TOMÁS
MARTINS

Anonimização Automatizada de Contratos Jurídicos em Português

Automated Anonymization of Legal Contracts in Portuguese

Universidade de Aveiro
2022

TOMÁS
MARTINS

Anonimização Automatizada de Contratos Jurídicos em Português

Automated Anonymization of Legal Contracts in Portuguese

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Manuel Neto Vieira, Professor Auxiliar convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professora Doutora Susana de Jesus Mota, Professora Auxiliar
Universidade de Aveiro

vogais / examiners committee

Professor Doutor Luís Filipe Pinto de Almeida Teixeira, Professor Auxiliar
Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Professor Doutor António José Ribeiro Neves, Professor Auxiliar
Universidade de Aveiro

agradecimentos / acknowledgements

Gostaria primeiramente de agradecer aos meus pais, ao meu avô e ao meu tio, por toda a paciência, apoio e motivação em todo este percurso. Sempre me apoiaram quando precisei, não me deixaram cair nos momentos mais em baixo e sem eles sem dúvida que esta dissertação não seria possivel.

Queria também agradecer à minha namorada, a pessoa que mais me ajudou não só durante a elaboração desta dissertação, mas durante o meu percurso académico. Para além de força, carinho e apoio, também me ensinou muito e sempre me incentivou a seguir o rumo certo e a fazer decisões acertadas. Esteve sempre disponivel para as minhas dúvidas e problemas de última hora e sempre deu o seu melhor para me poder ajudar.

Um agradecimento especial a todos os meus colegas de matricula. Um grande obrigado, especialmente a todos com quem tive o prazer de realizar projetos, apresentações ou até noitadas de estudo em conjunto, pois cada um ensinou-me algo que não só me tornou um melhor aluno, como uma melhor pessoa. Um agradecimento especial ainda para o meu colega Rodrigo Rosmaninho, representante da nossa matrícula durante estes cinco anos, que sempre defendeu os interesses dos colegas da melhor maneira, à frente dos seus próprios, e que facilitou bastante este percurso.

Um agradecimento à Universidade de Aveiro e ao Departamento de Eletrónica, Telecomunicações e Informática por terem sido a minha casa e me proporcionarem todos os recursos para a minha aprendizagem.

Agradeço também à IMPIC por nos ter proposto este desafio, ter providenciado o data set e ter esclarecido todas as dúvidas relativamente ao processo de anonimização.

Queria também agradecer ao co-orientador desta dissertação José Vieira, ao colaborador Ricardo Ribeiro e em especial ao orientador António Neves por esta oportunidade e por terem acreditado em mim para a elaboração deste trabalho. Todo o conhecimento transmitido, toda a paciência e orientação, foram aspetos fulcrais para a elaboração da dissertação.

Palavras Chave

reconhecimento ótico de caracteres, processamento de linguagem natural, processamento de imagem, spacy, aprendizagem automática, reconhecimento de entidades nomeadas, documento, português, pré-processamento, privacidade.

Resumo

Com a introdução do Regulamento Geral de Proteção de Dados, muitas organizações ficaram com uma grande quantidade de documentos contendo informações públicas que deveriam ser privadas. Dado que estamos a falar de quantidades bastante elevadas de documentos, seria um desperdício de recursos editá-los manualmente. O objetivo desta dissertação é o desenvovimento de um sistema autónomo de anonimização de informação sensível em contratos escritos na língua Portuguesa.

Este sistema utiliza a Google Cloud Vision, uma API de OCR, para extrair qualquer texto presente num documento. Como existe a possibilidade desses documentos serem pouco legíveis, é feito um pré-processamento de imagem através da biblioteca OpenCV para aumentar a legibilidade do texto presente nas imagens. Entre outros, foi explorada a aplicação de algoritmos de binarização, correção da inclinação e remoção de ruído.

Uma vez extraído o texto, este será interpretado por uma biblioteca de nlp, neste projeto optou-se pelo uso do spaCy, que contém um pipeline português treinado com os conjuntos de dados WikiNer e UD Portuguese Bosque. Esta biblioteca não permite apenas uma identificação bastante completa da parte do discurso, mas também contém quatro categorias diferentes de reconhecimento de entidade nomeada no seu modelo. Para além do processamento efetuado com o recurso à biblioteca de spaCy, e uma vez que a língua portuguesa não tem um grande suporte, foram implementados alguns algoritmos baseados em regras de modo a identificar outros tipos de informação mais especifica como número de identificação e códigos postais. No final, as informações consideradas confidenciais são cobertas por um retângulo preto desenhado pelo OpenCV através das coordenadas retornadas pelo OCR do Google Cloud Vision e será gerado um novo PDF.

Keywords

binarization, contract, document, google cloud vision, image processing, machine learning, named entity recognition, natural language processing, nlp, ocr, optical character recognition, portuguese, preprocessing, privacy, processing, pytesseract, spacy.

Abstract

With the introduction of the General Data Protection Regulation, many organizations were left with a large amount of documents containing public information that should have been private. Given that we are talking about quite large quantities of documents, it would be a waste of resources to edit them manually. The objective of this dissertation is the development of an autonomous system for the anonymization of sensitive information in contracts written in Portuguese.

This system uses Google Cloud Vision, an API to apply the OCR tecnology, to extract any text present in a document. As there is a possibility that these documents are poorly readable, an image pre-processing is done using the OpenCV library to increase the readability of the text present in the images. Among others, the application of binarization, skew correction and noise removal algorithms were explored.

Once the text has been extracted, it will be interpreted by an NLP library. In this project we chose to use spaCy, which contains a Portuguese pipeline trained with the WikiNer and UD Portuguese Bosque datasets. This library not only allows a very complete identification of the part of speech, but also contains four different categories of named entity recognition in its model. In addition to the processing carried out using the spaCy library, and since the Portuguese language does not have a great support, some rule-based algorithms were implemented in order to identify other types of more specific information such as identification number and postal codes. In the end, the information considered confidential is covered by a black rectangle drawn by OpenCV through the coordinates returned by Google Cloud Vision OCR and a new PDF is generated.

Contents

List of Figures

List of Tables

Glossary

CNN	Convolutional Neural Network
CRF	Conditional Random Field
CTC	Connection Temporal Classification
DT	Decision Tree
FN	False Negatives
FP	False Positives
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HMM	Hidden Markov Models
ICR	Intelligent Character Recognition
LSTM	Long Short-Term Memory
ME	Maximum Entropy
NER	Named Entity Recognition
NIF	Número de Identificação Fiscal
NLP	Natural Language Processing
NN	Neural Network
OCR	Optical Character Recognition
PIL	Python Imaging Library
POS	Part-of-speech
RF	Random Forest
RNN	Recurrent Neural Network
SVM	Support-vector machine
TN	True Negatives
TP	True Positives

CHAPTER 1

Introduction

This chapter contains a brief introduction to this work's main objectives, problems, structure and technology used to solve these problems.

1.1 Motivation

With the internet being more important each day in our lives, millions of documents are being held public in multiple platforms with easy access from everybody. This is useful as people can review contracts, bills, medical appointments, and several other paperwork that otherwise could have been lost, or simply too fatiguing to obtain. However, in 2018, the General Data Protection Regulation (GDPR) was created, with the aim to protect the personal data of each citizen and entity in the European Union and with these regulations, an enormous amount of documents became invalid as they contain certain information that now must be private. Since amending these documents manually would not be possible to execute in a satisfactory amount of time, and would result in a lot of financial, labor and time waste, it has become a necessity to automate this process.

The topic for this dissertation was proposed by IMPIC, the managing entity of the Portal BASE and the body responsible for regulating public contracts. It is also responsible for setting the good practices rules on public contracts and analyzing complaints and reports from different entities on the application of public procurement rules [1].

Many of these documents start out in paper format but are quickly converted to digital format for easy access, using Optical Character Recognition (OCR) technology. This can be used, for example, by a bank in order to easily analyze the checks deposited by the users or even for other applications like converting a physical book to audio and help blind people with reading. In figure 1.1 we can observe the use of the OCR technology by a bank to read and validate a check.

[1] `https://www.base.gov.pt/Base4/pt/o-portal/quem-e-quem/`

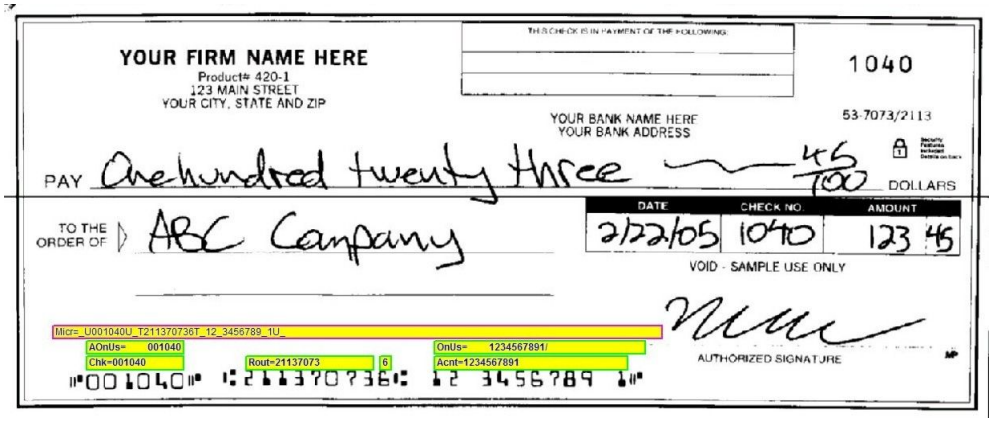


Figure 1.1: A bank utilizing OCR to read and validate checks

Since so many of these documents are now held on a virtual environment, it is also important to develop something to help the machine to comprehend them. That is the objective of Natural Language Processing (NLP), which tries to analyze and understand text in documents or other virtual formats and is commonly used, among other use cases, to create chatbots to help costumers, or find some structure in unstructured data. Figure1.2 represents the different entities identified by an NLP implementation, specifically, a Named Entity Recognition (NER) algorithm.

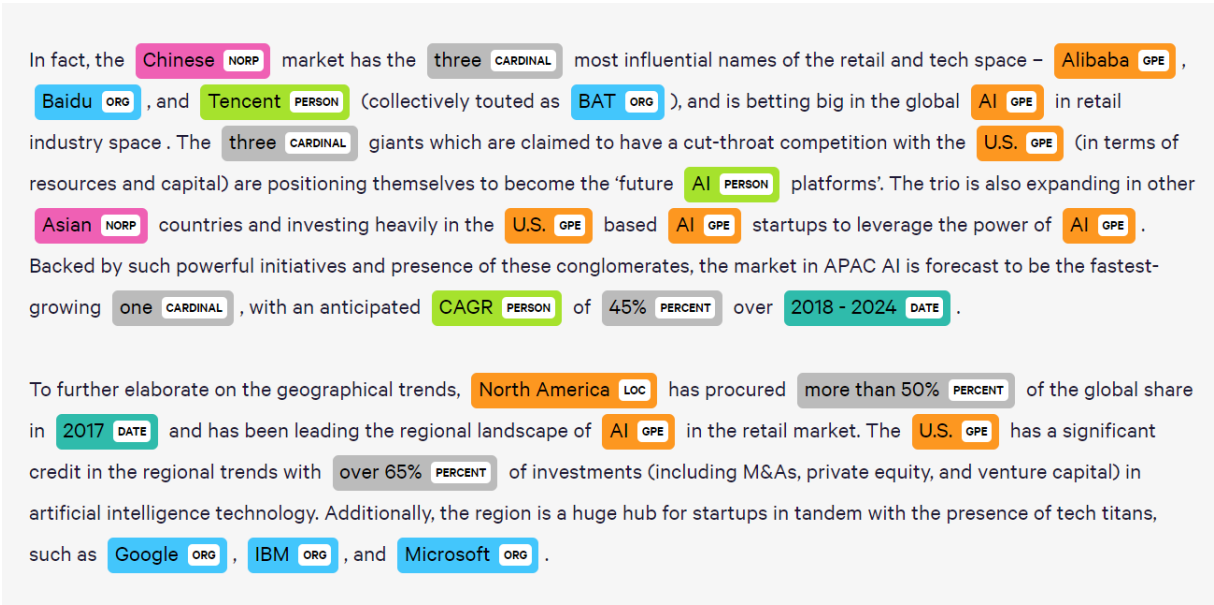


Figure 1.2: A named entity recognition result after being performed on a text.

1.2 Problem

The main goal of this work is to automate the process of erasing confidential data defined by a specific set of rules from multiple PDF files. In order to achieve this objective, it is necessary to:

1. Process the files in order to make them more perceptive to the machine.

2. Extract the text from it with an almost perfect accuracy to guaranty a correct identification about what is written in the document.
3. Automatically recognize which information is confidential given a set of rules,
4. Identify its location in the PDF and erase that specific region.

1.2.1 Optical Character Recognition

To be able to identify the text present in the document, we will resort to the use of OCR [1]. This is a technology used to extract text data from an image file and can be useful to edit, correct, analyze or process physical documents without the need of doing it manually, which could result in errors and waste of time. During early stages, OCR was used exclusively to recognize printed text, and it required each character in each font to be trained separately. Over time, researchers started to include different approaches mainly using machine learning algorithms (Decision Tree (DT), Random Forest (RF), among many others...) and fusing them with preprocessing techniques that led to an advancement in performance and results. This would evolve later to be able to identify nearly all the fonts with a good accuracy, using a variety of different format of files as input for its training. At the moment it can recognize handwritten text, Intelligent Character Recognition (ICR), from a variety of languages already and even identify some non-text elements present in the image.[1]

This system analyses the different light and dark areas of an image, finds patterns on the text, and evaluates the similarity between these patterns and the features contained in a data set[2]. To achieve this recognition, the following processing steps illustrated in Figure 1.3[3] are suggested.

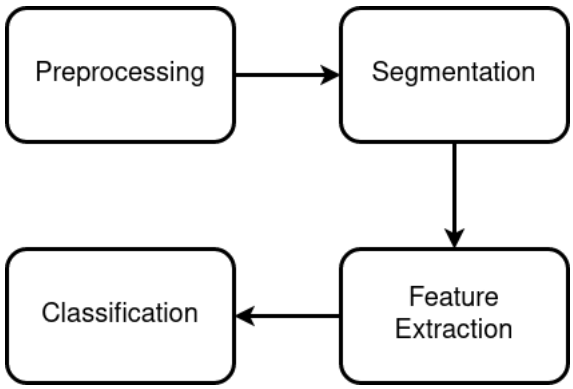


Figure 1.3: Different steps to achieve character recognition

Preprocessing

In order for the text included in the image to be more perceptible to the algorithm, we need to apply some modifications, for example as stated by [4], that increase the contrast of the text to the rest of the image. This process is done during the "preprocessing" module.

The document we are analyzing can be a digitization of a physical paper and it is common for it to include some noise, and therefore we have to try to remove as much of it as we can. The noise in an image is the dissimilarity of pixel colors present on the document, which can cause the image to lose some quality and perceptiveness.

Images are composed by a large set of pixels from multiple color values that range from 0 (black) to 255(white) and therefore certain segments of the document can be less distinguishable. To correct that, after the noise has been removed we can execute the binarization process, which consists in converting the image to a scheme of colors only using black and white. This process works by identifying a threshold value and divide the image into 2 groups (a group for black and a group for white pixels) using that threshold.[5]

After we convert the images into a black and white color scheme, we can start the image filtering operations. This step is used to enhance the image with the intent to better highlight the relevant parts of it. The algorithms for these operations often include applying comparing the values from each pixel with its neighbors and can be used to sharpen or enhance the edges of the different image segments.

It is also pretty common for an image to be rotated or tilted, which can increase the difficulty of the readability of the document. Therefore, we need to correct it before trying to segment and analyze it, and it can be done using the skew correction method. According to [6] using this method, the image is amended by rotating it by its skew angle, whereby it will be calculated by finding a peak in the gradient orientation's histogram of the skewed image.

Segmentation

After the preprocessing is done, as explained in [7], we then try to identify and separate the different components contained in the text. The aforementioned method can refer to three different levels of segmentation: Line, word or character level.

When the system is being segmented by line level, a horizontal scan of the image is carried out and consequently the pixels will be grouped by intensity and a small skew correction is also made to ensure the lines are not invalid. The word segmentation has a similar behavior to the line level, however, the image is scanned vertically, and the skew correction is applied to each individual word instead of the entire line. The character level segmentation works similarly to the two previous levels, but it also applies a ligature removal, which is the process to try to separate each character if they are overlapping.

To apply the segmentation there are multiple different approaches [7], being the most suitable when working with text the following: Pixel Counting, Histogram approach, Smearing approach, Stochastic approach and the Water Flow approach.

For the Pixel Counting approach, the image is scanned from left to the right and top to the bottom in order to binarize all the pixels and either assign them a black pixel value or a white pixel value. Thereafter, utilizing a threshold value, the segments will be separated if the number of white pixels (or black pixels if the image is already binarized and complemented) between them is greater than the threshold. This can lead to some inaccurate results if some characters overlap each other.

The Histogram approach is used for handwritten segmentation and starts by the binarization of the image and, consequentially, obtaining the Y histogram projection. The Y histogram projection represents the number of pixels per y-value, so the values of y where the number of pixels is low should indicate the end of a segment.

Using the Smearing approach, the image black pixels (or white if the image is complemented) will be smeared and if the distance between them is less than a pre-determined threshold, the empty space between them will be filled with more black pixels. The result will be a group of characters contained in a black area defining the different segments.

The Stochastic method works with Hidden Markov Models (HMM) in order to perform nonlinear paths between lines overlapping each other and consequentially, dividing the image in small cells corresponding to the state of the HMM. A search is then made following left to right to find the best segmentation path. If there are touching characters, the highest probability path will cross the point with as few pixels as possible.

As for the Water Flow approach, it utilizes an algorithm to simulate a flow of water flooding the image area from left to right and top to bottom. The water will then fill the pixels that are not black (which will be referenced as unwet) and each group of unwetted pixels will represent a text segment.

Pattern Recognition and Classification

Having the image properly processed and segmented, the next step is to extract the features from each character, so they can be classified and recognized. Through deep learning Neural Network (NN), it is possible to utilize a single model to perform both the feature extraction and the recognition of the characters. To perform the recognition, because the data input will be an image, usually the best course of action is to use Convolutional Neural Network (CNN) and combine it with a Long Short-Term Memory (LSTM) to handle the text, since it is better suited for sequence of data, where its output will be connected to a Connection Temporal Classification (CTC) which will result in a vector ready to be classified.

The classification can be achieved either by comparing the features extracted from the image, or by using probabilistic models to identify the characters.

Software

There is a lot of available software in the market to provide an OCR service either to companies or individuals in professional or personal projects.

Tesseract is on the most used OCR software available and is very simple to use, with good documentation, it is open-source, and it does not cost anything. The downsides of Tesseract are that the image processing is not very good, so to get a good OCR performance it is necessary to preprocess the image before using it, and it can have some recognition problems with non-English characters (e.g. “ç”).

Google Cloud Vision API is another option for OCR with a similar performance to Tesseract in a typed document recognition, although, unlike Tesseract, this one is a paid service. It can also easily recognize multiple characters from different languages and its performance in handwritten recognition is much better. Its biggest drawbacks are not being effective in detecting tables and not supporting documents with size over 10 MB.

OmniPage Nuance is also a very good paid alternative, specially if the need stands for recognizing table, columns or other type of complicated formats. Its biggest problem lies in the low performance when recognizing text in colored backgrounds.

1.2.2 Natural Language Processing

With the advancement of technology, it has become increasingly essential to understand human communication and, therefore, an area of artificial intelligence has been in development, which studies the understanding of natural language, whether written in a computer, handwritten or by voice. To be able to understand this language, the machines are equipped with microphones, cameras or keyboards to receive the information and then convert it to machine language so the computer can read it [8].

Preprocessing

Similarly to the introductory section of 1.2.1, to apply NLP algorithms it is also necessary to preprocess the input data before we can try to work with it. This process can include the following steps:

- Dividing the data into smaller groups, also known as tokenization. This is commonly done by split the words by either the spaces or the punctuation between them.
- Remove all the stop-words from the data. These are words that are commonly used in any sentence and do not carry any relevant information to the meaning of the text(e.g "the", "a") and may vary with the language we are working with.
- We can also apply lemmatization, which is the process of grouping all the words deriving from the same root word(e.g "reader", "reading", "read").
- After the lemmatization is completed, we can associate all the grouped words to their respective root word. This process is called stemming.
- We can also mark the different segments present in the input data with any appropriate tag(e.g "determinants", "nouns", "adjectives").

Natural language processing can be done using either syntax or semantic analysis of the data. The former, refers to the order in which the different words are arranged in a sentence and its techniques include: splitting the text by its word classes and analyzing the different parts, also called parsing and is used mainly for a more complex preprocessing; dividing the text by all of its words, also known as word segmentation; recognizing and analyzing each sentence present in the data individually, also known as sentence breaking; morphological segmentation, which consists in inspecting the morphological formation of the words; stemming, that we also use to preprocess the data.

Semantic techniques work with the true meaning of the words in the sentence and these can be: "word sense disambiguation", which runs an algorithm to try to identify the true meaning of a word based on the context of the sentence; "natural language generation" is a technique to find the meaning of words in order to generate new content through a database containing labeled text; "named entity recognition" 1.2.2 is another semantics technique where the machine tries to assign each word to a predetermined classification.[9]

The understanding of the text can be made with two different approaches: rule-based grammar and via a Machine Learning algorithm. Both of them have specific use cases in which they are better and more suited than the other.

Rule-based grammar

This is a system to identify words based on a specific set of handwritten grammatical rules defined by a human in order to recreate the common structure of a language. This method can be very scalable and flexible, since we can easily adjust or add any function or rules to our system without compromising the main code of our project. Because these rules are human-made, it can be very easy to verify any specific set of input data provided to the machine and any bug can be tracked without much effort.

However, for complex scenarios and text, it might take a bit more of time, becoming difficult to analyze and recognize a specific query with this method and some rules might end up overriding each other. For these reason, it is needed someone with a good set of language skills to program and scale the rules whenever we need a new case scenario of recognition.

In conclusion, the rule-based grammar is better used to identify a specific query in a sentence as it can easily understand the different relationship between the words contained in it.

Machine-Learning algorithm

This methodology utilizes statistical methods and algorithms trained via a data set in order to create its own ruling in order to understand and recognize a text without the need of a specific defined human rules.

This is great because it has an ability to learn by itself without the need of a skilled person in language and communication, and it needs very little code in comparison to 1.2.2. It is very useful to cluster words and classify documents since we have multiple reference points which its statistical hints can be easily learned by the machine.

On its downside, since these algorithms learn from big data sets, it can be very hard for a human to understand the reason the model learned something in a specific way and therefore, the outcome can be unpredictable when there is a modification to the training data, and it can become a difficult task to debug any problem that may arise.

The training data can also become a challenge to overcome since the outcome really depends on the quality of it. If the model has a good training data, the system can be very accurate and fast, however, it may not be easy to find good data, and sometimes we also need to manually fill the data, which is a very resource consuming task.

Word Vectors

An important data when it comes to NLP is the word vectors, also known as word embeddings, which are vectors representing each word and can be used to qualify the similarity degree of a set of text. There are some sets of algorithms developed in order to calculate this vector.

Latent semantic analysis is one of the most accurate algorithms to extract word vectors, and it is computed by converting a co-occurrence matrix into a vector. This matrix contains a list of words on its rows and columns present in a text[10], as well as the number of times

that each word is related to another, e.g. if we have the sentence: “o contrato acaba hoje. Hoje é preciso renovar o contrato”, the corresponding matrix is described in the table1.1.

	o	contrato	acaba	hoje	é	preciso	renovar
o	2	2	1	2	1	1	1
contrato	2	2	1	2	1	1	1
acaba	1	1	1	1	0	0	0
hoje	2	2	1	2	1	1	1
é	1	1	0	1	1	1	1
preciso	1	1	0	1	1	1	1
renovar	1	1	0	1	1	1	1

Table 1.1: Co-occurence matrix for the text: “o contrato acaba hoje. Hoje é preciso renovar o contrato”

Word vectors can also be computed through neural networks and an example of this case is the word2vector algorithm which is a particular NN. It has two possible types of architecture: “continuous bag of words” and “skip-grams”.

The former receives as an input a list of the words that appear previously to a specific word in analysis, and they are used to predict the next one. For the latter, the specific word in analysis is given as input of the network and used to output a prediction of the context it is inserted, whether before or after.

Named Entity Recognition

A branch of NLP that aims to identify and categorize different entities present in the text, is called NER. This can refer to either a standalone word or a group of multiple words that together may be recognized as being the same body, and each one of them will be assigned to a pre-determined category.

In order to identify an entity, some algorithms may search in a data set for a matching in a group of words, while others can be defined by a specific set of rules.[11] They can also be identified using machine learning classification algorithms[12], the better ones being supervised, using successful models like Maximum Entropy (ME), HMM and Conditional Random Field (CRF), which are considered better than rules-based, since they have a better adaptability to different domains, however, these require a large annotated training data set to be effective.[13]

ME models work with the posterior probability of a determined label to be assigned to a word and used to be one of the highest successful approaches to labelling Part-of-speech (POS) and NER. However, recently, CRF has surpassed it since using it, the problem of generative models is now overcame, the *label bias problem* does not occur because CRF has only one exponential model for the joint probability of the label sequence and the transition probability between the labels, now depends on either the past and the future of the observations.

In 2011, studies were conducted with a series of algorithms utilizing CRF and Support-vector machine (SVM) on chemical formula and name tagging, and the results displayed a

higher accuracy on performance than the pre-existing methods so far. Stanford NER, one of the most popular NER framework available, also used CRF as its base classifier.[14]

Software

When it comes to NLP, there are a variety of frameworks available, each with their own pros and cons.

PyTorch is a good library and entirely open-source. It is commonly used to work with NLP and Recurrent Neural Network (RNN) to solve classification, tagging and text generation problems. In terms of performance it is fast and due to the open-source nature it's also fast and supports GPU computation.

StanfordCoreNLP provides an API which makes it available through a web service and a multitude of different programming languages. It offers a variety of services and one of the best in delivering a good NER with a support for a total of 53 languages. On the downside it does not support a lot of customization which can be a problem, specially if there is a language without much support.

SpaCy is another great framework for NLP written in Python and Cython. Like Pytorch and StanfordCoreNLP it is also very fast and supports advanced NLP techniques like NER in a variety of different languages already, however, unlike StanfordCoreNLP, it can be customizable with the user's own data set of labeled text for a more personalized entity recognition.

1.3 Structure of the document

This thesis is divided into six chapters, namely:

- Chapter 1 - Introduction: describes the motivation, problem proposed and the solutions to solve it.
- Chapter 2 - System: presents the system architecture and the implementation of the image manipulation module.
- Chapter 3 - Optical Character Recognition: explains the OCR technologies chosen and how they were used.
- Chapter 4 - Natural Language Processing: describes the NLP library used and the implementation of the respective module.
- Chapter 5 - Results: presents the individual results of each module as well as the entire system.
- Chapter 6 - Conclusions: summary of results and references to future work.

CHAPTER 2

System

In this chapter is going to be presented the description of the architecture of the implemented system, the different technologies and how they interact with each other, the data set used and the image and PDF manipulation technologies.

As discussed in 1.2 our system main objective is to receive multiple contract documents, read and understand text contained in each one of them, identify sensitive information that is not allowed to be public and output another file with that same information hidden. To fulfill this task, as illustrated in Figure 2.1, it was developed an architecture for our system based on three main modules: image and PDF manipulation, OCR module and NLP module. The system will be fed with a variety of PDF files, being them editable, digitized or simple photographs.

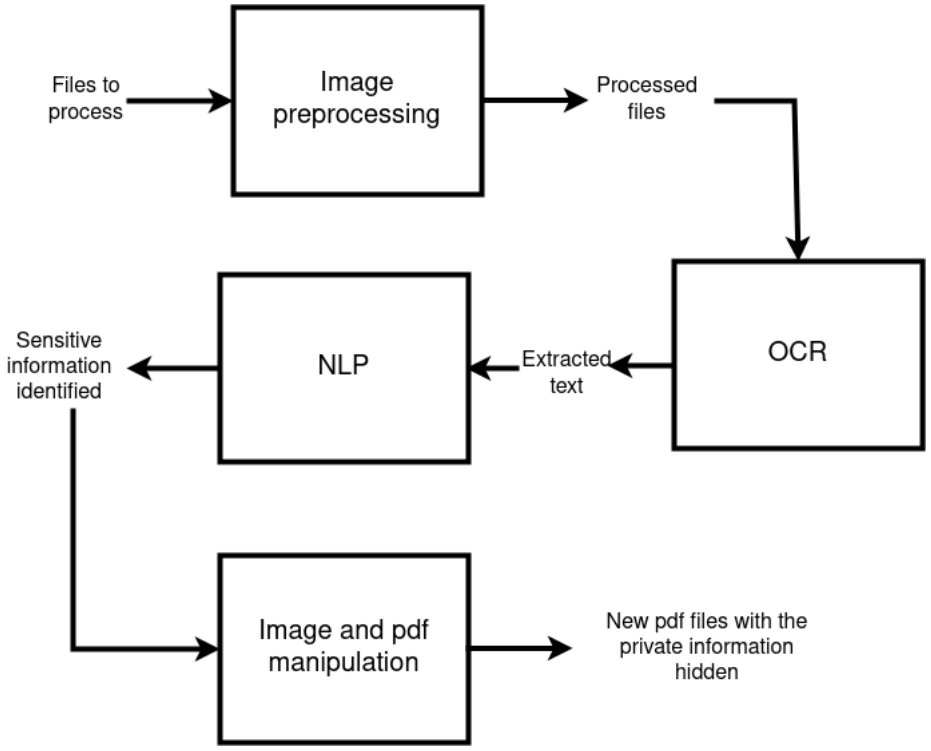


Figure 2.1: Description of the implemented system

The PDF and image manipulation will have responsibilities at the start and at the end of the program. It is the first module, and it is responsible for handling the input files with the modifications deemed necessary in order to improve the other component's performance. It's also used in the end of the program execution to apply the changes computed in the previous modules.

The OCR module will receive the processed file and its responsibility is to read what is contained in the document. The output will be the contract's content and the respective location on the document.

The NLP component has the job to understand the meaning and semantics of the words received from the OCR. It will then identify which information should be private and transfer that information to the image manipulation.

The technologies used in these modules can be seen in figure 2.2. The image and PDF manipulation will resort to the OpenCV[1], PDF2Image[2] and Pillow[3] libraries. The OCR and the NLP are implemented through the Google Cloud Vision and the spaCy library, respectively, but only the former needs a continuous internet connection.

[1] `https://docs.opencv.org/4.x/`
[2] `https://pdf2image.readthedocs.io/en/latest/overview.html`
[3] `https://pillow.readthedocs.io/en/stable/`

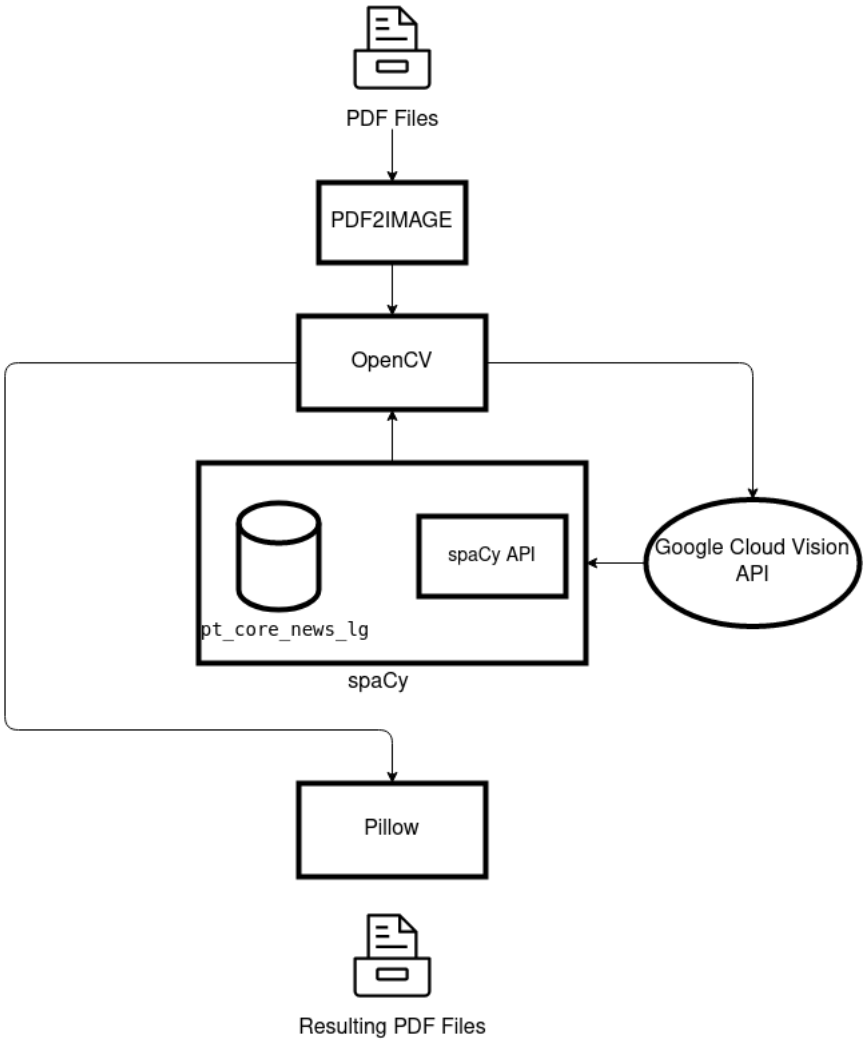


Figure 2.2: Description of the use and relation of the technologies present on system

2.1 Database

The database used to test and evaluate our system was made available by IMPIC1.1 and includes a set of PDF documents. These documents are contracts written in typed letters between entities (either organizations or individual person) with multiple pages and with different tasks to be applied to each of them.

There are files in which the private information consists of personal information of the individual present in the document, such as name, Número de Identificação Fiscal (NIF), phone number, e-mail and the identification document number. In these types of files, it is important to be able to identify who is representing an organization and who is the individual.

In some other files, we have to hide information about the collective that compromises data from employers. This information can be either the permanent certificate code from a company, or the personal information of the contractor representative or the contract manager.

There are also two distinct types of PDF files to analyze. The first one, being an editable PDF, where it is possible to select, look for text or even edit the file. In these types of files,

usually there is less noise, the images are aligned, and no skew correction is needed.

The second type of PDF file is a document that was scanned through a camera and therefore is not editable. These types of documents function like a photograph, and the data can either be selected or edited by a user. It is also more common in these types of documents to include some kind of blur or tilt in the image, which can difficult the OCR task.

To perform the entity recognition on the given text, the spaCy framework utilizes the `"pt_core_news_lg"` pipeline. This pipeline includes a model already trained with a large Portuguese written text data set composed by media articles such as news and social media, and its performance is optimized to work with the CPU.

The data set used for training the `"pt_core_news_lg"` pipeline was data retrieved from either WikiNer[15] and UD Portuguese Bosque [16]. The former is a set of words contained in Wikipedia articles which form a semi-supervised training data. The latter is a converted version of Bosque, a subset of the Floresta Sintá(c)tica tree bank.

2.2 PDF Processing

The PDF processing unit will be used to convert the files that are fed to the system into image format. This will allow the image processing component to carry out the preprocess operations, apply the OCR and modify the respective image. It will also be responsible for reconverting the final images into PDF pages and create the new reconstructed file.

To convert the PDF into JPEG we used the "pdf2image" module, a wrapper of the tools "pdftoppm" and "pdftocairo" which converts the original file to a list of PIL objects, each one representing a different page.

To perform the opposite operation, i.e. convert the multiple edited images into the final PDF, it was used the Pillow fork from the Python Imaging Library (PIL) library, made to improve the handling of images in python.

When the entire operation of the program is complete and the private information on an image is hidden, it will be saved in the root directory as a temporary JPEG and the Pillow library will read the image, create a PIL object and save it on a list of pages. Once all pages are completed, it will append them and output the final PDF file and the temporary image will be deleted.

2.3 Image Processing

The image processing unit will have two different responsibilities, being the first to preprocess the files and the last one to modify the image in order to hide the information considered private. Was used OpenCV for these operations, a library with a set of tools to work with computer vision and image processing. This library allows us to read, save or apply modifications to an image.

When the PDF file conversion to image is completed, each page will be handled separately. To be able to use OpenCV on these images, we have to create a temporary JPEG file in our computer to save the PIL object, since this library can't read objects of that type.

After the JPEG is read by the machine, we begin by binarizing (described in Section1.2.1) the image and for that we start to convert the image to grayscale and invert the bits value which will cause the background of the image to be black and the foreground(i.e. the text) will become white. It is a common practice to transform the foreground to light colors and the background to darker ones when working with Computer Vision.[17]

As explained previously 1.2.1, the binarization works by applying a threshold value and changing the pixels depending on whether they are greater or lower than the chosen threshold. In this case we applied the Otsu's Binarization, which instead of choosing an arbitrary value as our threshold, it determines automatically an optimal value from the histogram of our image.To apply this type of thresholding, the cv2.threshold function was used with the `cv2.THRESH_BINARY` and `cv.THRESH_OTSU` flags on top of the grayscale inverted image.[4] In the figures 2.3 and 2.4 we can observe the document before and after the binarization was applied respectively.

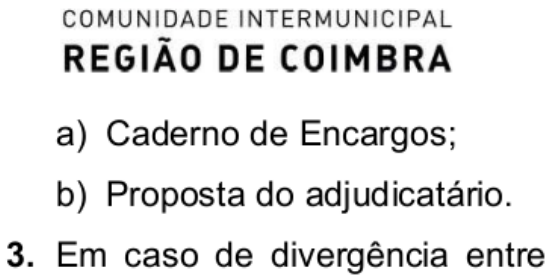


Figure 2.3: Document before binarization

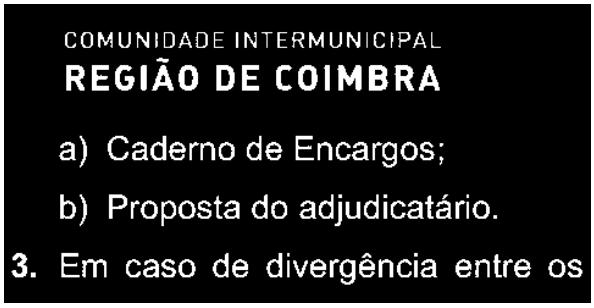


Figure 2.4: Document after the entire binarization process

It is possible that sometimes the target image is a little skewed, most likely because the digitized paper was slightly tilted, which would impair the recognition of the text. To correct this phenomenon, we apply a skew correction algorithm over the binarized image.

We start by getting the coordinates of all non-black pixels (i.e. pixels with a value greater than 0) present in the image, representing the coordinates of our foreground components, in this case, the text. Inserting these set of coordinates into the OpenCV function minAreaRect and will return a rectangle containing those specified points and its respective angle. With the angle of text area found, we need to rotate the image in order for it to be at 90º, so the target rotation angle will be (90 - current_angle) if the angle is less than 90º and (current_angle - 90) otherwise. However, after running some tests, it was concluded that the resulting angle was slightly over 90 degrees and that the proper rotation would only be half the original computed angle. In order to correct that error, the angle in which the image would be rotated was divided by 2. This approach was explained in this tutorial [17], however, it is stated that the rectangle angle provided by OpenCV was counted clockwise, but our tests showed the opposite(i.e. the returned angle was calculated counterclockwise).

Afterwards, the center of the image would be computed through its height and width and knowing the rotation angle and the center point we could create the rotation matrix and

[4] `https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html`

apply the rotation making use of OpenCV functions: getRotationMatrix2D and warpAffine respectively.

Unfortunately, in some images the OpenCV would not recognize the rectangle in the right angle which would cause the image to be even more skewed than its original format, so the rotation is not being applied in the final version of the system.

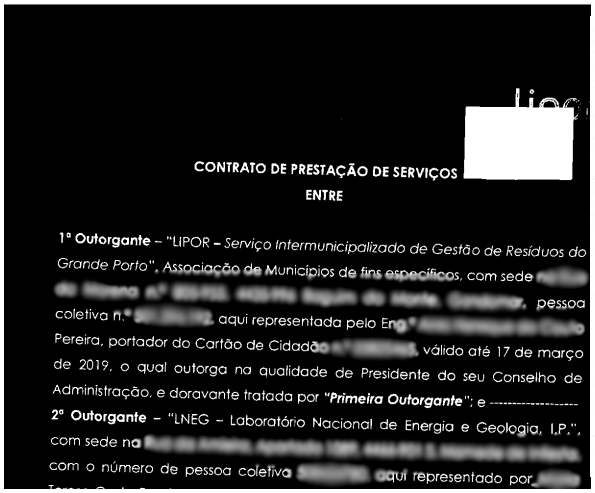


Figure 2.5: Document before being rotated

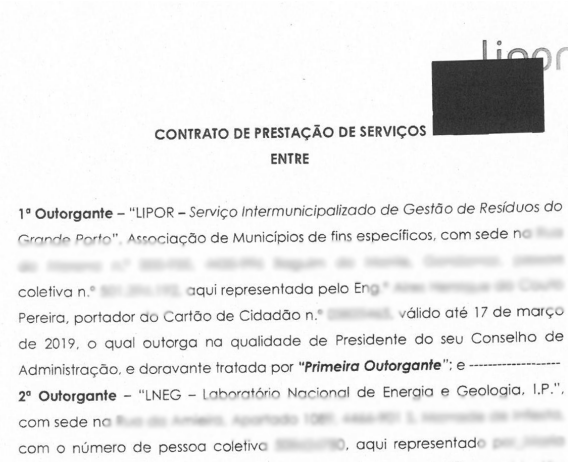


Figure 2.6: Document after skew correction

The thinning process was also tried, which consists in removing some pixels from the words found in binary images, to reduce its pixels with the intent of having a more defined letters and making it easier to detect its edges and improve the character recognition. It is often used to support the skeletonization process, which tries to reduce a figure to its basic skeleton [18].

The algorithm chosen was the Zhang-Suen Thinning Algorithm, the most popular thinning algorithm. It works by first verifying which pixels from the bottom right corner of the image can be eliminated and posteriorly checks the top left one.

The algorithm analyzes each foreground pixel that contains eight neighbors, whose neighbors describe a transition from white to black pixel only once. With these pixels filtered, they will be tested to verify if either one of the top, right and bottom neighbors are white pixels and if it confirms, then a similar verification is made for the right, bottom and left neighbors (bottom right). The pixels who pass these tests will be converted into background. Once the entire image is tested, another test similar to the previous one is going to be applied, with the difference in the neighbor pixels analyzed, since this time, the set of neighbors tested for white pixels will be either one of the top, right and left or the top, left and bottom. Having this second verification finished, the entire process is going to be repeated until there's not one pixel left to be chosen[5].

OpenCV already contains a function implementing thinning with this algorithm, in particular through the use of cv2.ximgproc.thinning(). It receives a binary image as input, aswell as the thinning type, which we can choose the Zhang-Suen through the enumerate

[5] `https://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm`

`cv.ximgproc.THINNING_ZHANGSUEN`. This function will return the input image skeletonized, and the result can be seen in Figure 2.8.

However, after multiple tests, we concluded that the OCR accuracy slightly dropped with the introduction of the skeletonization, and it ended up not being part of the final system.

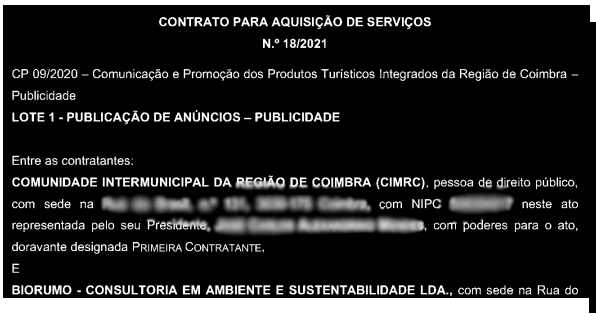

CONTRATO PARA AQUISIÇÃO DE SERVIÇOS
N.º 18/2021
CP 09/2020 – Comunicação e Promoção dos Produtos Turísticos Integrados da Região de Coimbra – Publicidade
LOTE 1 - PUBLICAÇÃO DE ANÚNCIOS – PUBLICIDADE
Entre as contratantes:
COMUNIDADE INTERMUNICIPAL DA REGIÃO DE COIMBRA (CIMRC), pessoa de direito público, com sede na [redacted], com NIPC [redacted] neste ato representada pelo seu Presidente, [redacted], com poderes para o ato, doravante designada PRIMEIRA CONTRATANTE,
E
BIORUMO - CONSULTORIA EM AMBIENTE E SUSTENTABILIDADE LDA., com sede na Rua do

Figure 2.7: Document before applying the thinning algorithm.

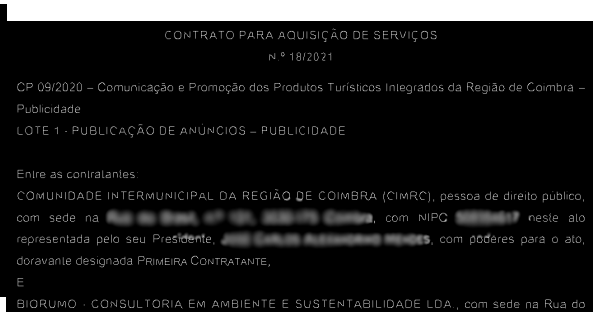

CONTRATO PARA AQUISIÇÃO DE SERVIÇOS
N.º 18/2021
CP 09/2020 – Comunicação e Promoção dos Produtos Turísticos Integrados da Região de Coimbra – Publicidade
LOTE 1 - PUBLICAÇÃO DE ANÚNCIOS – PUBLICIDADE
Entre as contratantes:
COMUNIDADE INTERMUNICIPAL DA REGIÃO DE COIMBRA (CIMRC), pessoa de direito público, com sede na [redacted], com NIPC [redacted] neste ato representada pelo seu Presidente, [redacted], com poderes para o ato, doravante designada PRIMEIRA CONTRATANTE,
E
BIORUMO - CONSULTORIA EM AMBIENTE E SUSTENTABILIDADE LDA., com sede na Rua do

Figure 2.8: Document skeletonized.

The last preprocess operation to apply is removing the noise. This is needed, since the documents can be digitized or photographed, which can usually lead to great amounts of noise on the image. Operations like blurring or other types of processing, which we apply to our image, can also increase the noise.

The noise removal is done using the Non-Local Means Denoising algorithm, which works by computing the average color of a great portion of the pixels with similar values and replacing their own pixel with the computed one. To find the similar pixels, a search is made in a large window around the pixel in question and is compared their resemblance[19].

OpenCV contains a function that will help us to implement this method with some computer optimizations. As for the search window size, we chose a size of 21 pixels since it is the recommended value for images with small or medium-sized standard deviation of the noise().

Figure 2.9: Document before removing the noise

Figure 2.10: Document after the noise is removed

Regarding the task of hiding information, it could have been done with two different approaches. The first one being to draw a rectangle over every single word deemed sensitive(e.g. to hide the name “Jonh Doe”, two rectangles would be drawn, one covering “Jonh” and the other covering “Doe”) which would be faster however, less secure since the user could be able

to identify how many words and its length contained that piece of information(in the previous example, we could assume that the person with the hidden information had two names with few letters each). The second method would consist in verifying a piece of information as a whole and draw a single rectangle covering all of that information. If this approach was applied to the previous example, only a single rectangle would be hiding the name "Jonh Doe" and the user reading the contract could not identify how many words or letters that person's name had, since it could either have one long name or two short names.

In order to draw these black rectangles covering the information from the reader, the OpenCV library was also used. Finding the size and the coordinates where the polygon would be drawn could be accomplished with the Google Cloud Vision OCR tool, since at the time it is performing the recognition it also returns the coordinates of each vertex of a rectangle representing the segment of the text it identifies. With the coordinates of each vertex as the input to the OpenCV, it is possible to draw the rectangle using the function fillPoly. This method was preferred over the function OpenCV.rectangle since the latter only received the top left and the bottom right vertices as input which would always drew rectangles at 0^o and when the image is skewed, the rectangles must have the same tilt angle as the expression we are trying to cover.

However, there was one problem with using a single rectangle for an entire set of words which was the case where some of those words could end up on the line below and in this case, two rectangles would be needed, one for each of the text segment in each line. To verify if two consecutive words are on the same line, an algorithm was developed by setting a predetermined arbitrary value that represents the space between each line and calculating for each word, their respective highest and lowest vertices in the y-axis. After computing this information, the distance from the lowest vertex of one of the words to the highest vertex of the other (representing the horizontal distance between the two words) would be compared to the arbitrary value corresponding to the size of a line and if the distance between vertices were smaller, the words would be considered on the same line.

To be able to hide an entire set of words with one single polygon we have to analyze the number of words each private expression has and, as discussed previously, verify if these are on the same line. For every segment of text in a set of information of the same line, there will be computed which are the leftmost and rightmost words and their leftmost and rightmost vertices respectively, to feed it as input to fillPoly to draw the adequate polygon. The result can be observed in figure2.11.

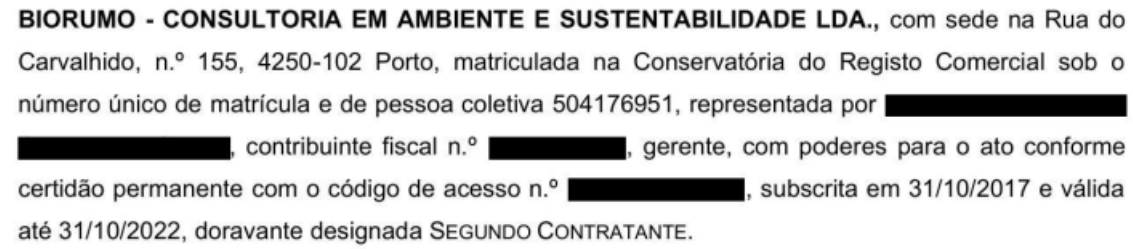

BIORUMO - CONSULTORIA EM AMBIENTE E SUSTENTABILIDADE LDA., com sede na Rua do Carvalhido, n.º 155, 4250-102 Porto, matriculada na Conservatória do Registo Comercial sob o número único de matrícula e de pessoa coletiva 504176951, representada por [redacted], contribuinte fiscal n.º [redacted], gerente, com poderes para o ato conforme certidão permanente com o código de acesso n.º [redacted], subscrita em 31/10/2017 e válida até 31/10/2022, doravante designada SEGUNDO CONTRATANTE.

Figure 2.11: Resulting file when hiding a name, NIF and access code.

CHAPTER 3

Optical Character Recognition

The Optical Character Recognition chapter describes the two APIs used, the performance of both of them, and the implementation of this technology on the system.

The OCR technology, as mentioned previously in Section 1.2.1, is used to recognize text contained in an image, and it usually requires some sort of preprocessing in order to obtain a better accuracy and performance. Since the objective of this dissertation is to look for private information in a document and hide that information, it is of extreme importance for the OCR implementation to be as close as 100% accuracy as possible.

OCR in our system, will receive an image previously preprocessed and will output the text discovered on the document into the NLP component. To carry out this recognition we tried two different state-of-the-art approaches of OCR technology, being them Tesseract and Google Cloud Vision OCR and compared them.

3.1 Pytesseract

The first framework we tried to use was Pytesseract, a python wrapper of the open-source OCR engine, Tesseract. It also includes a support for Pillow objects2.2 which we will use to handle our images and PDF files.

Tesseract is an engine available for different frameworks and programming languages with the API being provided to fully utilize it in our project, however, a direct executable is also available for those who don't need to customize and integrate it in their work. There is not an official Graphical User Interface (GUI) yet, however, there are multiple external ones available for users that are not very comfortable using the terminal [20].

To utilize this framework, firstly we need to set up the environment in order to recognize Portuguese characters. We need to download the tessdata from the GitHub repository and set our environment variable "TESSDATA_PREFIX" to the parent directory of the downloaded repository.

The recognition is made by calling the API function image_to_data giving the image object directly from OpenCV (in type of numpy.ndarray) as the input. As an additional argument, we also indicate the language as portuguese and the output of the function as dictionary, through an enumerate made available by pytesseract itself.

This API will respond with a dictionary about information of all the boxes found with five different levels of recognition as keys: page_num is the first box level and shows the current page of the box found; block_num is the second level and shows the number of blocks of text contained on the image; par_num is the third level and represents the paragraph of the current box; line_num is the fourth level and indicates the line in which the box is at; word_num is the last level and indicates the number of the word. Each of these boxes also contain information through the keys: 'left', 'top', 'width', 'height', 'text', which respectively represent the x coordinate of the left lower vertex, the y coordinate of the left lower vertex, the width of the box, the height of the box and the text contained in it.

To access the entire text recognized on the file in order to output it to the NLP, a loop was implemented to iterate over the image_data["text"] values and append them to a string object. This will form a string containing the entire text, which we can preprocess before trying to understand the semantics.

A downside that comes with this framework is the lack of preprocessing. Tesseract preprocessing capabilities are minimum, so the image being sent to the API must be fully preprocessed or the OCR won't be effective. Another problem we faced was that despite being configured with Portuguese language, sometimes it still had some troubles to recognize Portuguese characters like"ç" and when that happens, the entire word would become incomprehensible because it ended up affecting the other words altogether. The boxes were also a limitation because Tesseract identifies only horizontal boxes and consequentially, it only returns the lowest left vertex and the width and height of the box which will prevent an accurate box drawing for tilted images.

On the upside, it is a free open-source software and does not need a connection to the internet once the environment is set up. It also provides great information like number of paragraph or line number of a word which is useful in our system since we need to verify if the words are on the same line to draw the rectangle and would save some calculations and computational power.

3.2 Google Cloud Vision OCR

Since pytesseract3.1 had some flaws, another approach was tried, the Google Cloud Vision OCR, an API developed by Google with functionalities to implement OCR in our systems. This is a section of the Google Cloud Vision API, a resource aimed to help developers with computer vision tasks, such as identifying faces and classifying images, OCR.

Google Cloud Vision is available on multiple languages or even with direct use through a terminal, and this system is utilizing the python API. This API allows for the recognition of either local files stores in the user computer or located on the web only by passing the image

URL. It offers two different types of services of annotation resources for OCR and all of these services responses are returned in a JSON format, easy to use and understand by the machine.

TEXT_DETECTION is the first resource and receives any given image and identifies the text contained in it. It responds with a JSON containing the detected boxes of text identified in the image with their respective string and bounding limits to form the box rectangle.

DOCUMENT_TEXT_DETECTION is the other resource available and is similar to the previous resource but was build exclusively to identify text in a document. Therefore, it is more complete and optimized for that purpose and its response includes the page number of the segment of the text identified, the number of the respective block, the string contained in it and the if there is a line break associated. It can either identify typed PDF documents or handwritten files. This is the service that best fit the problem proposed for this system and the one in use.[1]

To be able to use Google Cloud Vision, we have to create a project in the Google Cloud Platform in order to get an API access key. Since this is not a free to use service, a payment method and plan must also be given. With the project all setup, the user will receive the API key or service account key in the format of a JSON file and define the environment variable on the operative system "GOOGLE_APPLICATION_CREDENTIALS" with the directory path to that file. The API is all setup, and we can start implementing the algorithm.

The detection using the Google framework is executed through a service named ImageAnnotatorClient and therefore, the first thing we have to do is create an object of that class. This object will contain the method "text_detection" which we will use to perform the OCR.

On the contrary to pytesseract, Google Cloud Vision does not support an OpenCV object as input, instead we have to convert the image to a Google Vision Image object. However, to make this conversion, the API does not support OpenCV object as well, so we have to locally save a temporary image on the computer, and read it as bytes through the help of the IO package.

Once we have a bytes object representing our image, it is possible to convert it to a Vision Image object with the function "vision.Image" and setting its argument "content" as our bytes variable. The response from this request will be an Image, which we can pass as an argument for the API call "text_detection" of our Image Annotator Client object. Optionally, we can pass the language as argument, however, this framework already supports automatic language detection and most of the time with better results than specifying a language. [2].

The API will respond with the JSON previously mentioned3.2 and automatically convert it to an AnnotateImageResponse object. An AnnotateImageResponse contains a list of EntityAnnotations, accessed through the attribute "textAnnotations", which corresponds to each segment of text found in the image and their respective properties. These properties include the description, which represents the string contained in that segment, the score, representing the confidence in that specific recognition and the boundingPoly, an object containing the coordinates of the vertices of the box area delimiting the segment of the text.

[1] `https://cloud.google.com/vision/docs/ocr#vision_text_detection-python`
[2] `https://cloud.google.com/vision/docs/ocr`

The boundingPoly object contains a list of the four vertices, each one with the x and the y coordinates.

Having received this information, we can input it to the NLP component and similarly to the approach using pytesseract, an iteration over the response.text_annotations will be performed, and for each text_annotation its respective description is going to be appended to a string variable in order to build the entire text of the page.

However, the Google Cloud Vision comes with some less desirable aspects. The first one being it is a paid tool and requires constant communication to the internet in order to fully utilize the API. The second comes with a bit more complexity in terms of handling the inputs to the API in comparison to pytesseract, since we need to create an additional temporary image solely to being able to read the image in bytes. The information returned about the line break is also a problem since during our tests it was not completely reliable, which led to the necessity to implement in our system a custom method to verify if words are on the same line. Since the objective of this system is to hide sensitive information that should be private, another problem relies on the fact that to use the API we need to send that data to the Google Cloud Vision servers where the OCR is being performed, which can create a conflict with our the main goal.

On the other hand, the OCR returned from this API had a better recognition than the pytesseract, specially with Portuguese characters. The performance is also a bit faster which could be very helpful if we are analyzing large volumes of documents, and it also contains a better set of preprocess operations. The information regarding the bounding box of the words found is also better than pytesseract since it returns the four vertices of the box instead of one vertex and the height and width, which makes the hiding process for tilted texts a lot easier.

Overall the Google Cloud Vision OCR is a better tool for our system and is the approach in use since high accuracy in identifying Portuguese text is crucial for this system to have a satisfactory result, and it has better support to identify bounding boxes on titles pages which may happen often.

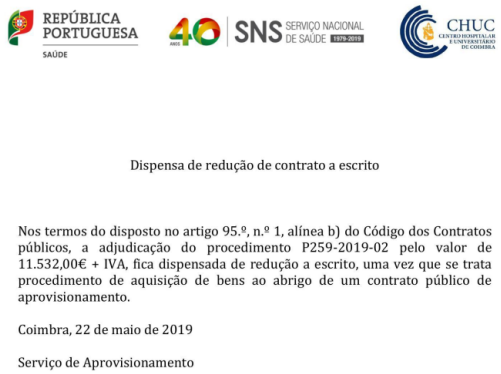


Figure 3.1: Document to be analyzed

```
república portuguesa chuc 40 sns serviço nacional saúde 19792019
centro hospitalar universitário coimbra ands saúde dispensa reduçao
contrato escrito nos termos disposto no artigo 95 n 1 alínea b )
codigo contratos publicos adjudicaçao procedimento p259201902 pelo
valor 1153200 € + iva fica dispensada reduçao escrito uma vez que
se trata procedimento aquisiçao bens abrigo um contrato publico
aprovisionamento coimbra 22 maio 2019 serviço aprovisionamento
( serviço aprovisionamento medicamentos reagentes ) ( praceta prof
mota pinto 3000075 coimbra ) 1/1 ( tel : 239 400 511 ) ( fax :
230705352 ) ( email : comprasmr @ : chuc.min-saude.pt )
www.chuc.min-saude.pt c
```

Figure 3.2: Text extracted from the document

CHAPTER 4

Natural Language Processing

This chapter describes the use of the Natural Language Processing component, covering the preprocessing of the text, the library used, the rules-based algorithms developed and its implementation.

As described in Section 1.2.2, NLP is a technology to allow the semantic and syntax comprehension of a text by the machine. This is an important topic to understand documents, contracts and more.

The objective of this dissertation is to understand which information is supposed to be private and hide it, therefore, NLP is the main unit of our system, being responsible for identifying that same information. This unit will receive the image recognized text as input from the OCR component and return a list containing the words to hide. It will be responsible not only for the identification of the text, but also for its preprocessing.

4.1 Text preprocessing

Given that we are working with contract documents and some can contain an extended amount of text, to obtain the best result with the least computational effort it is advised to apply some preprocessing modifications to the text. These are mainly removing or modifying specific words or letters contained in the text.

The first preprocessing we applied was to convert all the text words into lower case. This process is applied since both the computer and the NLP model will analyze capital letters and lower case letters differently, and we want to normalize the text as much as we can, so the model can understand it.

This method can improve the result since usually, the only letters that are supposed to be upper case in the model's analysis are personal and organization names, however, due to the sentence structure and the type of documents content, the first letter of a sentence and the person's job must also be capital letter which might cause some error.

However, by transforming all the letters into lower case, some names can be mistaken by objects or places (e.g. in Portuguese we can have names like “Maria das Dores” or “João Jardim”, and it converts to “dores” and “jardim” which means “pain” and “garden” respectively). This might confuse the NLP system and cause the names to not be identified as such and consequently, will not be marked as sensitive.

To decide which approach to take, a series of tests were conducted, and it was concluded that transforming the text to lower case was beneficial in comparison to the capital letters.

The second processing step we applied to the text, was to remove punctuation signals. Those are important to remove from the text, because they have no meaning to the semantics of the document.

This change improved the language processing accuracy since a lot of punctuation signals are attached to words which could cause a disturbance in the language processing. This is very common to occur in a middle of a sentence, when there is a comma or a pair of parentheses or even at the end of a sentence with a full stop.

However, there are some tricky cases that we have to work around and situations where punctuation gives us context about the information in question. These cases that we identified are the following: postal codes, that include a hyphen in the middle of two numbers; e-mails, since they contain at least a commercial at and a dot, and it may even contain hyphens or underscores; dates, because they may contain either slashed or hyphens separating the different digits; monetary amounts, which often include the currency symbol next to its value, and it may contain dot or commas to separate the different levels of a coin. There were not made any exceptions regarding the last two points since for the use case in question (contracts sensitive information), dates and monetary amounts would not need to be hidden, but for the other examples we had to create some rules.

In order to avoid losing the recognition of the postal codes, the hyphens are not being excluded from the text at all, since they might also be important to the e-mails and don’t compromise much of the rest of the content. The same logic was applied to the commercial at, i.e, it does not compromise the rest of the message and is import for the e-mail recognition, so it was kept. The dots or full stops however, they might be important for e-mails but they compromise a lot of the text, since they are usually attached to the final word of a sentence. To overcome this, a list of possible e-mail domains was created, which includes: .com (general domain), .net (network), .pt (Portuguese domain), .org (organizations domain), .eu (Europe domain), .es (Spain domain) and .co.uk (English domain). A verification is then made and only the expressions that do not contain any of those e-mail domains, will have their dots removed.

The next change implemented was to substitute diacritics for their respective root letter. Diacritics are letters which contain signals and symbols, often on top of them, e.g, “à” or “ã”. These types of letters are very common in the Portuguese text but they don’t carry a lot of meaning to the semantics of a phrase so we can replace them for their root form, which for the previous examples would be: “a”, and it will not only help the NLP model recognition but it will also be useful to identify stop-words since we would have only one form of each letter.

The diacritics being removed are: “à”, “á”, “ã”, “â”, “é”, “ê”, “í”, “ó”, “ô”, “ú”, and “ç”.

The last modifications applied to clean and improve the text to the NLP was the removal of stop-words, i.e. words that often appear in large quantities in a paragraph, however it does not carry any significant impact to the semantics of the text.

In order to identify which words would appear more in a document, we scanned a three-page contract while counting the words and displayed them in a bar graph. Figure 4.1 represents the plot containing the seven most common words that appeared in the document. However, from these seven words, we could not remove: “com” because it would affect the emails we could also not remove “do” or “de” because some Portuguese names include these words, e.g “Maria do Céu”, “João de Carvalho”, and by removing those the NLP would handle “Maria” and “Céu” as different people, the “n” could not be removed as well since it is an indicator that a number will be followed on the text, e.g, “telefone nº: 913 XXX XXX” and “contracto” (means “contract” in Portuguese) was another word that we could not remove because it is an important word for the document content.

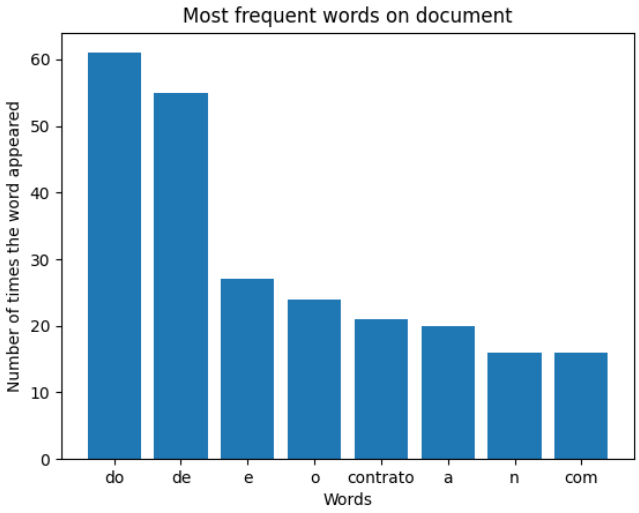


Figure 4.1: The seven words most common in the document.

After this analysis there were three words left to be identified as stop-words: “e”, “o” and “a” and for that reason we verified the next most 7 common words on the document. Figure4.2 is a plot representing those new words and we could already observe more two stop-words, and these are: “no” and “na”. We could not remove “coimbra”, “clausula”, “lote” or “presente” for the same reason we could not remove “contracto” and “da” for the same reason of the words “do” and “de”.

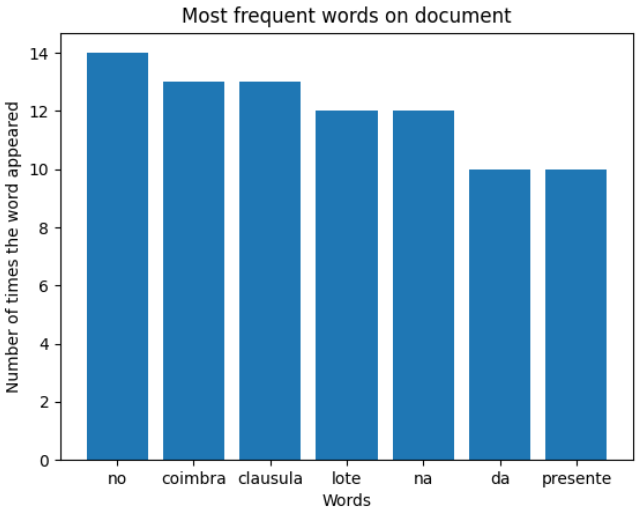


Figure 4.2: The next seven words most common in the document after the first seven.

Using this method, our stop-words list already contained seven elements but since there are a lot more we can clear, a new verification was made for the next seven most common words present in the document. Figure4.3 shows the next seven words, e.g, the words between the 14th and 21st position. The last stop-words we found were then: “pelo”, “em”, “os” and “para”. We could not remove “intermunicipal” or “contratante” since both of those are important words for the meaning of the text and “dos” for the same reason of the previous word “da”.

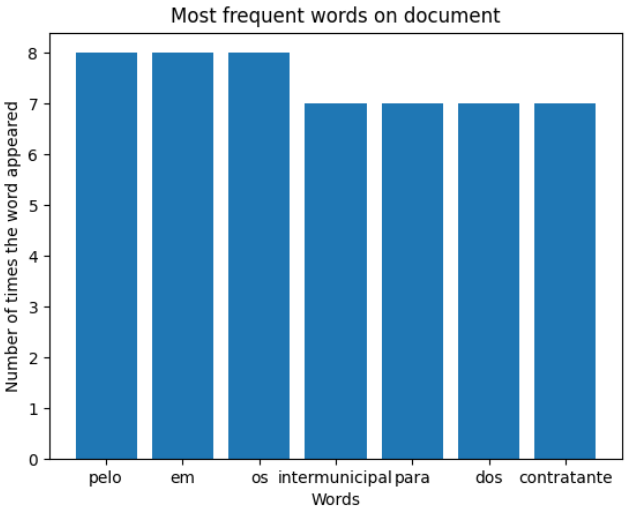


Figure 4.3: The next seven words most common in the document after the first fourteen.

In the end our stop-words list contained the words: “e”, “o”, “a”, “no”, “na”, “pelo”, “em”, “os” and “para” and posteriorly some more words were added, mostly derivations from the previous, specifically: “as”, “ao” and “que”.

4.2 spaCy

To accomplish the proposed objectives, the spaCy framework was used. This is a state-of-the-art free to use open-source python library in the NLP industry, with the goal to better understand big quantities of text, offering text processing, NER, POS, classification, and more types of support and linguistic annotations for a variety of different languages including Portuguese, the main one for the use case of this dissertation. These annotations make it easier to understand same words for different context, e.g, the word "claro", might either be an interjection or an adjective and the way it is understood can influence the entire phrase.

The spaCy NLP offers a set of trained pipelines, each for different use cases, in order to better understand the target text, however, it also accepts the use of custom ones. These pipelines can change in its language, size, speed or memory used and contain multiple statistical models trained using labeled data which often include: binary weight for different types of annotations with the intent of being able to identify them in the text and words dictionary as well as their attributes, word variations or vectors representing each word. To optimize the system performance and capacity, all of the strings are converted to hash values.

This workflow of this library starts by applying a tokenization1.2.2, which is done by splitting the text by its spaces or even by its punctuation. When a word is split, each of the separate segments will be analyzed again to verify if they must be targeted of a further splitting, and this process will be repeated until each word are down to their most basic form. This process is language dependent, so it may vary from different contexts. It will output a Doc object to the processing pipeline.

As for the different annotations, the identification of different POS and dependencies can be accomplished by feeding the model a great quantity of data to allow it to generalize and predict certain specific characteristics through the use of the previously mentioned pipelines. These annotations can be accessed through attributes of the "Token" object. Each Token is composed by: TEXT, the string without any modifications; LEMMA, that identifies the root word; POS, represents the part-of-speech of that specific word; TAG, is a more specific POS tag assigned to the word(e.g, the POS attribute of a word might be "PROPN", representing a pronoun and its TAG can be "NNP" which represents proper noun singular); DEP refers to the syntactic dependency of the token (e.g. "AUX", for an auxiliary verb); SHAPE, reproducing the format of the segment being analyzed (e.g. "Sr.João" would be represented as "Xx.Xxxx"); ALPHA, which indicates if the token contains only letters; STOP, indicates if that specific word is considered a stop-word.

The NER is identified through a set of statistical models and are also very language and trained data dependent. This can become a problem for languages with less support, e.g. the large Portuguese pipeline only includes NER for four different categories: LOC, PER, ORG and MISC, while the large English pipeline contains the categories: CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, ORDINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME and WORK_OF_ART.

To identify how similar two different words are, a comparison is made between each of

their word vectors, also known as word embedding. The spaCy tool utilizes the word2vector algorithm explained previously1.2.2. The small pipelines do not contain the word vectors, however, it is still possible to compare the similarity between words with less performance and accuracy.

The spaCy architecture is then divided into three main data structures which will be responsible for the entire workflow of the framework. These three are: Vocab, Language and Doc and each one has a specific responsibility in different times of the NLP execution.

Vocab objects have the main responsibility to centralize the data needed for the language processing. This data refers to strings, word vectors and the lexical attributes and prevents the data from repeat itself which optimizes the capacity of the system.

The Language is the structure responsible for receiving the text input of the NLP system. After receiving the text, this class will process it and create the respective Doc object via the tokenizer component.

The Doc object contains the text annotations extracted from the NLP of the text. It is created by the tokenizer and further altered by other components of the pipeline filling the different annotations. It contains the Span, slice of Docs containing the words belonging to a specific text annotation and Token, corresponding to each individual word, objects.

To be able to use the spaCy in our project, first we have to install the library in our machine and download the most suitable pipeline for the problem, which for this dissertation was the `"pt_core_news_lg"`, the large Portuguese pipeline, in order to deliver the best accuracy possible. Having finished the setup on the local machine, we can initialize the Language(in our case, Portuguese) object by calling the `spacy.load()` function with the desired pipeline (`"pt_core_news_lg"`) as the argument.

To carry out the text processing, we pass the text into the Language object which will return the Doc containing information relative to each Token and Span. To access each Token identified by spaCy we just need to iterate over the Doc elements.

With the Token object, a lot of information can be accessed about a specific word. It is possible to identify if said token is an e-mail (using `token.like_email`), a stop-word(`token.is_stop`), its base form(`token.lemma_`), if the word is the first or the last from a sentence(`token.is_sent_start` and `is_sent_end`), its descendants(`token.is_ancestor`), its dependencies(`token.dep_`) and more useful annotations. Since, as mentioned previously, the entire set of strings are converted to hash values, in order to access string attributes from tokens, those must have an underscore as suffix, e.g. `token.lemma` return an integer corresponding to its hash value, but `token.lemma_` returns the original string representing the word base form. The NER processing can also be found by calling the `Doc.ents` attribute which for each Token will return one of the four enumerate values: LOC, PER, ORG or MISC.
[1]

[1] `https://spacy.io/`

4.3 Rule-based implementation

Despite having a NER library analyzing our text, it can't recognize more specific Portuguese information, like postal codes, contract codes or distinguish between telephone, cellphone and identification numbers. It also could not distinguish in the contract, who is a singular person and who is a person working for an organization. Since that is an important specification for the outcome of this system, a few sets of rules-based verification were implemented.

The first thing to do, was to correct the numbers structure and join the different number segments contained in the document into a single number. This is a problem since the telephone, mobile phone numbers and some amounts of currency that appear in the contracts might be separated into smaller chunks of numbers, e.g. some phone numbers have the shape "XXX XXX XXX" or "+XXX XXX XXX XXX" with the prefix signaling the country. This resulted in the OCR detecting these numbers as separated words.

To fix this issue, an iteration will be made over all the text's words and add them to a string representing the new text. However, if said word is a digit, then it is saved and the iteration continues and if the next word is also a digit, it will be attached to the saved number and the new saved number becomes the combination of those two. This process will be repeated until the next word on the iteration is a non-digit segment, which in that case, the saved number gets appended to the string containing the new text and the loop continues until a new number is found or the entire document has been searched.

The first rule implemented was to verify if a set of numbers represented a postal code. This was done by developing an auxiliary function that would receive a token and verify if its shape is in the form "XXXX-XXX". If that is the case, the function would output that the token is in fact, a postal code.

Another implemented rule was to identify if a given set of digits represents a telephone or mobile number. To accomplish this, a hard-coded list was defined containing every possible prefix that a Portuguese contact can have either from a telephone or from a mobile number. This list contained the following prefixes: 91, 92, 93, 96, 234, 253, 276, 239, 266, 289, 21, 22 and 259. Having defined this list, an auxiliary function was created, which receives an integer or a list of integer as input and outputs a boolean confirming if said number is a contact.

The first thing done by the function is to convert the input into a string, considering it can be either an integer or a list of integer segments, to be easier to divide, manipulate and compare. The next thing the function does when receiving the input is verifying its length and if the number length is lesser than nine, then it is automatically categorized as a non-phone number and it will return False.

If the number has exactly nine digits, then a comparison is made to verify if the first few characters are equal to one of the prefixes defined in the previously mentioned list. The amount of characters of the input being compared will be equal to the size of the prefix list value being verified(e.g. if we are verifying if the characters starts by "91", then only the first two characters will be compared, but in case the verification is with the prefix "234", the first three characters will be analyzed). If the number starts with one of the prefixes, then it is

considered a contact and the function returns True, otherwise it returns False.

Because of the text preprocessing involving digits, the countries prefix, or even another error in writing the document, if the input has more than nine digits, then a phone number can still be contained in it. In this case, first, a process similar to the case where it has exactly nine digits will be applied, but for the first nine and if no contact could be recognized, then this same function will be recursively called but eliminating the first digit from the input integer, and all the previous steps will be repeated for this new input, e.g. if we are verifying if 391341XXXX (X represents a digit) is a cellphone, then a comparison will be made and concluded that "391" is not one of the prefixes and in that case, this same function is called passing 91341XXXX as input which will return true.

The last implemented rule was to identify if a given set of digits would be an identification number, also known as NIF or a permanent certificate number. These were implemented using the same algorithm, which can be described as finding specific keywords in their previous neighbors.

Similarly to the previous rule, the first thing to prepare when implementing this rule is to create two separate lists. These lists will represent the keywords associated to each of the information we are trying to find, in this case, a list with words to find the identification number and another one with words to find the permanent certificate. Associated with the NIF, the words chosen are: "nif", "contribuinte", "fiscal", "identificacao" and only one keyword was chosen to identify the permanent certificate: "permanente".

Having the lists been created, an auxiliary function was implemented, which received the entire document text, the index in that document of the number we are trying to identify and the corresponding keywords list as input. The function then, utilizing the index received, proceeds to analyze the six words previous to the number in verification and if any of those words are contained in the keywords list, then the number would be confirmed as being either an identification number or a permanent certificate, based on the associate words list used.

4.4 Implementation

As we have seen in the previous sections, the implementation of the NLP component is divided in multiple steps and auxiliary functions, which can be a bit disperse and confusing. This section's objective is to describe how the different algorithms come together in order to implement the system required.

The NLP starts by initiating the spaCy language as explained in section 4.2. This will initialize the Language object and save it as a variable named "nlp".

This component receives the raw text identified by the OCR in the format of a list containing all the words inside the document as well as their bounding box information, and the first task,as previously mentioned4.1, is to process it. To do it, an iteration is made over the entire list and process each word with the explained methods, which included substitute the diacritics for their original word, convert the entire text to lower case and removing signals and stop-words.

This could be accomplished by setting three different predetermined lists: a list with the email domains, a stop-words list and a list of tuples with each one containing the diacritics and their respective root form or the signals and an empty string. The normalization process would then use the replace function to substitute all the words contained in the first position of a tuple for their respective second position, which resulted in removing the punctuation signals and the converting the diacritics to their original form. However, there was a verification to prevent words that contained any of the e-mail domains present in the list, from being normalized which resulted in preserving the dot (".") in the electronic mail. Then, if the normalized word were not contained in the list of stop-words (this will remove all the stop-words from the document), the word and its box coordinates information would be added to a list of the processed words while also appending it to a string that will contain the entire normalized text. The string containing the entire text is going to be used as input for the spaCy library to the variable NLP defined before to apply the language processing algorithms and the list will be further used to find the coordinates of the sensitive words that need to be hidden.

A list to save words identified as sensitive is then created. Meanwhile, spaCy will return a Doc containing all the tokens found and using those tokens, an iteration is made to try to recognize any private information and apply the process of joining the different number segments as explained in section 4.3. For each complete number found, a set of functions will be called in order to try to identify them as a NIF, permanent certificate or phone number, as explained in the previous section4.3 and if any number is identified as one of the three mentioned categories, it would be added to the list of private words. Simultaneously, two comparisons to verify either if a token is an e-mail through the token's attribute `"token.like_mail"`, or verify if it is a postal code through the respective function are being made and if the word is confirmed to be one of them, then it is also appended to the list of words that should be hidden.

The next step is to iterate over the entire entity list found by the NLP algorithm and verify their label. If the label is equal to "PER", then the respective word is considered to be a name of a person, and therefore it is going to be appended to the list of sensitive information.

All the words or expressions appended to the sensitive information list, are appended in the form also of a list, where each element corresponds to a word of that expression, e.g. if we find the phone number "91384XXXX" and the name "Jonh Doe", then they will be appended to the list in the form of ["91384XXXX"] and ["Jonh", "Doe"] respectively, which would result in a list with the following format: sensitive_list = [["91384XXXX"], ["Jonh", "Doe"]]. This format was chosen in order to be able to identify which words belong to the same name or expression, and consequentially, as explained in 2.3, the machine will be able to hide the entire expression with a single rectangle.

With the sensitive words already identified, an iteration will be made through the entire list containing the processed words as well as their coordinate's information. If any of those words are equal to an element of the sensitive information list, then using the bounding box vertices from that same words, the image manipulation component will draw a rectangle to

hide it on the final document. The figure 4.4 illustrates the implementation described.

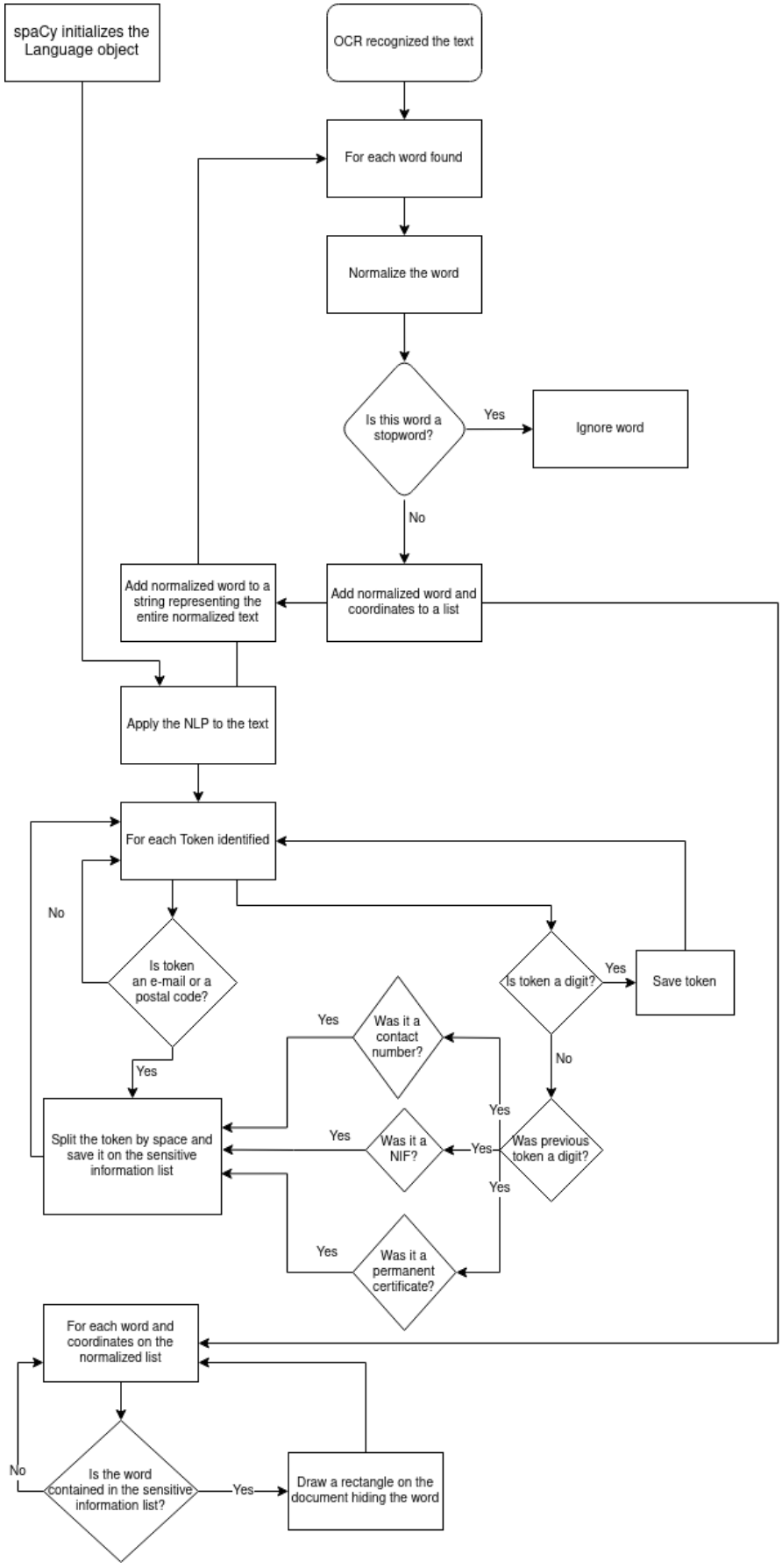


Figure 4.4: A diagram representing the NLP component algorithm, from the text processing to the anonymization of the word

CHAPTER 5

Results

This chapter presents the experimental results obtained in the individual components and on the system as a whole.

5.1 Image manipulation

The image manipulation is a core component of this system and therefore it's important to analyze its performance. The evaluation was divided into the preprocessing and the rectangle drawing algorithms. These tests were made by applying the modification and comparing it visually with original image.

5.1.1 Preprocessing

As stated in section 2.3 a set of image processing algorithms were applied to the input documents to facilitate the optical character recognition of the text, however, not all of those algorithms were successful. There were two successful changes applied and those were: binarization and image denoising while on the other hand, there were another two process changes that did not succeed, either one for different reasons: thinning and skew correction and both ended up removed from the system final version.

The thinning method was excluded because despite the algorithm worked and the thinning process was successfully applied to the image, this change would lower the accuracy of the OCR in some cases. This was mainly because when applying the thinning to each character, a few of them lost some of their shape. The main recognition problem deriving from this method was that most of the "t" letters would be recognized as "l" for being so thin. We can observe this problem on the Figure 5.1 which is a comparison between two files using the meld, a software to help in comparing two different text files, being the left side the text extracted from a thinned image and the right contains the extracted text from an image without the thinning process applied. [1]

[1] `https://meldmerge.org/`

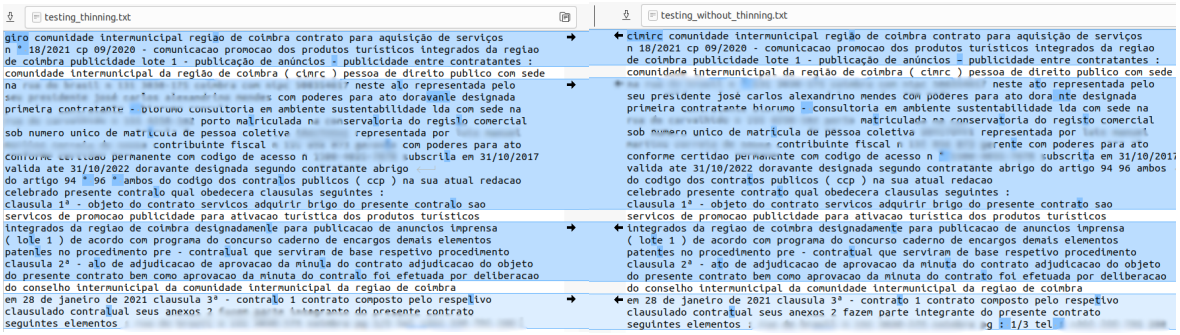
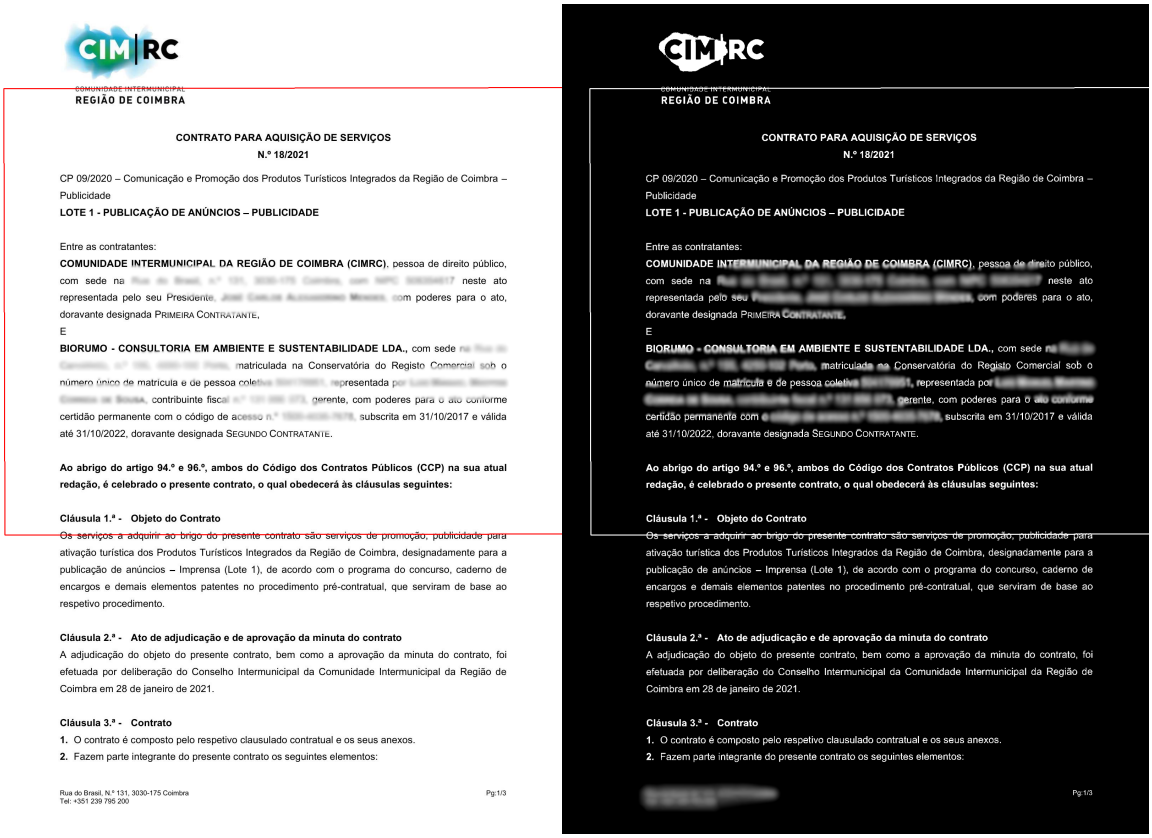


Figure 5.1: On the left, the text extracted from an image with the thinning process applied and on the right the same image text extracted with the thinning disabled

The skew correction process, on the other hand, failed in its implementation. To compute the angle in which the image is tilted, the OpenCV.minAreaRect function was used, which would return a rectangle containing the entire text and its respective angle. However, in the end this angle was not reliable because sometimes, for the same rectangles, it woAbraçould give different results. An experiment was conducted using a perfectly balanced document (editable PDF), and it would assume the first and the third page at a zero-degree angle and the second at ninety degree. This would cause the first and third pages to remain still while the second would rotate in that amount, which resulted in a page rotated in the horizontal direction. Figures 5.2 shows the rectangle identification and respective rotation on the first page and Figures 5.3 shows the same images corresponding to the second page. As we can observe, despite the found rectangle being the same, on the first one the rectangle has a different rotation associated.



CIM RC
COMUNIDADE INTERMUNICIPAL
REGIÃO DE COIMBRA

CONTRATO PARA AQUISIÇÃO DE SERVIÇOS
N.º 18/2021

CP 09/2020 – Comunicação e Promoção dos Produtos Turísticos Integrados da Região de Coimbra – Publicidade
LOTE 1 - PUBLICAÇÃO DE ANÚNCIOS – PUBLICIDADE

Entre as contratantes:
COMUNIDADE INTERMUNICIPAL DA REGIÃO DE COIMBRA (CIMRC), pessoa de direito público, com sede na Rua do Brasil, n.º 131, 3030-175 Coimbra, com NIPC 506394817 neste ato representada pelo seu Presidente, JOSÉ CARLOS ALEXANDRINO MENDES, com poderes para o ato, doravante designada PRIMEIRA CONTRATANTE,
E
BIORUMO - CONSULTORIA EM AMBIENTE E SUSTENTABILIDADE LDA., com sede na Rua do Carvalhido, n.º 150, 4050-252 Porto, matriculada na Conservatória do Registo Comercial sob o número único de matrícula e de pessoa coletiva 504170601, representada por Luís Manuel Monteiro Gomes de Sousa, contribuinte fiscal n.º 191 955 073, gerente, com poderes para o ato conforme certidão permanente com o código de acesso n.º 1555-4030-7678, subscrita em 31/10/2017 e válida até 31/10/2022, doravante designada SEGUNDO CONTRATANTE.

Ao abrigo do artigo 94.º e 96.º, ambos do Código dos Contratos Públicos (CCP) na sua atual redação, é celebrado o presente contrato, o qual obedecerá às cláusulas seguintes:

Cláusula 1.ª - Objeto do Contrato
Os serviços a adquirir ao brigo do presente contrato são serviços de promoção, publicidade para ativação turística dos Produtos Turísticos Integrados da Região de Coimbra, designadamente para a publicação de anúncios – Imprensa (Lote 1), de acordo com o programa do concurso, caderno de encargos e demais elementos patentes no procedimento pré-contratual, que serviram de base ao respetivo procedimento.

Cláusula 2.ª - Ato de adjudicação e de aprovação da minuta do contrato
A adjudicação do objeto do presente contrato, bem como a aprovação da minuta do contrato, foi efetuada por deliberação do Conselho Intermunicipal da Comunidade Intermunicipal da Região de Coimbra em 28 de janeiro de 2021.

Cláusula 3.ª - Contrato
1. O contrato é composto pelo respetivo clausulado contratual e os seus anexos.
2. Fazem parte integrante do presente contrato os seguintes elementos:

Rua do Brasil, N.º 131, 3030-175 Coimbra
Tel: +351 239 795 200
Pg:1/3

CIM RC
COMUNIDADE INTERMUNICIPAL
REGIÃO DE COIMBRA

CONTRATO PARA AQUISIÇÃO DE SERVIÇOS
N.º 18/2021

CP 09/2020 – Comunicação e Promoção dos Produtos Turísticos Integrados da Região de Coimbra – Publicidade
LOTE 1 - PUBLICAÇÃO DE ANÚNCIOS – PUBLICIDADE

Entre as contratantes:
COMUNIDADE INTERMUNICIPAL DA REGIÃO DE COIMBRA (CIMRC), pessoa de direito público, com sede na Rua do Brasil, n.º 131, 3030-175 Coimbra, com NIPC 506394817 neste ato representada pelo seu Presidente, JOSÉ CARLOS ALEXANDRINO MENDES, com poderes para o ato, doravante designada PRIMEIRA CONTRATANTE,
E
BIORUMO - CONSULTORIA EM AMBIENTE E SUSTENTABILIDADE LDA., com sede na Rua do Carvalhido, n.º 150, 4050-252 Porto, matriculada na Conservatória do Registo Comercial sob o número único de matrícula e de pessoa coletiva 504170601, representada por Luís Manuel Monteiro Gomes de Sousa, contribuinte fiscal n.º 191 955 073, gerente, com poderes para o ato conforme certidão permanente com o código de acesso n.º 1555-4030-7678, subscrita em 31/10/2017 e válida até 31/10/2022, doravante designada SEGUNDO CONTRATANTE.

Ao abrigo do artigo 94.º e 96.º, ambos do Código dos Contratos Públicos (CCP) na sua atual redação, é celebrado o presente contrato, o qual obedecerá às cláusulas seguintes:

Cláusula 1.ª - Objeto do Contrato
Os serviços a adquirir ao brigo do presente contrato são serviços de promoção, publicidade para ativação turística dos Produtos Turísticos Integrados da Região de Coimbra, designadamente para a publicação de anúncios – Imprensa (Lote 1), de acordo com o programa do concurso, caderno de encargos e demais elementos patentes no procedimento pré-contratual, que serviram de base ao respetivo procedimento.

Cláusula 2.ª - Ato de adjudicação e de aprovação da minuta do contrato
A adjudicação do objeto do presente contrato, bem como a aprovação da minuta do contrato, foi efetuada por deliberação do Conselho Intermunicipal da Comunidade Intermunicipal da Região de Coimbra em 28 de janeiro de 2021.

Cláusula 3.ª - Contrato
1. O contrato é composto pelo respetivo clausulado contratual e os seus anexos.
2. Fazem parte integrante do presente contrato os seguintes elementos:

Pg:1/3

Figure 5.2: The original image on the left with the respective identified rectangle and the corresponding rotation applied on the right

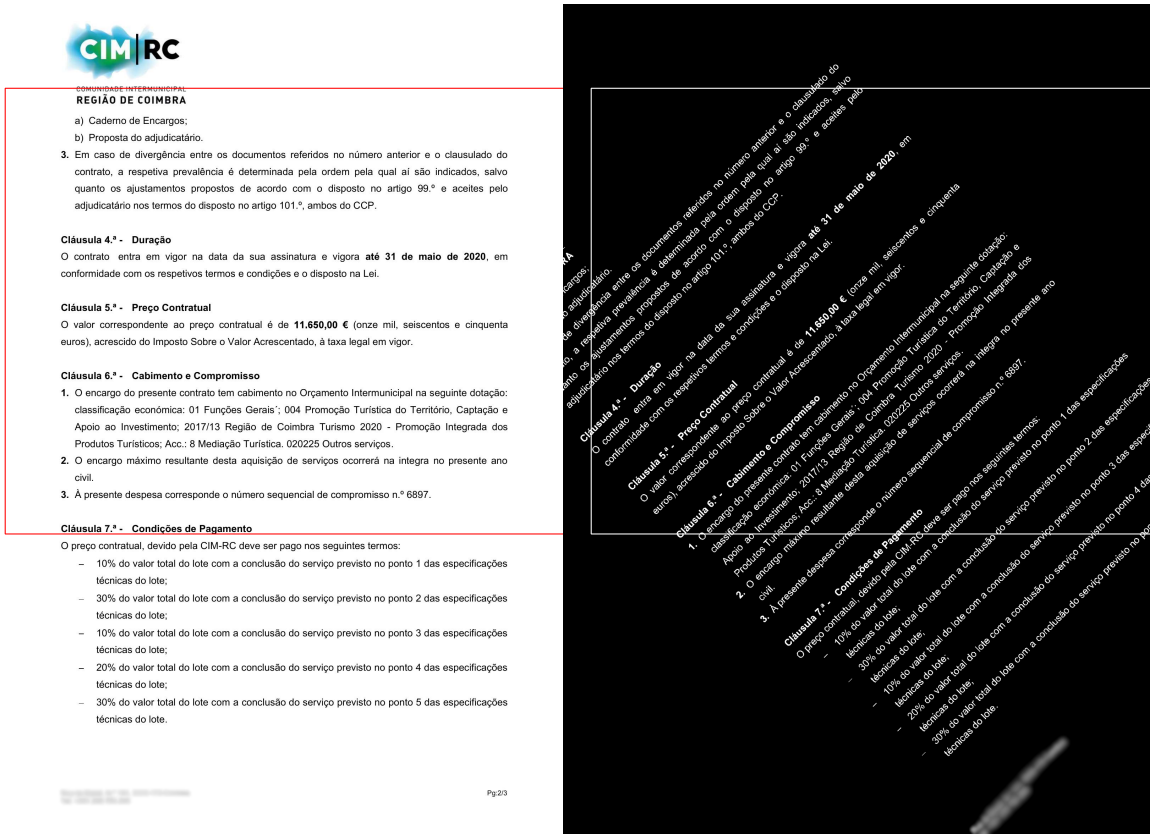


Figure 5.3: The original image on the left with the respective identified rectangle and the corresponding rotation applied on the right

5.1.2 Word erasing

As discussed, the method adopted to hide the private words is by drawing a black rectangle over them and this was accomplished successfully via the OpenCV function fillPoly. The rectangle can adapt to the multiple tilted words, sizes and lines, always covering exactly the information marked as private.

Since the system converts the initial PDF to an image, adds the rectangles and converts it back to a PDF, this hidden information becomes permanently inaccessible. Figure 5.4 exemplifies a tilted image where two rectangles were created with a small angle to cover the entire person name, distributed between different lines.

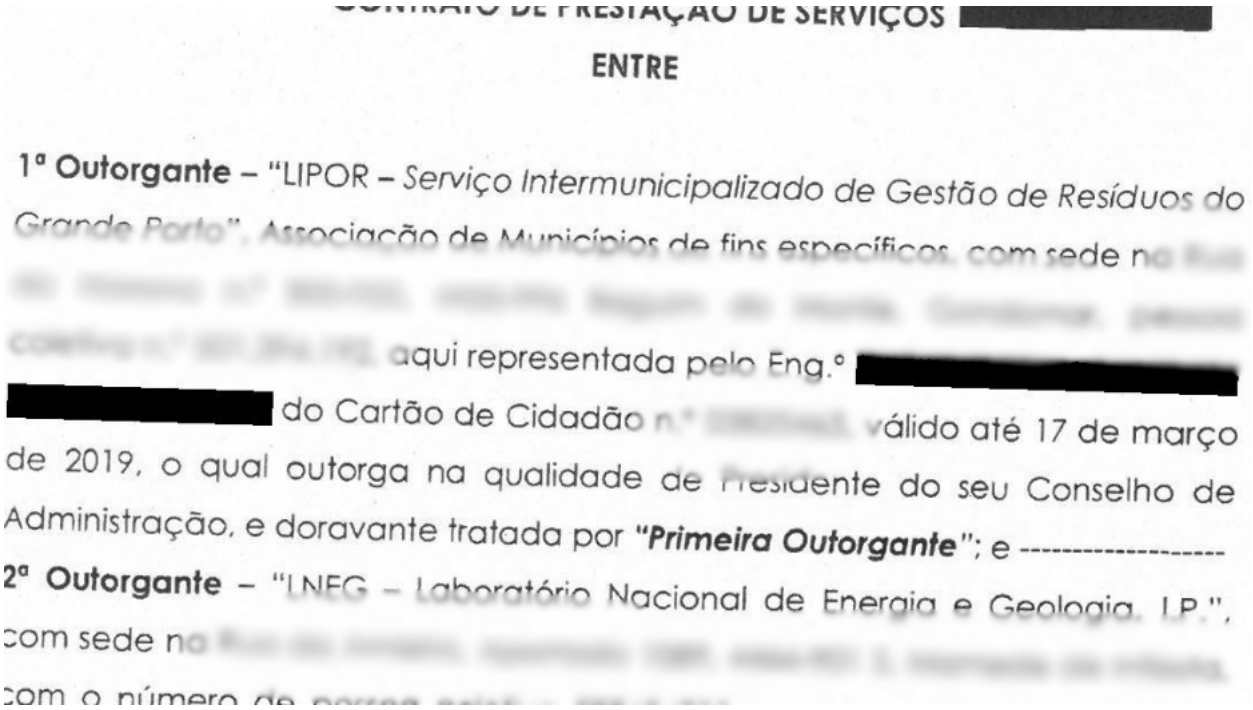


Figure 5.4: A person's name being hidden in a tilted document

5.2 OCR

To evaluate the OCR performance, the Meld software was used to compare two different text files, one contained the exact words contained in the document and the other containing the text recognized by the OCR. This process was repeated to two different files, the first one containing only sixty words and the second containing two hundred and forty. It is important to note that the files containing the exact document text do not include headliners or footnotes but the files recognized by the OCR technology do.

5.2.1 Pytesseract

As explained in a previous section3.1, the pytesseract framework had worse performance in our system than its Google counterpart. This can be observed in both tests made to the library, where it is possible to see a lack of performance in identifying Portuguese diacritics and respective words, as well as symbols, e.g. "º".

On the first test conducted, as observed in figure 5.5, not counting the headliners, the footnotes, diacritics or spaces, there were a total of five words incorrectly recognized in a total of 60, which results in an accuracy of approximately 91.67%.

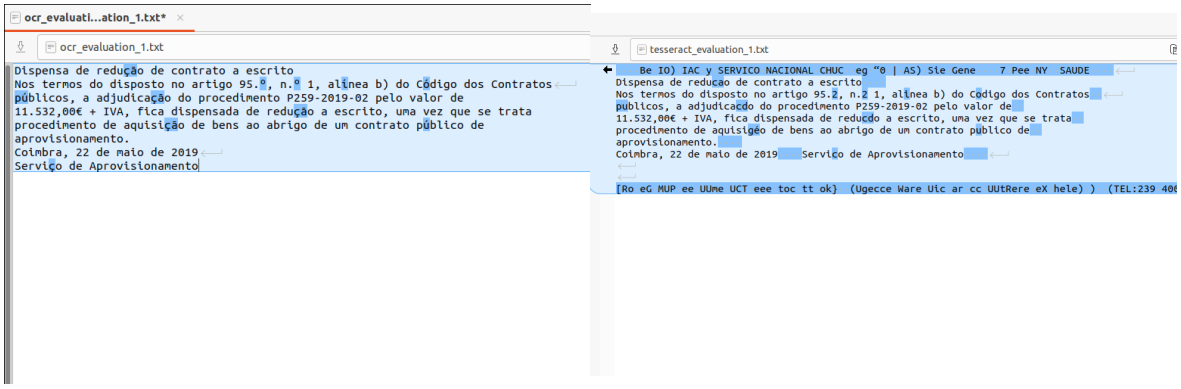


Figure 5.5: On the left, the original document text and on the right the text extracted by pytesseract, both of them with the differences highlighted in blue

The second evaluation can be observed in figure 5.6, and once again not counting the text components mentioned previously, it had a total of 23 errors in the recognition, from a total of 240, resulting in an accuracy of 90.42%. In this test, we can observe another flaw in the recognition different from the pattern of diacritics and symbols, which can be seen in second line, where the OCR had problems recognizing numbers in the format of "dd/dddd".

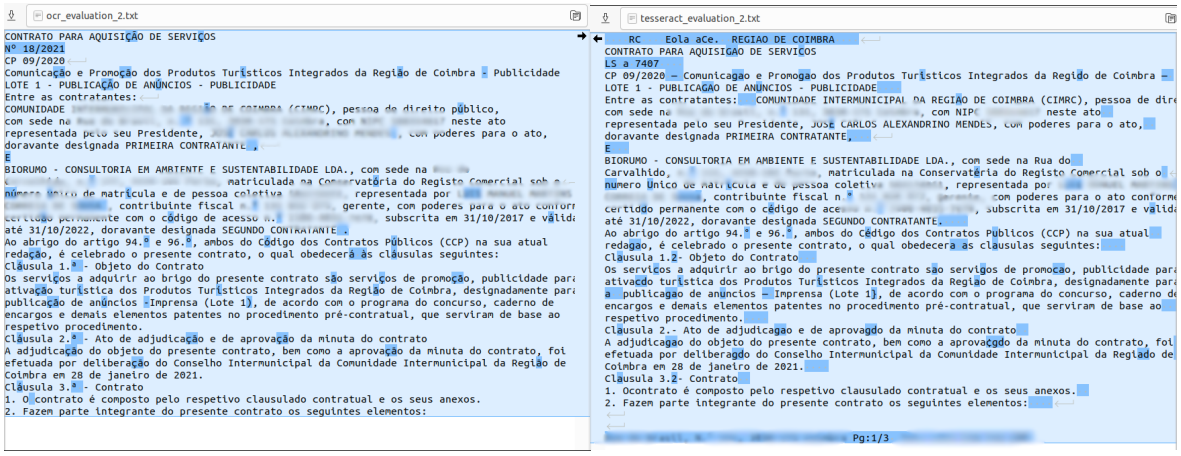


Figure 5.6: On the left, the original document text and on the right the text extracted by pytesseract, both of them with the differences highlighted in blue

5.2.2 Google Cloud Vision

The Google Cloud Vision was our API of choice to perform the OCR in the system since its accuracy had been better overall in comparison to Pytesseract, as explained in section 3.2. This API had no specific flaws and could recognize an entire pages with minimal errors either in words or punctuation. In Figure 5.7 presents the first test to the Google Cloud Vision and by comparing the files we can conclude that both are identical, only differing in a couple of white spaces caused by the printing algorithm implemented. Then, the result is 100% accuracy for a document with a length of sixty words.

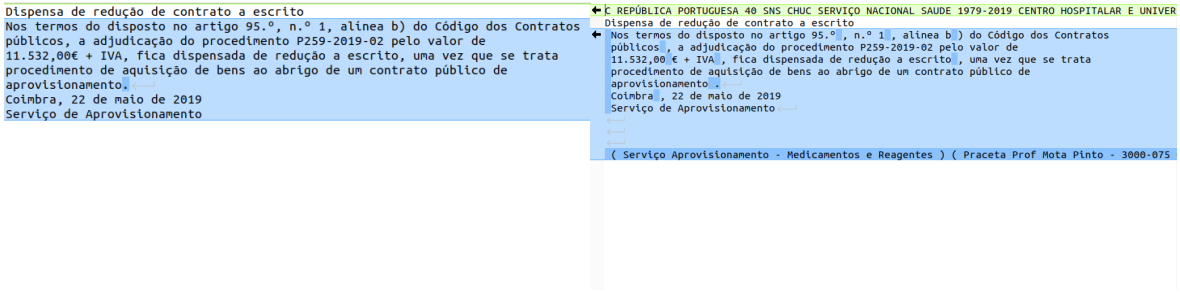


Figure 5.7: On the left, the original document text and on the right the text extracted by Google Cloud Vision OCR, both of them with the differences highlighted in blue

On the second test , the result was not as great as the first, but still very satisfactory. As we can verify with Figure 5.8, without counting the headline, footnotes, and spaces there were only two mistakes made by Google Cloud Vision. These were mistaking a degree symbol("º") for a full stop(".") and not recognizing the word "doravante", instead, it identified two different words "dora" and "nte". The program was re-run a couple of times, but the same mistakes kept happening, and it was not possible to identify the reason for this behavior, particularly in the second case. The accuracy was then of ninety-nine point seventeen percent, having two errors in two hundred and forty words possible.

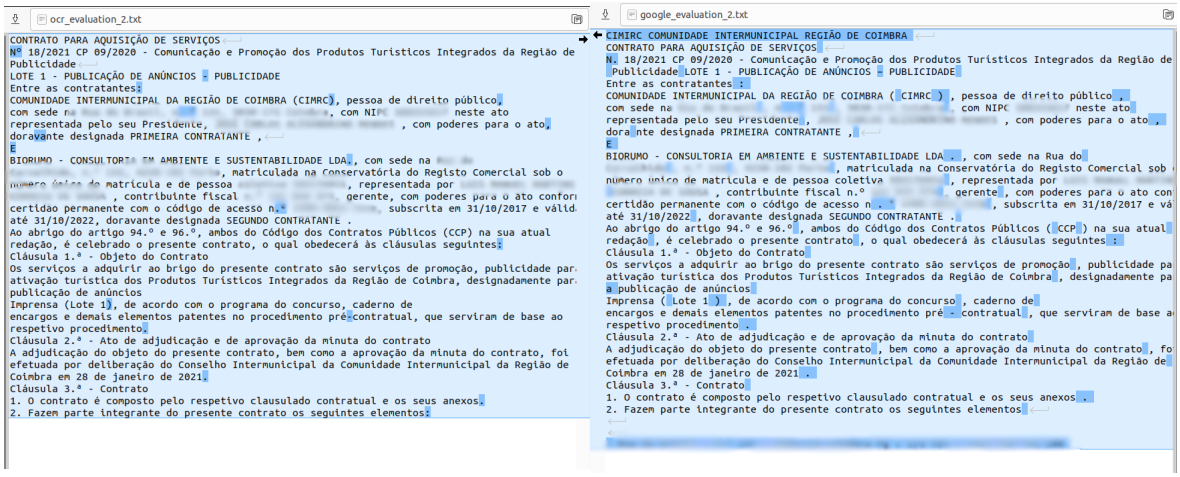


Figure 5.8: On the left, the original document text and on the right the text extracted by Google Cloud Vision OCR, both of them with the differences highlighted in blue

5.2.3 Final results

Having concluded the tests for both technologies, we can compare the performance and conclude that Google Cloud Vision OCR is a better choice for the problem this system is trying to solve and fulfills well its function. Not only it recognized a larger set of words but also symbols and punctuation, specially Portuguese references. It is also worth noting that there is not a big drop in performance as the text size increases. The overall evaluation of both APIs used can be observed on Tables 5.1 and 5.2, which represent respectively, the first and the second test, showing the incorrect words per API, the total number of words on the document and the respective accuracy.

Statistics / APIs	Pytesseract	Google Cloud Vision
Wrong words	5	0
Total words	60	60
Accuracy	91.67%	100%

Table 5.1: OCR Test 1 results for Pytesseract and Google Cloud Vision

Statistics / APIs	Pytesseract	Google Cloud Vision
Wrong words	23	2
Total words	240	240
Accuracy	90.42%	99.17%

Table 5.2: OCR Test 2 results for Pytesseract and Google Cloud Vision

5.3 NLP

The NLP component was implemented using spaCy, which supports Portuguese with a good accuracy and performance, containing a really good and complete POS tagging, but lacking a bit on the NER set of categories, only classifying the entities as people, organizations, locations, or others. The evaluation made was almost completely automated, which would reduce some of the human error analysis. Two different sets of tests were also made, being the second a much more complex version of the first.

Before implementing the automatic tests, two different files were manually created, one for each test, containing the exact same structure. The first line contains a string that is going to be analyzed, with the different words being separated by a white space. The second line contains the corresponding POS for each word above, e.g, if the first word of the first line is "Jonh", then the first word of the second line is going to be PROPN (means proper noun in the spaCy library). The third line represents the respective named entity recognition of each word and this can either be LOC (for locations), PER (for people), ORG (for organizations), MISC (for non-categorized entities) and ND for non-entity words.

The Python script running the tests, is responsible for reading the file and distributing the three different lines to three different lists and the association between the corresponding elements is made by the same index, e.g. the index one in the ner list corresponds to the word in the words list with also index one. Once each information is stored in its according data structure, the spaCy will process the first line of the document and return the corresponding Tokens. Then, for each token, a verification is made to confirm which POS or NER tag the library assign to them and if it corresponds to the values typed on the files, then the answer is counted as correct, otherwise it is counted as wrong.

The first test file contained the following sentence: "O Filipe vive em Portugal". This is a simple sentence but contains two different entities and four different types of POS tags. The file used can be observed in Figure5.9.

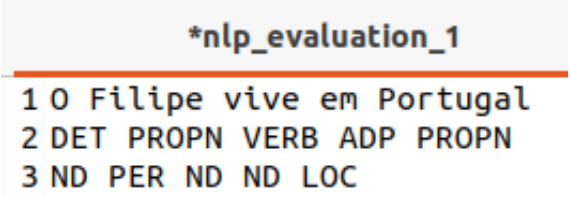


Figure 5.9: The file containing the first nlp test

The results, which can be seen in Figure5.10, of the NLP model were perfect in this test, having identified correctly all of the POS and NER tagging. It identified correctly seven, out of seven annotations possible, which shows an accuracy of 100%.

```
(env) tomasfilipe@tomasfilipe-GP66-Leopard-10UE:~/Desktop/Unive
######### NLP Evalutation 1 ############


#### Part-of-speech ####

Word: O | Solution: DET | NLP: DET

Word: Filipe | Solution: PROPN | NLP: PROPN

Word: vive | Solution: VERB | NLP: VERB

Word: em | Solution: ADP | NLP: ADP

Word: Portugal | Solution: PROPN | NLP: PROPN

Words right: 5
Words wrong: 0

#### Named entity recognition ####

['Filipe', 'Portugal']
['PER', 'LOC']
['PER', 'LOC']
Word: Filipe | Solution: PER | NLP: PER

Word: Portugal | Solution: LOC | NLP: LOC

Words right: 2
Words wrong: 0
```

Figure 5.10: The first test results of the nlp component

The second test file builds from the same structure as the first but it is more complex as it contains more words and more types of POS and NER tags. Its sentence is the following: “O Filipe gosta de ir às compras ao supermercado Auchan grande na Rua Costa Cabral quando está sol”. This phrase includes eighteen words, with nine different POS and three different entities to identify. This text file can be seen in Figure5.11.

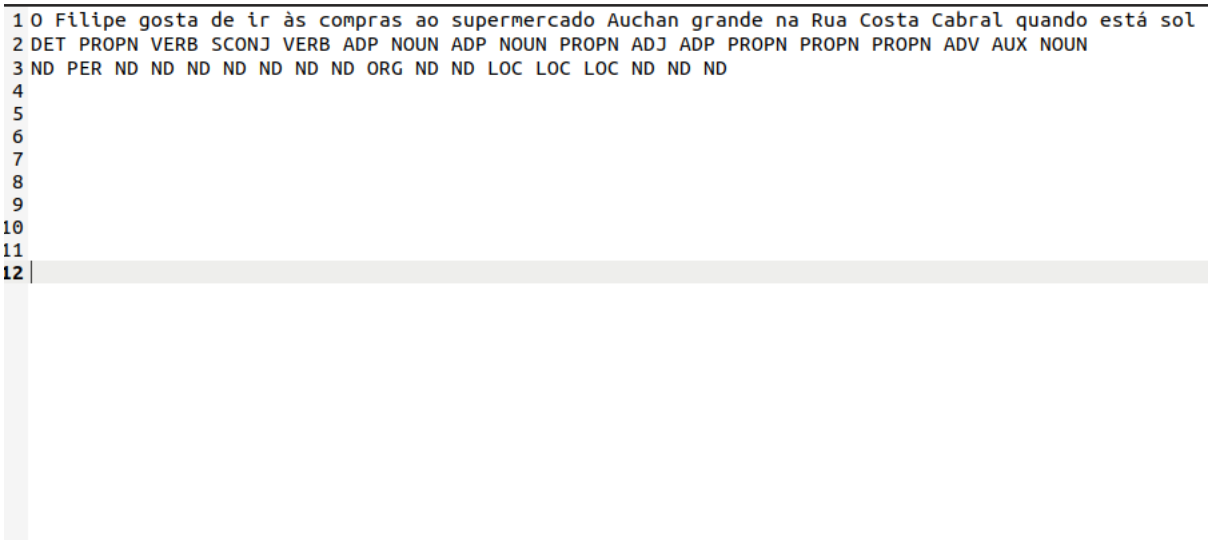


Figure 5.11: The file containing the second nlp test

The results for the second evaluation were not perfect but they were still very positive. It was able to identify correctly the entire five entities and in the POS segment, it recognized seventeen out of eighteen, having wrong only one word: “Auchan”, which it identified as a noun but it is instead a proper noun. In total, the spaCy in the second test categorized 22 out of 23 words right which resulted in an accuracy of 95.65% and this test can be verified in the figure5.12.

```
######### NLP Evalutation 2 ############


#### Part-of-speech ####

Word: O | Solution: DET | NLP: DET
Word: Filipe | Solution: PROPN | NLP: PROPN
Word: gosta | Solution: VERB | NLP: VERB
Word: de | Solution: SCONJ | NLP: SCONJ
Word: ir | Solution: VERB | NLP: VERB
Word: às | Solution: ADP | NLP: ADP
Word: compras | Solution: NOUN | NLP: NOUN
Word: ao | Solution: ADP | NLP: ADP
Word: supermercado | Solution: NOUN | NLP: NOUN
Word: Auchan | Solution: PROPN | NLP: NOUN
Word: grande | Solution: ADJ | NLP: ADJ
Word: na | Solution: ADP | NLP: ADP
Word: Rua | Solution: PROPN | NLP: PROPN
Word: Costa | Solution: PROPN | NLP: PROPN
Word: Cabral | Solution: PROPN | NLP: PROPN
Word: quando | Solution: ADV | NLP: ADV
Word: está | Solution: AUX | NLP: AUX
Word: sol | Solution: NOUN | NLP: NOUN
Words right: 17
Words wrong: 1

#### Named entity recognition ####

['Filipe', 'Auchan', 'Rua', 'Costa', 'Cabral']
['PER', 'ORG', 'LOC', 'LOC', 'LOC']
['PER', 'ORG', 'LOC', 'LOC', 'LOC']
Word: Filipe | Solution: PER | NLP: PER
Word: Auchan | Solution: ORG | NLP: ORG
Word: Rua | Solution: LOC | NLP: LOC
Word: Costa | Solution: LOC | NLP: LOC
Word: Cabral | Solution: LOC | NLP: LOC
Words right: 5
Words wrong: 0
```

Figure 5.12: The second test results of the nlp component

5.4 System

The final version of the system allows it to read the text with great precision, identify personal names, identification numbers, postal codes, phone numbers, organizations, e-mails and permanent certificate numbers. This is possible with the integration of all of the components previously presented.

To evaluate our system, multiple tests were conducted, utilizing each of the data set documents provided and verifying their output. For the first two tests, which will be explained in detail, it was only evaluated one page from two documents, being one an editable and perfectly formatted PDF and another digitized and tilted file. After those two initial tests, the same evaluation was made for the entire of the data set documents, each with all of their pages. For each test, the program is responsible for reading the specific document, identify the names, postal codes, phone numbers, e-mails and permanent certificate, draw the respective

rectangle over them and output the corresponding PDF file. The elements that should be hid were manually registered for a later evaluation.

The evaluation is made based on the number of True Positives (TP), which represents the words that should have been and were hidden, True Negatives (TN), indicates the words number that should not be hidden and weren't, False Positives (FP), representing the words that were hidden but shouldn't have been and False Negatives (FN), indicating the words that should have been erased but were not. With these statistics we can compute, the accuracy (a ratio between the correctly predictions and the total samples), the precision (a ratio of the correctly positive predictions and the total positive samples), the recall (a ratio between the correctly positive predictions and the total of actual positive predictions) and the last one is the F1 score (a computed mean between the precision and the recall).

For the first test, an editable PDF was utilized. This one doesn't need much preprocessing, since the document and its text is well-defined and easily readable. In this page, the following entities that should be private were recognized:

- A name in line five.
- A name in line ten and eleven.
- An e-mail in line eleven.
- A contact number in line eleven.
- A name on the left side of the footnotes.
- The same name on the same side of the footnotes but smaller.
- A telephone number on the left side of the footnotes.
- A name on the right side of the footnotes.
- The same name on the same side of the footnotes but smaller.

In the following Figure5.13, the demonstration of the output for this test can be seen. The system identified and erased correctly all of the items mentioned above, however, it also hid some information that should not be hidden, also called false positives. As a false positive, the word identified two times a common name as being a personal name and a job title as being a continuation of the first name on the list.

Regarding this test, the page contained 406 words in total and the system obtained a performance of finding 33 true positives, 4 words false positives, 0 false negatives and 369 true negatives. The accuracy of this test resulted in 99.1%, the precision obtained a total of 89.18%, the recall was valued as one-hundred percent and the F1 score is computed as 94.29%.

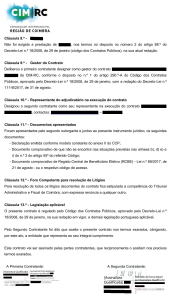


Figure 5.13: Output of the system on this editable pdf page

The second test made used a harder document to identify since it is scanned and the page is tilted and can be see5.14. This implies that some good preprocessing is needed for the text to be well understood. The sensitive items found in this document were:

- A postal code in line three.
- A name in line four and five.
- A postal code in line nine.
- A name in the lines ten and eleven.

For this evaluation, the document contained 333 words in which the system obtained the observations: 15 true positives, 0 false positives, 317 true negatives and 1 false negative, which in a specific name, it didn't hide part of it, with the document text being partially to blame since there is a typo and that word is connected to another one via an underscore. The accuracy in this test increased to 99.7%, the precision increased to 100% percent with no word being wrongly hidden, the recall decreased to 93.75% percent and the final F1 score increased to 96.77.

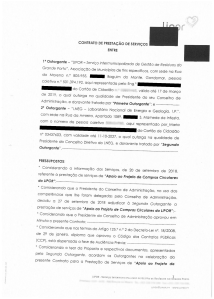


Figure 5.14: Output of the system on this scanned and tilted PDF page

As mentioned previously, this process was repeated for all the provided documents. The results of these tests can be verified in the table 5.3 and 5.4. This evaluation allowed us to conclude, that:

- The system misses very few words that should be anonymized (FN). These words were mainly proper nouns that can be confused with regular nouns, like "Rosa", or "Campos", which might mean respectively "rose" and "fields".
- There are a lot of words that should not be erased by the system, but they are (FP). These are mostly company, job titles or even other types of words that the system perceives them as proper nouns and it greatly affected the F1 score of each document. A reason for this value to be so high is mainly because some word, that might be confused with a proper noun, can appear multiple times throughout the text (e.g. on "Doc 12 test", which had seventeen FP, fourteen of them were because of the same company name).

Statistics / Documents	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
True Positives	20	2	3	9	6	23	38	7
True Negatives	1052	117	252	2549	1507	6106	3632	911
False Positives	4	0	4	7	4	10	12	1
False Negatives	0	1	1	0	0	3	0	0
Total words	1076	120	260	2565	1517	6142	3682	919
Accuracy	99.6%	99.2%	98.1%	99.7%	99.7%	99.7%	99.7%	99.9%
Precision	83.3%	100%	42.9%	56.3%	60%	69.7%	76%	87.5%
Recall	100%	66.7%	75%	100%	100%	88.5%	100%	100%
F1 Score	90.9%	80%	54.5%	72%	75%	78%	86.4%	93.3%

Table 5.3: Results of the final output of the system performed in the dataset documents

Statistics / Documents	Doc 9	Doc 10	Doc 11	Doc 12	Doc 13	Doc 14	Doc 15
True Positives	11	39	12	3	10	74	10
True Negatives	1275	981	903	2764	2425	3206	1319
False Positives	6	3	3	17	3	11	3
False Negatives	0	2	0	0	0	2	2
Total words	1292	1025	918	2784	2438	3293	1334
Accuracy	99.5%	99.5%	99.7%	99.4%	99.9%	99.6%	99.6%
Precision	64.7%	92.9%	80%	15%	76.9%	87.1%	76.9%
Recall	100%	95.1%	100%	100%	100%	97.4%	83.3%
F1 Score	78.6%	94%	88.9%	26.1%	87%	92%	80%

Table 5.4: Results of the final output of the system performed in the dataset documents

CHAPTER 6

Conclusion

This chapter contains the main conclusions drawn from the results of the dissertation and indications about the future work.

This dissertation had the objective to solve a real life problem that is affecting multiple organizations: existing old outdated documents containing public information that should be private towards the new GDPR. The project focuses on the automation of the task of identifying private information on a document and erase it. Since many of these organizations are receiving some backlash for containing the information as it is and the number of documents is too large to be manually changed, the urgency in the completion of this system is greater as the time passes.

The first thing that was done was to investigate and conclude how can the text contained in a document be identified and which technology or API should we use. This problem is covered in the intro section of the OCR 1.2.1 and further developed in the system implementation of this component 3. A conclusion was made to utilize the Google Cloud Vision to support the text recognition of this system.

The next step taken was to find the state-of-the-art method to erase certain segments of text from a PDF document, which can be found in Section 2.3. The best approach to this problem was to convert the PDF to an image and draw a black rectangle over the sensitive information. This guarantees not only the elimination of the word, but also makes it impossible for anyone to bypass the removed word.

The next step was to develop an approach who could understand the semantics of the text and identify this private information. Since our data set is relatively small and didn't have any labeled data, we had to find an already existing model that would best fit our problem. After analyzing a set of NLP libraries and how to make the best use of them, which can be found in the introductory chapter of the NLP, Section 1.2.2, the spaCy was the chosen one with the best support for Portuguese language since it already contained a pipeline trained with a model constructed by two large Portuguese data sets. However, since the Portuguese

support still isn't as big as the other languages, some different edge cases had to be solved by rules-based algorithms.

After being able to identify different types of Portuguese personal data, a requirement gatherings was made with the company who proposed this project and a bit more insight was gained about which data is actually sensitive and how only the single individuals information should be private, instead of every single person in the document.

To better evaluate our system, each component was tested individually and then as a whole. Part of the tests were automated in order to reduce the probability of a human mistake in the evaluation and create a more reliable source of statistics.

6.1 Future Work

Despite the fact that the system has its basic functionalities already implemented and set up, there is still a lot of work to be done before this project is considered complete. Some pending tasks are more important and crucial to the project than others but all of them contribute a bit to the development of this project. A list containing these tasks is going to be presented next:

- At the moment, it is possible to identify different types of information but not to distinguish whom that information belongs to. An important task to work on is to try to associate each information to each person found, to be able to identify which information to erase and to keep.
- We were told by the company when the requirements gathering was made, that only the information about the singular individuals was supposed to be hidden. The information relative to an organization should stay public. Therefore, it is important to develop an algorithm to identify which person is representing a company and who is a singular individual.
- As of this moment, the entire system is initiated and controlled through the command line interface, however, most users are not familiarized with this type of interface. In order to allow non-programmers or non-terminal experience users to utilize the application, it would be important to create a graphical user interface for the system.
- Since some documents can be scanned, it means we might have to deal with tilted pages. To solve this issue, an algorithm to apply skew correction was tried on the document, however, we could not properly and consistently identify at what angle the original image was rotated, and sometimes the skew correction would just rotate the image to an even less readable angle. Since scanned documents are pretty common in this use case, a good future work would be to develop an algorithm to apply a successful skew correction. This would help to improve the OCR in multiple contracts.

References

[1] P. G. Sushant Chandra Saurav Sisodia, "Optical character recognition – a review," *International Research Journal of Engineering and Technology (IRJET)*, vol. 07, no. 04, 2020, ISSN: 2395-0072. [Online]. Available: https://www.irjet.net/archives/V7/i4/IRJET-V7I4583.pdf.

[2] I. C. E. IBM Cloud Education, *What is optical character recognition (ocr)?* 2022. [Online]. Available: https://www.ibm.com/cloud/blog/optical-character-recognition.

[3] R. R. L. v. R. B. S. i. N. Esube Bekele, *Text extraction via optical character recognition*, July 15, 2021. [Online]. Available: https://www.iqt.org/text-extraction-via-optical-character-recognition/.

[4] S. Reddy, *Pre-processing in ocr!!!* 2019. [Online]. Available: https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7.

[5] T. J. Blätte, F. Schmid, S. Mundus, *et al.*, "Binarize," 2019. [Online]. Available: https://cran.r-project.org/web/packages/Binarize/vignettes/Vignette.pdf.

[6] D. S. Changming Sun, "Skew and slant correction for document images using gradient direction," 1997. [Online]. Available: http://engr.case.edu/merat_francis/EECS%5C%20490%5C%20F04/References/Document%5C%20Deskew/00619830.pdf.

[7] G. Mehul, P. Ankita, D. Namrata, G. Rahul, and S. Sheth, "Text-based image segmentation methodology," *Procedia Technology*, vol. 14, pp. 465–472, 2014, 2nd International Conference on Innovations in Automation and Mechatronics Engineering, ICIAME 2014, ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2014.08.059. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017314000954.

[8] B. Lutkevich, *Natural language processing (nlp)*, 2021. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP.

[9] C. Marshall, *What is named entity recognition (ner) and how can i use it?* 2019. [Online]. Available: https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d.

[10] A. Bilogur, *Notes on word embedding algorithms*, 2019. [Online]. Available: https://www.kaggle.com/code/residentmario/notes-on-word-embedding-algorithms/notebook.

[11] A. R. O. Pires, "Named entity extraction from portuguese web text," 2017. [Online]. Available: https://repositorio-aberto.up.pt/bitstream/10216/106094/2/202922.pdf.

[12] M. Dorash, *Machine learning vs. rule based systems in nlp*, 2017. [Online]. Available: https://medium.com/friendly-data/machine-learning-vs-rule-based-systems-in-nlp-5476de53c3b8.

[13] S. Kannan, S. Karuppusamy, A. Nedunchezhian, *et al.*, "Chapter 3 - big data analytics for social media," in *Big Data*, R. Buyya, R. N. Calheiros, and A. V. Dastjerdi, Eds., Morgan Kaufmann, 2016, pp. 63–94, ISBN: 978-0-12-805394-2. DOI: https://doi.org/10.1016/B978-0-12-805394-2.00003-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128053942000039.

[14] V. N. Gudivada, "Chapter 12 - natural language core tasks and applications," in *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, ser. Handbook of Statistics, V. N. Gudivada and C. Rao, Eds., vol. 38, Elsevier, 2018, pp. 403–428. DOI: https://doi.org/10.1016/bs.host.2018.07.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169716118300257.

[15] J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. R. Curran, "Learning multilingual named entity recognition from wikipedia," 2017. [Online]. Available: https://doi.org/10.6084/m9.figshare.5462500.v1.

[16] A. Rademaker, F. Chalub, L. Real, C. Freitas, E. Bick, and V. de Paiva, "Universal dependencies for portuguese," in *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling)*, Pisa, Italy, Sep. 2017, pp. 197–206. [Online]. Available: http://aclweb.org/anthology/W17-6523.

[17] A. Rosebrock, *Text skew correction with opencv and python*, 2020. [Online]. Available: https://pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/.

[18] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, *Thinning*, 2003. [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm.

[19] A. Buades, B. Coll, and J.-M. Morel, "Non-Local Means Denoising," *Image Processing On Line*, vol. 1, pp. 208–212, 2011, https://doi.org/10.5201/ipol.2011.bcm_nlm.

[20] A. S. Filip Zelic, *Opencv documentation - image thresholding - otsu's binarization*, 2022. [Online]. Available: https://nanonets.com/blog/ocr-with-tesseract/.