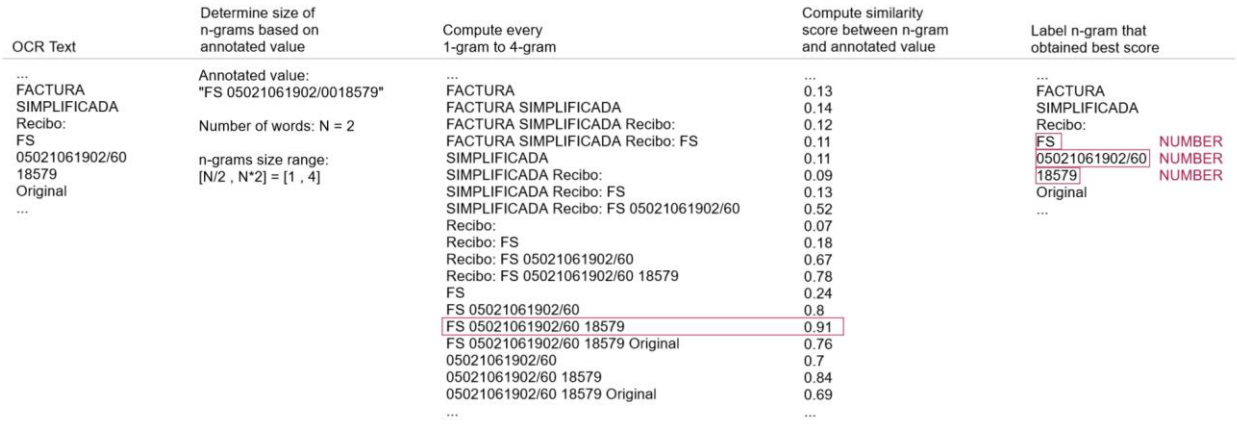



Figure 5.6 – Example of steps of the algorithm that labels string fields

This labeling methodology is an essential feature of this work. Consequently, its advantages and limitations demand some discussion, as it overcomes a big proportion of the errors on the OCR process but not all. When a part of the text obtained by OCR is largely different from the real text, the algorithm does not consider it as a match and therefore does not identify that class in the text. This would lead to both training and validation samples missing some of their classes. Consequently, when running validation tests, the results can be either overestimated, when the model does not predict a label that the labeling algorithm missed, or underestimated when the model predicts a label correctly that the algorithm missed.

Table 5.1 shows the performance of the labeling algorithm for the eight fields of the FLH dataset – company name, company address, document date, total amount, seller tax number, buyer tax number, document number, and vat amount – and for the FLH and SROIE datasets when labeling four fields – company name, company address, document date and total amount. The first row of Table 5.1 indicates how many classes would be labeled by using an algorithm based solely on exact matches (similarity score equal to 1). 1862 fields would not be labeled, representing 30,5% of the fields, justifying the need to use a different approach.

The last three rows of Table 5.1 show the results of the labeling algorithm based on a similarity score. For the FLH dataset with eight classes, the algorithm misses 5,6% of the classes, while for the cases with four fields, it misses 3,6% and 2,2% for the FLH and SROIE datasets, respectively.

| Dataset | Num of documents | Num of classes labeled | Num of blank classes | Num of missing classes | Num of matching classes | % missing classes (excluding blank) |
|---|---|---|---|---|---|---|
| FLH (only exact matching) | 813 | 8 | 409 | 1862 | 4233 | 30,5% |
| FLH | 813 | 8 | 409 | 339 | 5756 | 5,6% |
| FLH | 813 | 4 | 112 | 114 | 3026 | 3,6% |
| SROIE | 727 | 4 | 3 | 64 | 2841 | 2,2% |

Table 5.1 – Results of labeling algorithm for FLH and SROIE datasets

One the one hand, the figures show that this is not an exact algorithm, leading to some error as parts of the text are left unlabeled. On the other hand, some advantages of this algorithm must be highlighted. First, it requires a much simpler annotation process, based only on annotating the text that belongs to each field, eliminating the need to use graphical interfaces to annotate the bounding boxes. Second, it is easily adaptable to most real cases in which there is already data available. If there is the objective to fine-tune the model for a new company, it is much more probable that the company has a database with images and the annotated text – in excel spreadsheets or accountant software –, when compared with a dataset with images and their bounding boxes. Finally, the results of the algorithm, like the ones presented in Table 5.1, may be indicative of the quality of the images, and therefore, of the performance of the model to a specific dataset.

The samples are labeled using an IO tagging format. The choice of another format would require further study. As stated in the literature, there is no optimal format beforehand; the applicability of each format depends on factors such as language, type of text and frequency of labels (Alshammari & Alanazi, 2021). Given that labels do not repeat frequently in each sample, an IO tagging format is thought to be the best and simpler option.

## 5.3. MODEL AND RUN CONFIGURATIONS

The code to run the model and the pre-processing packages are available in GitHub repositories. Consequently, both are easily cloned into a Google Colab notebook, which allows training using graphical processing units (GPU). Although is not a user's choice, in this work much of the processing-intensive tasks are processed using a Tesla P100-PCIE-16GB.

To avoid heavier computations, the pre-trained model chosen is LayoutLM Base, with 110 million parameters. The model hyper-parameters are mostly based on the ones presented in the LayoutLM presentation article (Yiheng Xu et al., 2020). Also, the model training and evaluation resorts to a script made available by the authors of LayoutLM in which PyTorch is used. After running a brief parametrization, the original parameters are kept, except for the number of attention heads, which was switched from 12 to 8. Table 5.2 shows the parameters used in every computation.

For consistency, all the tests are run using hold-out, splitting the datasets with 75% of training samples and 25% of evaluation samples. The batch size is always 4, both for train and evaluation. Although being a simple and fast strategy, hold-out is very dependent on the train/test split. Nevertheless, it is admitted (and confirmed in the results section) that the dataset is big enough to reduce this effect.

| Parameter | Value |
|---|---|
| architecture | LayoutlmForTokenClassification |
| attention_probs_dropout_prob | 0,1 |
| hidden_act | gelu |
| hidden_dropout_prob | 0,1 |
| hidden_size | 768 |
| type_vocab_size | 2 |
| initializer_range | 0,02 |
| intermediate_size | 3072 |
| layer_norm_eps | 1e-12 |

| Parameter | Value |
|---|---|
| max_2d_position_embeddings | 1024 |
| max_position_embeddings | 512 |
| vocab_size | 30522 |
| model_type | bert |
| num_attention_heads | 8 |
| num_hidden_layers | 12 |
| output_past | true |
| pad_token_id | 0 |

Table 5.2 – LayoutLM parameters

## 5.4. DATASETS

Two datasets are used in this work. FLH is created specifically in the scope of this work and presented on Section 3. SROIE (Scanned Receipts OCR and Information Extraction) dataset (ICDAR, 2019) 727 files follow the same structure of FLH dataset, with image files and their corresponding annotation files. The two main differences between datasets are: SROIE documents are mostly in English and SROIE dataset only annotates four relevant fields – company name, company address, total amount and invoice date.

## 5.5. TYPLESS TRAINING AND PREDICTION

The software Typless is used in this work to compare the models with a baseline commercial solution. From other available commercial tools, the choice over this software resides in the fact that it offers a trial with API access, which is essential to input hundreds of documents. There is no available content about how Typless works, just that it is an "Easy-to-use invoice OCR REST API powered by AI".

The software requires training of the models prior to any extraction tasks, which indicates that some machine learning-based model is used. Three endpoints are used:

- add-document – to add a document image file and the relevant fields, feeding the training dataset,
- start-training – to instruct the software to start training the model,
- extract-data – to predict relevant fields from new image documents.

Another endpoint – add-document-feedback – could also be used to improve model results, yet it is ignored in this work.

## 5.6. EVALUATION AND METRICS

To be in line with the research results of LayoutLM (Yiheng Xu et al., 2020), this work uses mostly F1 score to evaluate results, resorting to the python package Seqeval (Nakayama, 2018) for the computations. Considering there is no specific class with more importance than others, the option "micro" is used to average the score over the classes, meaning that the evaluation is made at the word level. F1 is computed for a multi-class problem, yet it excludes the correct predictions on the class "other".

To compute the F1 score, both ground-truth and predicted text must be available, which is true for all the computations made with LayoutLM. Nevertheless, there are other algorithms from which only the text value from each class is obtained, such as the commercial software Typless. To overcome the impossibility of using F1 to compare the performance with Typless, Word Error Rate (WER) and Character Error Rate (CER) are also used, by means of Jiwer package (*Jitsi/Jiwer: Evaluate Your Speech-to-Text System with Similarity Measures Such as Word Error Rate (WER)*, n.d.).

Both WER and CER derive from Levenshtein distance, so the lower the score, the better is the prediction. The error rates are computed by summing the number of substitutions, deletions, and insertions and dividing by the number of words or characters, depending on which – WER or CER – is being computed. While WER is always a number between 0 and 1, CER may take values over 1 in cases when a high number of insertions occur. Basically, it may be said that WER and CER indicate the percentage of words or characters, respectively, that are incorrectly predicted.

# 6. RESULTS

## 6.1. BASELINE COMPARISONS

In order to evaluate the gains on using LayoutLM for predicting the relevant fields on invoices, two baseline computations are made, using BERT architecture and a commercial software. Both comparisons are made by training and testing on the FLH dataset (813 samples) and classifying 8 fields: company name, company address, total amount, document date, seller's NIF, buyer's NIF, document number, and tax amount.

The first baseline model uses a BERT architecture. This is an obvious choice for a baseline as LayoutLM integrates extra features into the original BERT architecture. Table 6.1 shows the results obtained both for the baseline and LayoutLM models. In accordance with the results presented by Yiheng Xu et al. (2020), there is an improvement when using LayoutLM, from 82% to 90%. Given that both models use similar architectures, it may be stated that the use of 2D features marks an expressive difference for this kind of problem. It is also noticeable that most of the improvement is based on precision. This means that while both models are quite effective in correctly predicting the real classes, BERT model tends to wrongfully predict more labels than the real ones.

| Model | Precision | Recall | F1 |
|---|---|---|---|
| LayoutLM | 0,87 | 0,93 | 0,90 |
| BERT | 0,73 | 0,90 | 0,82 |

Table 6.1 – Comparison between BERT baseline model and LayoutLM

Table 6.2 shows the performance detailed by class. Predictions of address and company name achieve a similar performance. On the other six classes, LayoutLM outperforms BERT, with larger differences in the numerical values, total, and tax (iva) amount. Such differences in numeric categories may be explained by the fact that LayoutLM takes advantage of the fact that both values usually appear on the right side of the document.

| Class | F1 (LayoutLM) | F1 (BERT) |
|---|---|---|
| address | 0,95 | 0,93 |
| company | 0,82 | 0,83 |
| date | 0,85 | 0,77 |
| iva_amount | 0,81 | 0,61 |
| nif_buyer | 0,96 | 0,75 |
| nif_seller | 0,9 | 0,86 |
| number | 0,9 | 0,81 |
| total | 0,9 | 0,73 |

Table 6.2 – Comparison between BERT baseline model and LayoutLM. Performance for each class

The second baseline model is based on a commercial software, Typless. As explained in the previous section, it is not possible to compute sequence evaluation metrics (precision, recall, F1) when using

the software. Consequently, the comparison is made through the average word error rate (WER) and average character rate (CER) of the test samples. One unexpected behavior on Typless is that the "Address "class was never predicted in the test samples. To have this issue into consideration, another set of tests is made by considering every one of the eight classes, except the "Address".

Table 6.3 shows that error rates are considerably lower when using LayoutLM. Even when removing the "Address" class, the results with LayoutLM outperform the ones using Typless.

| Model | Avg CER | Avg WER | Notes |
|---|---|---|---|
| LayoutLM | 0,25 | 0,45 | Predicting 8 classes |
| Typless | 0,58 | 0,53 | Predicting 8 classes |
| LayoutLM | 0,24 | 0,38 | Predicting 7 classes (removing Address class) |
| Typless | 0,45 | 0,48 | Predicting 7 classes (removing Address class) |

Table 6.3 – Comparison between Typless baseline model and LayoutLM

## 6.2. APPLICATION ON FLH AND SROIE DATASET

Taking advantage of having another dataset available (SROIE), computations are made to understand how useful it is to use both datasets. However, it must be noted that SROIE is annotated only for four fields: company name, company address, total amount, and document date. Consequently, every model trained in this section is only trained for those four fields, ignoring other fields that are available in the FLH dataset.

First, the modeling and evaluation are made separately for each dataset, as shown in Table 6.4. Both models obtain results over 90% for F1, which would be considered as a state-of-the-art result. The better result obtained with the SROIE dataset may be explained by two factors. First, the dataset images have much better quality, showing almost no skewness, blurring, or elements such as staples. This certainly improves OCR quality, resulting in predictions closer to ground truth. Second, the version of the LayoutLM model used in this study was pre-trained only in an English corpus. As known, documents of the FLH dataset are fully in Portuguese while documents of the SROIE dataset have most of their text in English.

A model pre-trained in English being so effective on a Portuguese dataset may be explained by the very specific structure of the text in an invoice, which is less dependent on the language when compared to text written freely.

| Job # | Train | Test | Num samples | Precision | Recall | F1 |
|---|---|---|---|---|---|---|
| 6 | FLH | FLH | 813 | 0,87 | 0,93 | 0,91 |
| 7 | SROIE | SROIE | 727 | 0,95 | 0,97 | 0,96 |

Table 6.4 – Modeling on FLH and SROIE dataset separately

## 6.3. EFFECT OF PREDICTING ON A DIFFERENT DATASET

To understand the impact of the data used for training, two additional tests are done. First, the model is trained for every 813 FLH samples and evaluated on 626 SROIE samples. In opposition, the second model is trained for SROIE samples and evaluated for the whole FLH dataset. The results presented in Table 6.5 confirm the strong influence of the data used for fine-tuning. When the dataset used for training has different characteristics of the dataset used for evaluation, the F1 scores drop to unacceptable values near 60%. This is an issue that must not be ignored when deploying the model into production, as predictions on data with different characteristics will be much less effective.

| Job # | Train | Test | Num samples | Precision | Recall | F1 |
|---|---|---|---|---|---|---|
| 8 | FLH | SROIE | 1540 | 0,88 | 0,50 | 0,64 |
| 9 | SROIE | FLH | 1540 | 0,69 | 0,55 | 0,61 |

Table 6.5 – Modeling and evaluating on different datasets

## 6.4. EFFECT OF MIXING DATASETS

To simulate a situation in which both datasets would be available for training, three more jobs are defined. The first (job 10) trains on whole SROIE dataset together with 75% of the FLH dataset and evaluates on the remaining 25% of the FLH dataset. The second (job 11) trains on whole FLH dataset together with 75% of the SROIE dataset and evaluates on the remaining 25% of the SROIE dataset. The third job (job 12) trains on a batch that would include 75% of the SROIE dataset and 75% of the FLH dataset, evaluating on a batch with 25% of both datasets. The results shown in Table 6.6 indicate that no improvements are achieved when compared to training the datasets separately (see Table 6.4 – jobs 6 and 7).

On job 10, in which the model is trained on whole SROIE dataset together with 75% of the FLH dataset, representing 1235 training samples, the obtained F1 score on the evaluation of 305 FLH samples is 91%. This is the same value obtained when only the FLH dataset is used for the train (see Table 6.4 – job 6). The same analogy is valid for job 11, as it presents similar results to job 7.

By mixing two datasets in the evaluation stage, as done in job 12, it is noticeable that the obtained score is between the ones obtained in jobs 6 and 7 (see Table 6.4). This leads to the idea that the models have reached their full capacity and consequently using data from another dataset is useless or even has a negative impact.

| Job # | Train | Test | Num samples | Precision | Recall | F1 |
|---|---|---|---|---|---|---|
| 10 | FLH + SROIE | FLH | 1540 | 0,89 | 0,93 | 0,91 |
| 11 | FLH + SROIE | SROIE | 1540 | 0,95 | 0,97 | 0,96 |
| 12 | FLH + SROIE | FLH + SROIE | 1540 | 0,93 | 0,97 | 0,95 |

Table 6.6 – Using mixed FLH and SROIE samples for training

## 6.5. EFFECT OF DATASET SIZE

The following question arises from the previous sub-section: if there is a limit above which adding extra data to train models turns into small or no gains, what is that limit? This question is especially useful to have an estimation of how many samples need to be annotated when training the model for a new dataset.

To focus on this question, additional jobs are defined to train and evaluate the models in batches. As more data is available, the batch size increases as well as the F1 score for the evaluation dataset, a relation which may be named as learning curve. In every case, the available batch was split into train and evaluation set in a proportion of 75% to 25%.

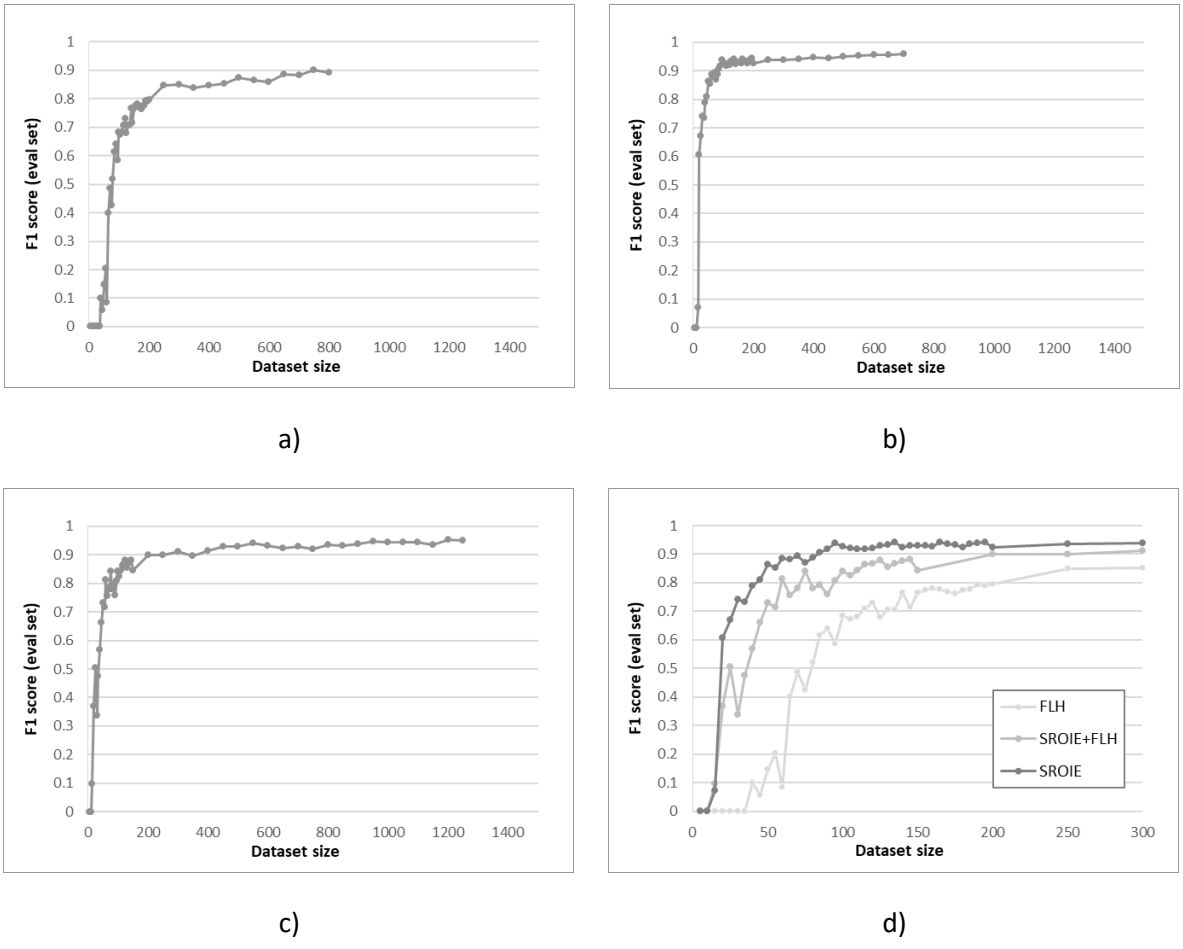



Figure 6.1 – Learning curves of model fine-tuned for different datasets. a) FLH dataset. b) SROIE dataset. c) Mix of FLH and SROIE datasets. d) Detail of three previous combinations for first 300 samples

Figure 6.1 shows the evolution of the F1 score with increasing batch sizes for different datasets. Figure 6.1a and Figure 6.1b show learning curves for the FLH and SROIE datasets, respectively, trained and evaluated separately. Figure 6.1c plots the curve for a dataset that is a random mixture of both FLH and SROIE datasets.

In all three cases, the learning curve may be split into three stages. The first occurs in cases with very few data to train and evaluate, being represented by values of F1 near 0%. The second stage is

characterized by a high inclination in the evolution of the score with batch size. It is the phase in which the available data has a strong impact on the model performance. Finally, the third stage represents a near-plateau, in which extra data adds little or none to model performance.

As expected, the results obtained with the SROIE dataset reach a higher F1 value. It may also be highlighted that a higher F1 score shows a relation with a steeper dataset size/score rate, meaning that when a model learns faster it tends to reach higher scores. This could be indicative of the further application of this model to different datasets. The third stage for each of the plots suggests that when using LayoutLM, both FLH and SROIE datasets have more data available than needed to reach very good scores.

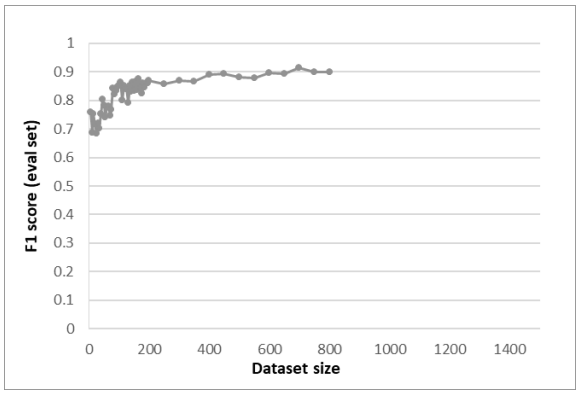



Figure 6.2 – Learning curve for the model trained and evaluated for FLH dataset and previously fine-tuned for SROIE dataset

A different scenario is illustrated in Figure 6.2. In this case, only the FLH dataset is considered for training and evaluation, yet the model is previously fine-tuned with the SROIE dataset. Only with a few FLH dataset samples, the model scores are much higher than 0%. However, based on the results presented in Table 6.5, one dataset is not suitable to make prediction on another, so the model scores are still far from the best performance, with values near 70%-75%. Notwithstanding, as the model is also trained for an increasing FLH dataset size, the performance increases.

It should also be highlighted that the steepness on the relation between the size of dataset and performance is much lower in this case when compared with Figure 6.1. This means that a previously tuned model can predict on different datasets, yet it learns slower on the new data.

## 6.6. DISCUSSION AND CONSIDERATIONS FOR PRODUCTION ENVIRONMENT

The results presented above may bring to the discussion relevant aspects about deploying to production the models trained in this work.

Undoubtfully it is worth it to use the LayoutLM model in production when compared to BERT models or even commercial software. The extra features given by the bi-dimensional position of the bounding boxes provide a relevant upgrade in performance. There is obviously some overhead in including extra features, mostly related to pre-processing of the bounding boxes. Nevertheless, as most OCR tools already provide the bounding boxes together with the text, the overhead is considered minimal. It is proven that LayoutLM is also suitable for modeling documents in the Portuguese language, enabling the next steps for implementation of the model on Feels Like Home (FLH) internal accountant software.

When in production, it must be considered that different clients will probably predict documents with different characteristics. Table 6.5 presents results on predicting using a model trained with data that had different characteristics, showing that a sudden drop in performance occurs. The usage of data with different characteristics should be done carefully and user expectations on the prediction of the first samples must be managed. Also, the results of this work leave the possibility of implementing two scenarios on how models should be trained. The first scenario would assume a generalist model, which is being trained with every data despite its origin. In the second scenario, each client has its own model, that is trained with only its data. The advantages and drawbacks of each scenario in short- or long-term performance are summarized in Table 6.7.

The evolution of the score with available data shown in Figure 6.1 and Figure 6.2 indicates that, when in production, an online learning algorithm would need to be implemented for the model to reach acceptable performance when data with different characteristics is used. Such an algorithm uses new predicted samples together with user feedback to retrain the model.

However, the presented results show that an online algorithm only has a strong impact in the first phase, showing almost no improvements after reaching a certain limit. Consequently, when deploying this model into production, one should analyze the possibility of implementing an online algorithm against an option without online learning but with an initial setup of the model that would include a fine-tune with a certain amount of client data. Table 6.8 presents a brief description of the advantages and drawbacks of both scenarios.

The presented results are essential to define a strategy to implement the model into production. This will be done in a proof of concept that will be a plug-in of Feels Like Home internal accountant and financial software. Regarding the scenarios presented in Table 6.7, the client-specific model approach will be used, training only on the FLH dataset. This will allow the model to predict eight relevant fields, instead of only the four fields that a model trained with both datasets could predict. As for the scenarios described in Table 6.8, as there is already a model trained for FLH data, it is considered that the initial setup is already complete. Consequently, an online learning algorithm is thought not to be worth the costs of its implementation at the moment.

| Advantages and drawbacks of each scenario | Based on |
|---|---|
| **Generalist model scenario** | |
| **Advantages** | |
| Although far from top performance, prediction on the first samples of a new dataset would return fair results. | See jobs 8 and 9 in Table 6.5 and Figure 6.2. |
| Deep learning models are known for their ability to model data with different characteristics. Training a model with data with different characteristics should make it more robust. | |
| **Drawbacks** | |
| If a new dataset has good quality images, the top performance of the model might be truncated because of the worst quality samples used in training. | See job 7 on Table 6.4 and job 12 on Table 6.6. Job 7 reaches the same score with half the data. |
| The number of training samples needed for the model to reach a good performance would not be smaller when compared to a client-specific model scenario. | Figure 6.1a and Figure 6.2 show that in both cases ~200 samples would be needed to achieve a good score. |
| Model cannot be fine-tuned for specific fields that did not exist in the previous training data | |
| **Client-specific model scenario** | |
| **Advantages** | |
| Models adapt specifically predicting the fields that are relevant for each client, which might improve the top performance of specific clients | See job 7 in Table 6.4 and job 12 on Table 6.6. Job 7 reaches the same score with half the data. |
| Data is treated separately for each client. This may be relevant in certain data compliant scenarios, when for example data of clients must not be stored in different countries or regions. | |
| In the need to re-fine tune the model, less data is used, requiring less computational power. | |
| **Drawbacks** | |
| The model is not fine-tuned previously, so performance on predicting first samples might be zero | See Figure 6.1a and Figure 6.1b. |

Table 6.7 – Advantages and drawbacks of each modeling scenario

| Advantages and drawbacks of each scenario | Based on |
|---|---|
| **Implement an online learning algorithm** | |
| Advantages | |
| The model would always be learning and adapting to new characteristics. | See in Figure 6.1 and Figure 6.2 that the model performance is always improved as more data is provided. |
| Drawbacks | |
| Ground-truth depends on user feedback. Depending on the user, error on feedback may be high, reducing model quality | |
| For the first samples, the results may be poor or fair, which can lead to user frustration. | See in Figure 6.1 and Figure 6.2 that a few hundred of samples are needed to reach top performance. |
| Costs of implementing and maintaining an extra algorithm | |
| After reaching a certain threshold, the model performance gains are minimal with increasing training data | See near-plateau reached in Figure 6.1 and Figure 6.2. |
| **Initial setup without learning algorithm** | |
| Advantages | |
| The model would be set up for the client's specific data. After this initial setup stage model would be near top performance. | |
| Setup requires new annotated samples. However, by using the pipeline presented in this work, only pictures and text of relevant fields would be required. In many companies' databases, this is data is already available. | |
| Drawbacks | |
| If the client starts using data with different characteristics, a new setup must be done. | See in Figure 6.2 that even already trained, the model needs a few hundred samples to reach top performance on new data. |
| Setup requires new annotated samples, which involve costs in cases these do not exist yet. | |

Table 6.8 – Advantages and drawbacks of each learning scenario

# 7. CONCLUSIONS AND FUTURE WORK

This thesis presents a machine learning end-to-end pipeline for the extraction of relevant fields from invoices and receipts. The pipeline comprehends two layers. The training layer receives batches with images of documents and their corresponding annotations and fine-tunes a Transformers-based pre-trained model – LayoutLM – for a sequence labeling task. The production layer inputs images and predicts the relevant fields.

The images are pre-processed extracting the whole document text and bounding boxes using OCR. Also, in the training layer, the OCR text is labeled using an algorithm that searches parts of the text equal or highly similar to the annotations. Considering that LayoutLM is pre-trained both with additional spatial features, the model is fine-tuned with text and its bi-dimensional position within the document.

In addition, a dataset is made available to the community (FLH dataset). It comprehends 813 images of invoices and receipts, as well as the annotated text with eight relevant fields – company name, company address, document date, document number, buyer tax number, seller tax number, total amount, and tax amount.

The model is compared with two baseline models, one BERT model and a commercial software. The model is fine-tuned both with the created dataset and another available dataset, which allows extracting valuable knowledge from the results and consequently defining a strategy for deploying the pipeline into production.

The four objectives purposed for this work were successfully completed:

- A dataset is made available with data that largely overcomes the need for model fine-tuning.
- Overall, the implemented pipeline is a functional proof of concept and after minor engineering implementations (creating endpoints, testing, etc.) will be ready for production.
- An NLP Transformers-based model specifically pre-trained for document comprehension tasks proves to perform accurately in the data provided.
- A strategy for model deployment is defined. The model will be fine-tuned based solely on the FLH dataset and there is no need to implement an online learning algorithm.

During the development of each thesis' section, many interesting conclusions were registered and may be compiled in the following items.

**Literature review**

There are two periods of time in which developments were made in the field of extracting relevant fields from documents. The first is related mostly to algorithms that rely on pre-defined layouts, requiring user intervention and showing no flexibility to new layouts. The second, developed in the past few years, is based on deep learning techniques, mostly by using NLP and CV techniques.

The very recent models that use an architecture based on Transformers achieve state-of-the-art results on document comprehension tasks. LayoutLM, available in two versions, uses both spatial and image

features and is suitable to the objective of this thesis. A multi-language LayoutLM model is also available.

**Created dataset**

By creating a dataset to be used in an end-to-end pipeline important questions arise. Which information should be annotated, the whole document text, the relevant text, and bounding boxes, only the relevant text? The option of annotating only relevant text became evident, not only by its simplicity but also because that is the type of data that is most probable to obtain from further companies if there is the need to train the model to new data. This approach has some limitations: as the image is converted to text by using an OCR, there is a chance that the OCR text does not match the annotated data, which can lead to errors in labeling the text to train the model.

The dataset includes 813 images of invoices and receipts. There is a high variability in the data obtained, namely on the seller companies, having the top 10 companies representing only 37% of the whole dataset. This indicates the high variation of layouts used. As expected, the buyer tax number has a much lower variability – only two different tax numbers in the whole dataset – which is explained by the fact that the whole dataset belongs to one company.

**Methodology**

From within different possibilities, the end-to-end pipeline is based on two main tasks – transcription of text from an image using an OCR tool and training the model using a Transformers-based model. Despite being essential for the task, the inner workings of the OCR are left out of the scope of the thesis by using an external tool – Google Vision API or Pytesseract.

The pre-processing of the data is complex, including treatment of image, OCR, and transformation of annotated data into text labels for training purposes. The labeling, based on the annotated fields, is achieved using a search algorithm based on a similarity score. Such an algorithm proves to be very effective in overcoming the differences in OCR text and annotations, by reducing the rate of non-matched annotations from 30,5% to 5,6%. It is considered that the errors resulting from using it are overcome by its practicality. This algorithm also shows a good application to measure both the quality of the images and the OCR tool.

**Results**

The results of applying LayoutLM to the created dataset show unequivocally its high performance, when compared to BERT and a commercial software. The use of LayoutLM both for FLH and SROIE datasets have F1 scores higher than 90%. By analyzing the effect of mixing the datasets, it is noticeable that the performance drops when predicting using a model trained with another dataset.

The learning curves suggest a relation between the steepness of the learning rate and the top performance obtained in the dataset. Also, it is shown that for both datasets, their size largely overcomes the number of samples needed to achieve the top performance of the model. A rule of thumb may be defined, in which around 100-200 samples are enough to fine-tune this model for this kind of data.

The results presented and a discussion is schematized in two tables, so a strategy is defined for deployment in production as a plugin in FLH internal accountant software. Taking advantage of the fact that specific client-relevant fields may be extracted and that the model is already fine-tuned with high performance on predictions, a model fine-tuned only on FLH data will be used. Also, it is argued that there is no need to use online learning tools for such cases.

**Research limitations**

The main limitations of this research are considered to be:

- The use of hold-out as a training and testing evaluation. As known, by using this strategy, the results achieved depend more on the way the split is made. Given the dataset size and the conclusions taken from the analysis of the learning curves, it is thought that hold-out is enough to have confidence in the results. Nevertheless, this hypothesis was not confirmed.
- The number of datasets. While it is considered very relevant to have two different datasets for the evaluation of the model, the conclusions taken cannot be generalized to the model.
- The capacity on predicting correctly is too dependent on OCR or image quality.
- Model parametrization was simplistic and might influence results.

**Future work**

There is an expectation of developing further work on this subject. From an academic perspective, this thesis is expected to be reduced in size and transformed into a scientific paper to submit to a reference journal. Also, a data paper will be submitted after minor adaption of Section “Created dataset”. From a production perspective, the pipeline will be improved with engineering features, such as an API or performance monitoring, and deployed in an FLH server to be plugged in the FLH internal software.

Further research would be very useful to improve understanding of different fields approached in this thesis, such as data annotation, pre-processing, or model improvement. The most relevant contributions would be:

- A detailed comparison between different annotation strategies to confirm that the simplifications introduced in this study compensate for the increase in error when compared to using the whole document text as ground truth. This study can be made with the SROIE dataset, as the OCR ground truth, bounding boxes, and relevant fields are made available.
- Create more datasets including samples with different characteristics. Fine-tuning the model for more datasets would allow generalizing the conclusions about the model. Such datasets could even be created synthetically from the existing datasets by applying for example exchange of words within the document or translation of documents.
- Perform the same calculations using cross-validation.
- Perform a deeper comparison and parametrization between OCR tools: Google Vision API and Pytesseract.

- Incorporate in the pre-processing step an algorithm to estimate image or OCR quality. This algorithm would allow defining confidence in the prediction.
- Perform the same calculations using the second version and the multi-lingual version of LayoutLM.

## BIBLIOGRAPHY


Alshammari, N., & Alanazi, S. (2021). The impact of using different annotation schemes on named entity recognition. *Egyptian Informatics Journal*, *22*(3), 295–302. https://doi.org/10.1016/j.eij.2020.10.004

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2021). *On the Opportunities and Risks of Foundation Models*. http://arxiv.org/abs/2108.07258

Borisyuk, F., Gordo, A., & Sivakumar, V. (2018). Rosetta: Large scale system for text detection and recognition in images. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 71–79. https://doi.org/10.1145/3219819.3219861

Cesarini, F., Gori, M., Marinai, S., & Soda, G. (1998). Informys: A flexible invoice-like form-reader system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(7), 730–745. https://doi.org/10.1109/34.689303

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, *12*, 2493–2537. http://arxiv.org/abs/1103.0398

Dai, A. M., & Le, Q. V. (2015). Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems*, *2015-January*, 3079–3087. http://arxiv.org/abs/1511.01432

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 4171–4186. https://github.com/tensorflow/tensor2tensor

*difflib — Helpers for computing deltas — Python 3.10.0 documentation*. (n.d.). Retrieved October 26, 2021, from https://docs.python.org/3/library/difflib.html

Enendu, S., Zylab, J. S., Zylab, J. S., Hiemstra, D., & Theune, M. (2019). *Predicting Semantic Labels of Text Regions in Heterogeneous Document Images*. https://research.utwente.nl/en/publications/predicting-semantic-labels-of-text-regions-in-heterogeneous-docum

Goldmann, L. (2019). *Layout Analysis Groundtruth for the RVL-CDIP Dataset*. https://doi.org/10.5281/ZENODO.3257319

Harley, A. W., Ufkes, A., & Derpanis, K. G. (2015). Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, *2015-November*, 991–995. http://arxiv.org/abs/1502.07058

Hoffstaetter, S. (n.d.). *madmaze/pytesseract: A Python wrapper for Google Tesseract*. Retrieved October 26, 2021, from https://github.com/madmaze/pytesseract

Holecek, M., Hoskovec, A., Baudis, P., & Klinger, P. (2019). *Table Understanding in Structured Documents*. https://doi.org/10.1109/icdarw.2019.40098

ICDAR. (2019). *ICDAR 2019 Robust Reading Challenge on Scanned Receipts OCR and Information*

*Extraction*. International Conference on Document Analysis and Recognition. https://rrc.cvc.uab.es/?ch=13

Jaume, G., Kemal Ekenel, H., & Thiran, J.-P. (2019). *FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents*. https://doi.org/10.1109/icdarw.2019.10029

*jitsi/jiwer: Evaluate your speech-to-text system with similarity measures such as word error rate (WER)*. (n.d.). Retrieved October 27, 2021, from https://github.com/jitsi/jiwer/

Kavasidis, I., Pino, C., Palazzo, S., Rundo, F., Giordano, D., Messina, P., & Spampinato, C. (2019). A saliency-based convolutional neural network for table and chart detection in digitized documents. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11752 LNCS*, 292–302. https://doi.org/10.1007/978-3-030-30645-8_27

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539

Lee, J., Hayashi, H., Ohyama, W., & Uchida, S. (2019). Page segmentation using a convolutional neural network with trainable co-occurrence features. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1023–1028. https://doi.org/10.1109/ICDAR.2019.00167

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 7871–7880. http://arxiv.org/abs/1910.13461

Liu, W., Yuan, X., Zhang, Y., Liu, M., Xiao, Z., & Wu, J. (2020). An End to End Method for Taxi Receipt Automatic Recognition Based on Neural Network. *Proceedings of 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2020*, 314–318. https://doi.org/10.1109/ITNEC48623.2020.9084712

Liu, X., Meng, G., & Pan, C. (2019). Scene text detection and recognition with advances in deep learning: a survey. *International Journal on Document Analysis and Recognition*, *22*(2), 143–162. https://doi.org/10.1007/s10032-019-00320-5

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. http://arxiv.org/abs/1907.11692

Mao, S., Rosenfeld, A., & Kanungo, T. (2003). Document structure analysis algorithms: a literature survey. In T. Kanungo, E. H. Barney Smith, J. Hu, & P. B. Kantor (Eds.), *Document Recognition and Retrieval X* (Vol. 5010, pp. 197–207). SPIE. https://doi.org/10.1117/12.476326

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 16). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. http://ronan.collobert.com/senna/

Nagy, G. (2000). Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 38–62. https://doi.org/10.1109/34.824820

Nakayama, H. (2018). *chakki-works/seqeval: A Python framework for sequence labeling evaluation(named-entity recognition, pos tagging, etc...)*. https://github.com/chakki-works/seqeval

Nguyen, M. T., Le, D. T., & Le, L. (2021). Transformers-based information extraction with limited data for domain-specific business documents. *Engineering Applications of Artificial Intelligence*, *97*, 104100. https://doi.org/10.1016/j.engappai.2020.104100

*opencv/opencv-python: Automated CI toolchain to produce precompiled opencv-python, opencv-python-headless, opencv-contrib-python and opencv-contrib-python-headless packages.* (n.d.). Retrieved October 26, 2021, from https://github.com/opencv/opencv-python

Park, S., Shin, S., Lee, B., Lee, J., Surh, J., Seo, M., & Lee, H. (2019). *CORD: A Consolidated Receipt Dataset for Post-OCR Parsing*.

*python-pillow/Pillow: The friendly PIL fork (Python Imaging Library)*. (n.d.). Retrieved October 26, 2021, from https://github.com/python-pillow/Pillow

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (n.d.). *Language Models are Unsupervised Multitask Learners*. https://github.com/codelucas/newspaper

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*, 1–67. http://arxiv.org/abs/1910.10683

Ratcliff, J. W., & Metzener, D. E. (1988). Pattern Matching: The Gestalt Approach. *Dr. Dobb's Journal*, 46.

Schreiber, S., Agne, S., Wolf, I., Dengel, A., & Ahmed, S. (2017). DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, *1*, 1162–1167. https://doi.org/10.1109/ICDAR.2017.192

Stahl, C. G., Young, S. R., Herrmannova, D., Patton, R. M., & Wells, J. C. (2018). *DeepPDF: A Deep Learning Approach to Extracting Text from PDFs (Conference) | OSTI.GOV*. https://www.osti.gov/biblio/1460210-deeppdf-deep-learning-approach-extracting-text-from-pdfs

Tsujimoto, S., & Asada, H. (1990). Understanding multi-articled documents. *Proceedings - International Conference on Pattern Recognition*, *1*, 551–556. https://doi.org/10.1109/icpr.1990.118163

Van Der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). Scikit-image: Image processing in python. *PeerJ*, *2014*(1). https://doi.org/10.7717/peerj.453

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-December*, 5999–6009. https://arxiv.org/abs/1706.03762v5

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. https://doi.org/10.1145/1390156.1390294

Wahl, F. M., Wong, K. Y., & Casey, R. G. (1982). Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing*, *20*(4), 375–390. https://doi.org/10.1016/0146-664X(82)90059-4

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T.,

Kazawa, H., … Dean, J. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. http://arxiv.org/abs/1609.08144

Xu, Yang, Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., Lu, Y., Florencio, D., Zhang, C., Che, W., Zhang, M., & Zhou, L. (2021). LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL) 2021*, 2579–2591. http://arxiv.org/abs/2012.14740

Xu, Yiheng, Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). LayoutLM: Pre-training of Text and Layout for Document Image Understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, *20*, 2020. https://doi.org/10.1145/3394486.3403172

Xu, Yiheng, Lv, T., Cui, L., Wang, G., Lu, Y., Florencio, D., Zhang, C., & Wei, F. (2021). *LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding*. http://arxiv.org/abs/2104.08836

Zhang, Z., Zhang, C., Shen, W., Yao, C., Liu, W., & Bai, X. (2016). Multi-oriented Text Detection with Fully Convolutional Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 4159–4167. https://doi.org/10.1109/CVPR.2016.451