Research Project

Rapport

Intelligent data extraction from medical reports combining OCR and NER approaches

by

Antoine PRADEL

Research master : Data Science and Machine Learning

ETIS UMR8051, CY Cergy Paris Université / ENSEA / CNRS
6 avenue du Ponceau, 95014 Cergy-Pontoise Cedex, France

Supported on 30 march 2022

In front of the jury composed of :

Pr.	Son Vu	ENSEA	President of the jury
Pr.	Michel Jordan	ENSEA	Reporter

The supervisor :

Pr.	Stefan Bornhofen	CYU Paris Cergy Université

Intelligent data extraction from medical reports combining OCR and NER approaches

Antoine Pradel, *CY Cergy Paris Université*

Abstract—Doctors and healthcare professionals deal every day with an important amount of medical documents and especially medical reports. In order to improve the medical reports processing, and ameliorate the monitoring of the patients, it is necessary to retrieve the key information of each medical report. Named Entity Recognition (NER), one of the most popular tasks in Natural Language Processing (NLP), allows to detect and extract, from an unstructured text, key information and classifies them into predefined categories. Several uses of NER can be found in the literature, but the models are generally based on the English language. Moreover, medical reports are usually in image format. Therefore, it is necessary to preprocess them and extract the text. In this paper, we will introduce a method to identify and extract key information from a French medical report image. This method combines a step, or preprocessing, consisting of image segmentation and text extraction using Optical Character Recognition (OCR) techniques and an implementation of a custom NER. Even with limited data, results are acceptable and validate the proposed method.

Index Terms—French NER, page segmentation, OCR, medicals reports

1 Introduction

Since the dawn of time, man has sought to improve medicine. In the last century, the medical field has grown enormously in complexity. Nowadays, medical reports of interventions are commonplace and hundreds of them are created every day. It would therefore be interesting to be able to quickly find the essential information in medical reports and process it in a system. This would save valuable time for healthcare professionals and doctors and also provide a better follow up of their patients. Several applications could be derived from it. However, unlike parsing standardized documents (homogeneous documents) medical reports are not standardized (heterogeneous documents). It is therefore impossible to recognize the information defined above according to a predefined position. This adds a challenge, because the same information may not be found in the same place. Many systems already exist to extract information from standardized documents, but this is not the case for non-standardized documents. We restricted the scope of the project to the following entities (key information) to be recognized :

- the patient's name, and date of birth
- the date of the medical intervention
- the type of medical intervention (for example : radiology)
- the name of the doctor who performed the medical intervention
- the address of the intervention
- the referring doctor

By extracting these entities, we will know the essential information of the medical report. Named entities Recognition or NER is part of the NLP techniques that allows to retrieve and extract, by the use of machine learning, information from an unstructured text and classify them into categories defined in advance.

Another challenge stands in the format of medical reports. They are usually scanned and therefore stored in the form of images. However, the NER needs text to be able to extract the information. Thus, it is necessary to pre-process these images of medical reports in order to extract the text. Consequently, our proposed approach will combine image pre-processing, including segmentation and text extraction from each image, using Optical Character Recognition techniques and NER to detect and extract the entities defined above from the text.

First, we will present the OCR and how we want to use it. The OCR is presented first because it is not part of the research. It has been studied in a previous work. Optical Character Recognition or OCR is a technique used to translate a scanned or printed image document into a text document [23]. This method allows a computer to automatically recognize text and thus gives it the ability to ”read”.

OCR research is ”relatively old in the field of pattern recognition” [16]. Today, many solutions exist, both open source and commercial. In our case we do not need complex OCR solutions but a simple, direct solution supporting French text recognition. During a previous project, two master 1 students at the University of Cergy carried out a study on the various OCR solutions available and adapted them to our needs. Google Tesseract OCR seems to be ”the most popular choice among the open source category

of OCR solutions" [10]. In addition to being open source, easy to install and to use, a wrapper, pytesseract, has been developed in python. Pytesseract is also useful as a stand-alone invocation script for Tesseract. It can therefore read all types of images managed by python libraries such as Pillow or OpenCV. At the beginning of our research, we used the work of these two students and accepted Tesseract as the OCR library to read our medical reports.

In this paper, we will introduce a technique to extract information from an image document using a combination of computer vision techniques to segment and extract the text (OCR) from the image medical report and natural language processing to recognize the desired information. To segment the image we have created the ACABS algorithm, which allows us to segment the image in text blocks and to remove the blocks containing useless information. We will extract the text contained in the image with the OCR introduced in section II. Finally, the desired information will be extracted from the text using a natural language processing technique called NER (Named entities recognition).

2 Related works

The first question we have to ask ourselves is the following : are there complete solutions allowing to automatically find elements and data defined in advance in a document, if possible medical, not following a precise model? Indeed, there are many solutions for documents following a template, but few offer real solutions for non-standardized documents. However, the company Skilia proposes a solution using artificial intelligence named Laera. Laera has been trained to read medical reports in a human way and to extract some information from them.

2.1 Named Entity Recognition

OCR allows us to extract the text from the image, so we have to recognize in this text the information that interests us. In our case, this information is basic data such as the name and date of birth of the patient, the name of the doctor, the date of the examination, etc... The recognition of information in a text is part of the large family of natural language processing or NLP. It is used for a wide variety of purposes such as knowledge extraction, text mining or web scraping. Named Entity Recognition or NER is a technique that allows to extract information and then to name it with an entity such as a person, a location or a date (i.e. labeling a part of a text). Here is an example of what we can have at the output of a NER process : "John [Person] goes to Paris [Location] on October 12th [Date]". An entity can very well be composed of several words. For example, the entity Person "John Smith" is composed of two words.

NER was already an important topic of the MUC conferences and associated tasks (Marsh and Perzanowski, 1998), and later that of CoNLL 2003 and ACE shared tasks (Tjong Kim Sang and De Meulder, 2003; Doddington et al. 2004). Today there are many NER methods and libraries that have been adapted to different needs. There are already several NER tools applied in the medical domain such as NER for Chinese language medical consultations [25] (Guihua Wen et al 2020) in order to automatically answer certain questions and thus relieve the congestion of medical offices.

There are many NER solutions and many articles comparing existing NER libraries. In order to make our choice, we will base ourselves on three scientific articles. For each of these articles we will present the solutions considered, and the conclusions that were drawn.

2.1.1 First comparaison

Fouad Omran and Christoph Treude, 2017 [1] compared *Google SyntaxNet*, *Stanford CoreNLP*, one of the most used libraries in software engineering with natural language, *NLTK Python* and finally *SpaCy*.

Google SyntaxNet is a parsing model based on TensorFlow and has been made open source since 2016. This model is based on the recurrent parsing model of Kong et al. 2017 and character-based representations learned by a recurrent LSTM (Long Short Term Memory) neural network [2]. A version of this model trained for the English language is called by Google "the most accurate parser in the world".

Stanford CoreNLP is an NLP Framework originally developed in Java to facilitate NLP processes such as annotation thanks to several methods such as tokenization, separation, xml generation, or NER... The NER annotator works on two principles : text entities are recognized with a combination of CRF (Conditional Random Field) sequence markers [14] based on the work of Finkel et al. 2005, numerical entities are recognized through a system of rules. Finkel defines a CFR as a conditional sequence model that represents the probability of a sequence of hidden states [9].

NLTK (Natural language Toolkit) is a software library developed in Python allowing to do many NLP tasks like tokenization of parse trees or NER. NLTK is similar to Stanford CoreNLP and proposes a library architecture oriented in modules.

SpaCy is an open source library developed for NLP task processing. SpaCy uses its own machine learning library Thinc, and is based on the deep learning formula "integrate, code, assist, predict". The integration of words is done through a Bloom filter. The hash of the words is kept as keys in a dictionary. This allows, among other things, to keep a compact dictionary. The encoding consists in encoding the words in a matrix of sentences (to take into account the context) via a convolutional neural network (CNN). The "assist" phase consists in choosing which parts are the most informative according to the query. The class is then predicted using the Sofmax activation function [20] [8].

The comparison of these solutions was performed on software bug texts written in natural language from the 3 sources : Stack Overflow, GitHub ReadMe files and Java API documentation. The evaluation focused on tokenization and part-of-speech tagging (POS) and a comparison of the results between a manual annotation and an annotation made by these NER libraries.

NLTK gives better results for tokenization with an average over the 3 sources of 99.09%. However *SpaCy* performs much better for the comparison of the manual annotations and the labeling of the parts of texts with

results around 90%. *NLTK*, on the other hand, produces much less satisfactory results for the comparison of the manual annotations and the labeling of the general and specific parts of texts. Google's *Stanford* and *SyntaxNet* libraries produce generally weaker results than *SpaCy* or *NLTK*. According to this comparison, *SpaCy* seems to be the library which offers the best performance on these 3 data sources. However NLTK reaches the best accuracy on tokenization. The choice of the NER library depends also on the part of the NLP pipeline used.

2.1.2 Second comparaison

The second study we will use for our comparison was done by Ridong Jiang et al [11]. The libraries *Stanford NER* (previously seen as *Stanford CoreNLP*), *SpaCy*, *Alias-i LingPipe* and *NLTK* are compared. *Alias-i LingPipe* is a tool for word processing using computational linguistics. *LingPipe NER* is based on an n-gram model of character language. These languages allow to define probability distributions $p(\sigma)$ on strings $\sigma \in \sum$, $\sum$ fixed character alphabet [5].

The performance evaluation was done on WikiGold : a corpus of 145 annotated Wikipedia articles (BIO format). The evaluation of these libraries focused on 2 criteria : the recognition of the entity and a good tokenization and the recognition of the entity and a partial tokenization.

For each of these criteria, and for each entity to be recognized, the precision, the recall and the F1 score are calculated. For the criterion of entity recognition and partial tokenization, the libraries produce better results (on average 5 to 10%) than with entity recognition and complete tokenization. The libraries *SpaCy* and *Stanford* achieve the best performances on this dataset with both partial and complete tokenization.

2.1.3 Third comparaison

The third comparison was done by Hemlata Shelar et al. It compares NER approaches for custom models. The 3 libraries compared are : *TensorFlow* with Keras using a Long-Short Term Memory (LSTM) algorithm, the *SpaCy* library and *Apache OpenNLP* using the Token Name Finder model. For each of these libraries a model was created and trained on the IBM Advanced Search Panel dataset. These data are search queries. In order to evaluate the performance of each model, the study is based on 4 criteria : accuracy during training, error during training (training loss), F1 score and prediction probability. The model created with *TensorFlow* performs well, with an accuracy of 98% on the test data. However, the separation of terms of the same entity is a problem for *TensorFlow*. The transformation of one entity into several others makes it hard to count the real number of detected entities and therefore, makes it difficult to use in a real case like ours where we will often find entities composed of several words such as a person's identity (name and surname) or a date in full (e.g. Thursday June 4, 2020). Another disadvantage of the TensorFlow model is that it only accepts numerical data and not text, so it is necessary to add a hot encoding step to transform the text. In addition to achieving better performance on the dataset, *SpaCy* and *Apache OpenNLP* work directly with the data in text format. There is therefore no need to add a data transformation step. *SpaCy* and *TensorFlow* have an incredibly short prediction time (0.2 microseconds) compared to *Apache OpenNLP* (2.3 milliseconds). One of the big problems with *Apache OpenNLP*, in addition to not being able to access the F-score and precision for each entity, is that it treats labeled text parts as unknown during prediction, which creates a problem when generating an SQL query. *SpaCy* also has a disadvantage : the size of its model is much larger than that of the other models.

2.1.4 Conclusion on comparisons

Based on the results of these three studies and the NER solutions considered, *SpaCy* seems to be the most efficient solution in terms of performance and is the most suitable for our needs : open source, fast, efficient, compatible with the French language and allows to create customized models.

2.2 Page segmentation

During the first tests of Tesseract on our medical reports, we noticed that it has a major flaw : the in-line reading. Indeed, Tesseract reads the document line by line without taking into account the layout of the text and the text blocks. In medical reports, the header is often built in blocks containing information about the doctor, the medical facility, information about the patient and sometimes the date of the intervention. For a human, reading in blocks is intuitive but not for Tesseract. The line by line reading therefore mixes this information. It will therefore be very difficult for our NER model to recognize each entity correctly.

In order to overcome this problem, it is necessary to carry out an analysis of the document's layout beforehand. This analysis allows us to identify and then segment an image document by regions of interest (here text blocks).

There are two main families of algorithms for page segmentation. The first family includes algorithms using the bottom-up approach. These algorithms iteratively analyze a document starting with the smallest components (pixels) and then group them into regions, characters, words, lines and then text blocks. On the contrary, the second family of algorithms has a top-down approach. The image of the document is divided several times to form smaller and smaller regions [18]. The bottom-up approach, called traditional, has the advantage of not needing to make assumptions about the global structure of the document. However, due to its iterative nature, this approach can be time-consuming. The top-down approach allows to directly analyze the layout of a document without having to group regions of interest, but generally requires assumptions on the document structure. In order to obtain good performances for these algorithms, a pre-processing phase of the image is necessary. This pre-processing phase mainly consists of a noise removal, a background detection and a tilt correction if the document is tilted.

We will therefore present several algorithms for the segmentation of pages and then a comparative study of these algorithms.

The X-Y cut algorithm [17] is a top-down tree-based algorithm that recursively divides the document into blocks,

each leaf representing the final segmentations. At each step, the horizontal and vertical projections are computed. The algorithm then compares these projections to predefined thresholds to perform a segmentation of the page at the right place.

The Smearing algorithm [26] works on binary images. By applying a transformation on the rows and columns of pixels it allows to group together components and then to classify them as text or non-text.

The Whitespace analysis algorithm [3] searches for the largest background rectangles in the binary image (i.e. white rectangles). The borders of the regions not included in these rectangles are defined as the text boundary frames.

The Constrained text-line detection algorithm [4] searches for large (in terms of height) white rectangles that it considers as column separators in the text. These rectangles are then used as delimiters for a line detection algorithm in the text.

The Docstrum algorithm [19] uses a bottom-up approach with a connected nearest neighbor clustering. This clustering allows, by computing the transitive closure of each pair of neighbors, to find the lines. The lines are then grouped into blocks using a threshold for vertical and horizontal distances.

The algorithm based on Voronoi diagrams [12] is also a bottom-up approach using the generation of a Voronoi diagram on the image document.

The comparison of these 6 algorithms was done by Shafait et al. 2006 [22]. The evaluation of these algorithms is performed on the UW-III database. In order to perform the evaluation, the "text-line detection accuracy" metric was used.

The X-Y cut appears to be sensitive to noise. The Smearing Whitespace and text-line algorithms produce too many errors compared to the other algorithms and are inconsistent (large standard deviation). The Docstrum and Voronoi algorithms give, on this dataset, a relatively low error and are rather constant in their results (about 5% error with optimized parameters).

Medical reports also very often contain bold text. This bold text is usually very important and often corresponds to one of the entities we are looking for. It would therefore be interesting to be able to detect whether a word or words are bold or not on a document.

Several researches have already been done on this subject. In 1998, Chaudhuri Garain proposed a method to detect bold italic and capital letters words. The detection of bold words is based on the stroke length of black pixels along several directions at certain boundary points [6]. This information is used to generate a histogram which after analysis allows to conclude whether the word is bold or not. The evaluation of this method gives good results. In 2004 Ma et al suggested an article on the classification of word style. It consists of a training step where an exhaustive feature selection is performed and then a step of style classification (blod, italic or normal) using these features [13]. It also performs well, with an accuracy of about 90%. The algorithm MOBDoB [21] proposed by Saikrishna et al uses a morphological opening based detection of bold approach with binarized images.

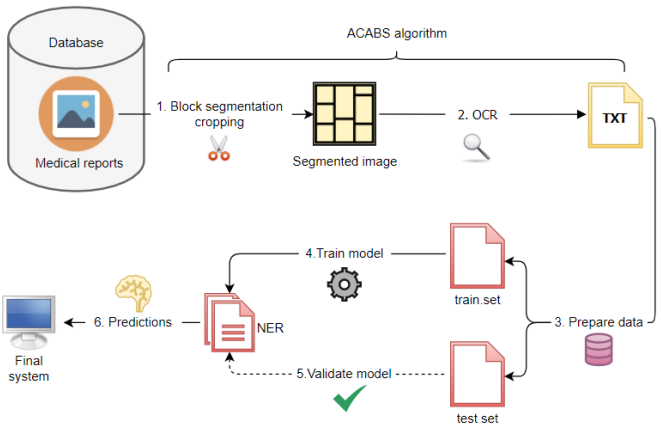


Fig. 1: System architecture

3 Proposed Method

The goal of this research is to develop a system capable of extracting from non-homogeneous image documents the information (entities) that we defined in section II, such as doctor name or dates. The solution that seems to be the best answer to our problem is a combination of OCR for the detection of text in the image and NER techniques to find these entities in the text. However, after the first tests of NER on the text extracted from the images thanks to the Tesseract library, an important problem that could affect the efficiency of the system was detected. As mentioned in the literature review, Tesseract has one major flaw : inline reading. Tesseract reads a document line by line without taking into account the layout of the text and the text blocks. In medical reports, the layout of the information is often organized in blocks, especially in the first part and the header of the medical report. When using OCR on a medical report, this information is mixed up. The extracted text provided to the NER is different from the one in the image of the medical report.

Therefore we will add a data preprocessing step using page segmentation techniques. The ACABS algorithm (see section IV.A) aims at preprocessing the image by segmenting and cropping it in order to avoid mixing the text when extracting the text from the image.

The final system is thus composed of a segmentation of the medical report image, an extraction of the text from the image thanks to OCR and then the recognition of the entities thanks to a NER model. Figure 1 summarizes the architecture of the system. We will first introduce the ACABS algorithm and then discuss the NER model.

3.1 ACABS algorithm

ACABS stands for Automatic Cropper And Block Segmenter. This algorithm first segments an image document into blocks of text, removes the blocks of text containing unnecessary information and then extracts the text through OCR. It allows us to preprocess each medical report and then extract the text using OCR.

Indeed Tesseract has a defect that can strongly impact the performance of the final system : not being able to read text blocks by blocks. By segmenting the image in blocks of

text and passing them one by one to the OCR this problem disappears and the text extracted is not mixed anymore. In order to detect the text blocks in an image and to segment it according to the blocks found, the ACABS algorithm performs several treatments on the image.

First, the image is binarized using the Otsu adaptive thresholding method. Since the background of medical report images is uniform but may contain some noise if the quality of the photo is poor, Otsu method is perfectly adapted. Local image binarization methods are not necessary.

Then the ACABS algorithm uses mathematical morphology methods to process the image content. A dilation with a K1 kernel (matrix of 1) is applied to the binarized image. Thus the text is transformed into black pixel blocks. Then an erosion (optional step) with a K2 kernel (matrix of 1) can also be applied to the image in order to refine the bounding boxes of the blocks that will be created in the next step.

After dilating and eroding the image, ACABS will detect the contours of the previously created black pixel blocks. To do this we will use the Python implementation of the contour detection algorithm of Suzuki et al.1885 [24].

Finally, the bounding boxes of the text blocks are generated from the sets of contours found previously. The algorithm therefore has two parameters : the size of the kernel K1 and the size of the kernel K2, both matrices of 1. The larger the K1 kernel is, the larger and coarser the detected bounding boxes will be. The kernel K2, on the other hand, allows to refine the bounding boxes found.

It is then possible to extract the text from each block by using the bounding boxes to delimit the new image containing only the text block and to use OCR on this new image.

The ACABS algorithm also removes unnecessary text blocks and thus saves time and improves performance. In fact, when we first looked at the medical reports, we noticed that many of them contain a header and a footer. The information contained in this header and footer is very general and not useful in our case. For example, the header often contains the complete and very detailed name of the medical facility as well as the name, title and more of all the doctors working there. The footer usually contains legal information.

In order to remove this unnecessary information contained in the header and footer, the ACABS algorithm will apply the OCR and extract the text only to a selection of text blocks. To do this, a top horizontal threshold and a bottom horizontal threshold are set on the image. These thresholds correspond to a percentage of the image. This percentage is defined as the crop_percentage parameter of the ACABS algorithm. All text blocks above the top threshold and below the bottom threshold are excluded and will not be sent to the OCR step for text extraction. However, it is quite possible that one of the thresholds, top or bottom, passes through the middle of a text block. If this is the case, the threshold is lowered to the bottom of the block. This way, the threshold is no longer in the middle of a block of text and the algorithm can choose to integrate or exclude this block from the OCR step (see appendix 1 for an example of the threshold adaptation).

Algorithm 1: ACABS algorithm

Input: Image document: img
Output: text
parameter: K1, K2, crop_percentage

```
binary_img ← Otsu(img);
img ← Dilate(binary_img, K1);
img ← Erode(img, K2);
contours ← FindContours(img);
bounding_box ← BuildBox(contours);
create thresholds using crop_percentage;
adapt thresholds to bounding_box;
for box in bounding_box do
    if box is between the thresholds then text +=
     OCR(box);
    else pass;
end
return text;
```

3.2 NER model

Now that we have extracted the text of the medical reports, we need to recognize and extract the following entities :

- doctor's name : DOCTOR
- referent doctor's name : REFERING_DOCTOR
- patient's name : PATIENT_NAME
- patient's birth date : PATIENT_BIRTHDATE
- address of the medical facility (city) : ADRESS
- date of procedure : DATE
- type of procedure : TYPE

These entities are the key information of a medical report. Before we can create a SpaCy model to recognize these entities, we will need to generate an annotated dataset. This annotated text data must respect a precise syntax. Here is an example of this syntax with the extract "Doctor Brunaud performs a radiology" :

["Dr. Brunaud fait une radiologie", {'entities': [[0,11, DOCTOR_NAME], [21,31, TYPE]}]

There are many annotation tools that allow us to follow this syntax. SpaCy offers its own annotation tool, Prodigy. However, Prodigy is not free and there are many similar open source tools. We will therefore use the NER Annotator software library from Arunmozhi [1].

Our dataset was provided by DocAsssyt. It is composed of 400 medical reports in PNG and PDF format. PDF format has been changed to PNG format thanks to the pdf2image library [2]. It is composed of 74 medical reports coming from various medical facilities, and 326 medical reports coming from five medical facility sources.

These medical reports have been anonymized in order to respect the anonymity of the patients. They come from different medical facilities. Thus, there are different layouts of medical reports that represent the variety of layout we can find in French medical reports. As data annotation is a

1. https://github.com/tecoholic/ner-annotator
2. https://github.com/Belval/pdf2image

very time-consuming task and the research project time is not unlimited, we did not try to enlarge the dataset size.

Before training the model with our manually annotated data, we need to split the data into a training and a testing set. The training set will be used to fit the model, while the testing set will be used to provide an unbiased performance evaluation of the final model. The dataset is split into 75% training set and 25% testing set, that is 300 examples for training and 100 examples for testing set. Since the NER model can be sensible to imbalanced data, we tried to make it as balanced as possible (see appendix 2).

The NER searches for text objects e.g. a word or a group of words belonging to a class, a category defined during the design of the NER model such as a place, a name of people, a date etc. In our case, we want to extract the entities defined above. For this, we will use the SpaCy library (version 3.2). SpaCy supports, in version 3.2, more than 64 different languages, including French. There is already a NER model pre-trained by SpaCy : 'fr_core_news_lg' trained on the UD French Sequoia 2.8 and WikiNER datasets. However, this model is only trained on four entities : Locations, Miscellaneous entities, (e.g., events, nationalities, products) People and Organizations (LOC, MISC, PER, ORG).

We will therefore create our own model from scratch so that it recognizes the entities defined above and train it on our training dataset. In order to reduce the training time, which can be quite long for a NER model, we trained the model on a GPU (Nvidia Quadro RTX 8000).

4 Experiments and Results

In order to evaluate the proposed method we have separated the evaluation into two parts. First we will evaluate the ACABS algorithm and then we will evaluate the NER model created to extract the information defined in section 2.

4.1 ACABS algorithm evaluation

The ACABS algorithm is composed of three main steps : segmentation, cropping and text extraction. We will evaluate the segmentation and cropping step. In order to perform the evaluation, we will use an adapted version of the "text-line detection accuracy" metric [15] that we have already mentioned in section II. We will use the Docstrum algorithm as a point of comparison for the segmentation step. Since Doctsurm does not do any cropping, we will evaluate the ACABS cropping error , but there will be no comparison with another algorithm. As there is no training of the algorithm but a search for the best parameter value, the dataset used will be composed of 74 images of various medical reports provided by DocAssyst.

We will first evaluate the segmentation part of the ACABS algorithm. The segmentation of an image into blocks can generate good blocks, but also can miss, merge or split certain blocks and also create false positives. We will therefore create 2 error formulas : one taking into account the false positives, missed, merged and split blocks (equation 1) and the other without the false positives (equation 2).

$$seg\_errorFP = \frac{M + S + F + FP}{T} \quad (1)$$

$$seg\_error = \frac{M + S + F}{T} \quad (2)$$

M is the number of undetected (missed) text blocks, S is the number of text blocks split into several blocks, F is the number of text blocks detected as one (merged). FP is the number of false positives generated in the document, and T is the number of true text blocks we would like to find (ground truth).

These two errors are calculated for each of the 74 medical reports, and the mean of these errors is then computed. For the ACABS algorithm, we tested for the parameter K1 (size of the dilation kernel) the following values : 10, 12, 15 and 20. For the Doctsrum algorithm we choose 15 and 20 nearest neighbors as values of the main parameter. The results are presented in figure 2 where k10, k12, k15 and k20 are the values of the parameters K1 of the ACABS algorithm and doc_15 and doc_20 are the parameter values for the Docstrum algorithm.

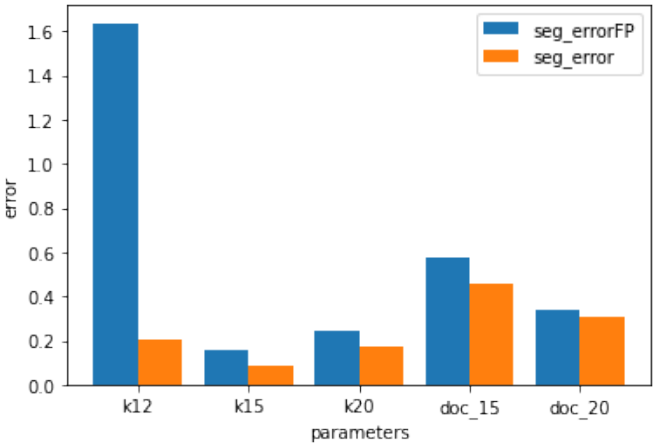


Fig. 2: Segmentation error comparison

As we can see in figure 2, the ACABS algorithm has a lower error than Docstrum algorithm with and without false positives for a K1 value of 15 or 20. For ACABS with K1 = 15, the seg_errorFP is 0.15 (and seg_error = 0.08). For Doctsrum with 20 neighbors the seg_errorFP is equal to 0.57 (and seg_error = 0.45). The ACABS algorithm, with K1 equal to 15 or 20, also has a lower standard deviation. It appears that the ACABS algorithm performs better than the Doctsrum algorithm on our medical report image dataset. The complete table of results of this evaluation, as well as an example of robustness to font size change of the K1 value, is available in Appendix 3.

We will now measure the error realized by ACABS during its cropping step. As a reminder, the crop_percentage parameter is the parameter that defines the position of the top and bottom horizontal threshold. The text blocks outside these thresholds will be excluded from the OCR step. The cropping error can therefore be of two types : 1) ACABS crops a block of text that should not be cropped, 2) ACABS does not crop a block of text that should be cropped. Two error formulas were therefore created. The equation 3 only takes into account only the first type of

error, while the equation 4 takes into account both. Since the second type of error is less important, it has a lower weight in the equation 4.

$$crop\_error1 = \frac{BCT + BCB}{T} \tag{3}$$

$$crop\_error2 = \frac{BCT + BCB + 0.25(BCTI + BCBI)}{T} \tag{4}$$

BCT and BCB are respectively Bad Crop at the Top of the document and Bad Crop at the Bottom of the document e.g the first type of error where ACABS crops text blocks that should not be cropped. BCTI and BCBI are the same but for the second type of error where ACABS do not crop a text block that it should be cropped.

As for the segmentation evaluation, the two errors are calculated for each of the 74 medical reports and then the average is done. In order to find the best value of the crop_percentage parameter we tested the following : 8%, 10%, 11%, 12%, 15%, 17%. The calculated errors are presented in the table 1.

TABLE 1: Cropping evaluation results

Crop_percentage	crop_error	crop_error_full
8	0.000	0.159
10	0.000	0.064
11	0.068	0.132
12	0.081	0.101
15	0.176	0.189
17	0.216	0.230

According to the table 1 the value of crop_percentage that generated the smallest error is 10 with a cropp_error of 0.00 and a crop_error_full of 0.064. The small errors for the values of 8 and 10 can be explained in part by the small size of the dataset. In order to have accurate errors we would need to compute these errors on a much larger dataset.

Thanks to these evaluations, we can conclude on the best values to use for the parameters K1 and crop_percentage for preprocessing and extracting text from the images of our dataset. These values are respectively 15 for K 1 and 10 (percent) for crop_percentage.

In figure 3 we can see an example of segmentation and cropping performed by the ACABS algorithm. The red lines correspond to the top and bottom thresholds. The text blocks (in green) that are not between the two thresholds will not be sent to the OCR step.

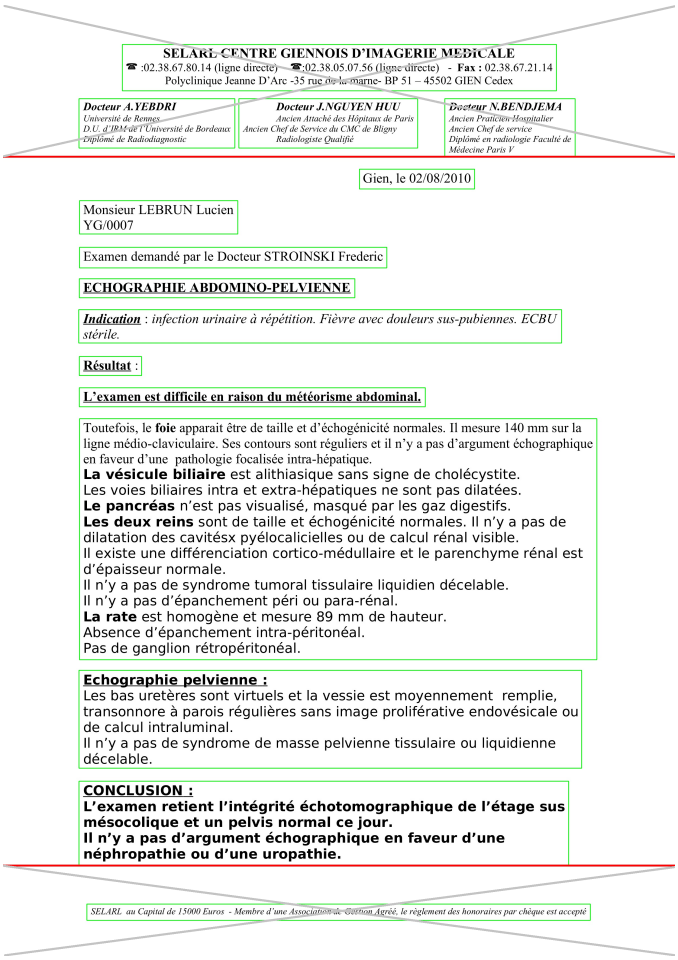


Fig. 3: Exemple of ACABS segmentation and cropping

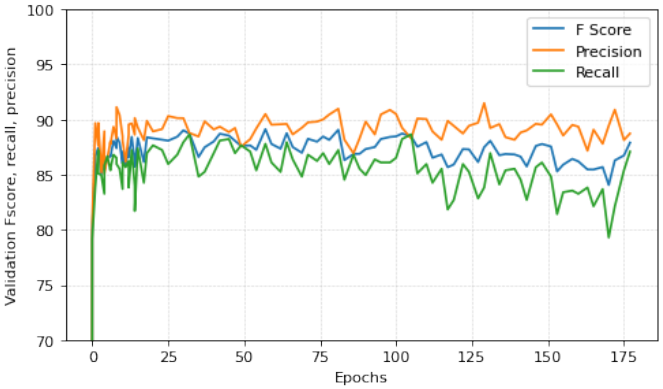


Fig. 4: Precision, Recall, F-score evolution

4.2 NER model evaluation

In order to evaluate our NER model, we will use our annotated dataset of 400 medical reports. This dataset has been manually annotated on the entities : DOCTOR, REFERING_DOCTOR, PATIENT (name), BIRTHDATE (of the patient), ADRESS, DATE, TYPE (of intervention).

As a reminder, the dataset has been separated into two parts : 75% for the training set (2254 entities) and 25% for the testing set (705 entities). The evaluation of the model will be based on a testing set with three metrics : (i) precision to measure the ability of a system to present only the relevant names; (ii) recall to measure the ability of a system to present all the relevant names; and (iii) F-measure, which is a harmonic mean of precision and recall [7]. We will calculate the evolution of these metrics during the training of the model (e.g. numbers of epochs) and then we will calculate these metrics for each entity in the final model.

The figure 4 shows the evolution of precision, recall and F-score as a function of the number of epochs. We are particularly interested in the F-score, a metric that achieves

	Precision	Recall	F-Score
PATIENT	97.50	95.90	96.69
BIRTHDATE	100.00	96.61	98.28
REFERING_DOCTOR	55.68	62.82	59.04
ADRESS	98.89	98.89	98.89
DATE	97.83	90.91	94.24
TYPE	88.07	84.96	86.49
DOCTOR	94.53	84.03	88.97

TABLE 2: Final model metrics

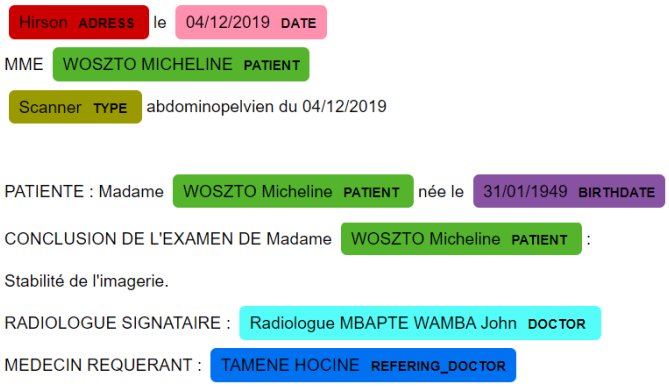


Fig. 5: Exemples of entities recognition

a balance between precision and recall. We can see that the model quickly reaches a precision, an F-score and a recall close to 90%. The model reaches a satisfying level of entity prediction. The overall precision of the model is 90.50%, the recall is 87.50% and the F-score is 89.13%. The loss, computed between the entities of the test set and the predicted entities, also decreases rapidly (see Appendix 4). The table 2 shows the three metrics calculated for the final model. We have very good results (F-score) for the entities PATIENT : 96.69%, BIRTHDATE : 98.28%, ADRESS : 98.89%, DATE : 94.24%, TYPE : 86.49% and DOCTOR : 88.97%. However, the results for the entity REFERING_DOCTOR are less good, with an F-score of only 59.04%. As a matter of fact, the referring doctor is one of the entities with the most varied formatting in the dataset. We therefore suppose that the NER model needs a lot of data to properly learn how to recognize the entities, but only 236 REFERING_DOCTOR entities are provided.

We can see in figure 5 an example of entity detection realized with our NER model. Each word or group of words detected as an entity is surrounded and labeled with the name of the entity predicted by the model.

The model correctly detects the entities present in the text. However, this does not mean that our model will be able to correctly detect the entities present in any French medical report. Appendix 5 presents examples of missed or bad detection of entities by the model. Indeed, the model is trained on a very small number of data : 300 medical reports, around 2200 entities from few different sources. The variety of the layout but also the variety of the values for the entities ADRESS, DATE and TYPE is small.

5 Discussion

The ACABS algorithm that we have proposed allows for a good preprocessing of medical reports, and thus permits a better extraction of the text than using the Docstrum algorithm and OCR. Indeed, the segmentation error is much lower with ACABS than with Docstrum. Moreover, ACABS allows the medical report to be cropped and therefore to ignore the text contained in the header and footer with a very low error. In terms of execution time for the segmentation and OCR, on a CPU I7, ACABS is also faster with an average time of 0.136 second per image while Docstrum takes about 0.4 second. However, ACABS tends to generate more false positives text blocks : 1.20 per image against only 0.39 per image with Docstrum (20 nearest neighbors). This problem is partly compensated by a safety measure in ACABS. As the OCR part represents the majority of the processing time, a method to check the ratio of the amount of white and black pixels has been added before sending a block of text to the OCR stage. If this ratio is greater than a threshold, i.e. there are no or too few black pixels, this text block is considered as a false positive and is not sent to the OCR stage (cf Appendix 3).

Our NER model is able to detect, with good performances, the name and birthdate of the patient, the name of the doctor, the address and the date of the medical report and the type of intervention performed. The proposed approach allows to recognize and extract the previously defined entities from a medical report in French language. Nevertheless, it is important to underline that the dataset used is small, with only 400 medical records or about 3000 entities. Indeed, the annotation of the data, done manually, takes a considerable amount of time. The model could be greatly improved and be able to generalize more with a larger dataset.

6 Conclusion

During this research, we designed and proposed an approach to identify and extract key information from a French medical report in a not-template-based image format. The ACABS algorithm pre-processes the image and extracts the text of the medical report using OCR techniques. This text is then analyzed by our NER model. The proposed model is able to detect and extract key information. However, in order to make it more generic and to be able to use it in a production environment, a large and varied dataset coming from multiple sources should be used. Sufficient time must also be devoted to the annotation of the data because this step is very time-consuming, and the quality of the annotation can affect the performance of the model.

The current approach and more specifically the ACABS algorithm does not take into account the style of words, and more specifically bold italic or underlined words. Word style often provides additional information about the value of the words. A bold or underlined word usually contains important information of the text. It would therefore be interesting to be able to detect the style of words and integrate it into our method to improve its performance.

The NER model allows, with a sufficient dataset, to detect new entities. Thus, many applications can be derived

from the extraction of information from medical reports using this approach, such as an automatic patient monitoring system to help doctors, an application allowing patients to understand their medical reports, etc...

7 Acknowledgment

We would like to thank the ETIS Laboratory (Cergy Paris - France) for providing computing resources needed for this work. Thanks to David Fortier for having partly supervised and followed the evolution of the project. We would also like to thank Fouad Aouinti for his collaboration and his precious help throughout the project. Finally, a big thank you to Stefan Bornhofen, supervising professor, for his help from the beginning to the end of this project.

References

[1] Al Omran, F. N. A., and Treude, C. Choosing an nlp library for analyzing software documentation: A systematic literature review and a series of experiments. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (2017), pp. 187–197.

[2] Alberti, C., Andor, D., Bogatyy, I., Collins, M., Gillick, D., Kong, L., Koo, T., Ma, J., Omernick, M., Petrov, S., et al. Syntaxnet models for the conll 2017 shared task. *arXiv preprint arXiv:1703.04929* (2017).

[3] Baird, H. S. Background structure in document images. *International Journal of Pattern Recognition and Artificial Intelligence 8*, 05 (1994), 1013–1030.

[4] Breuel, T. M. Two geometric algorithms for layout analysis. In *International workshop on document analysis systems* (2002), Springer, pp. 188–199.

[5] Carpenter, B. Lingpipe for 99.99% recall of gene mentions. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop* (2007), vol. 23, Citeseer, pp. 307–309.

[6] Chaudhuri, B. B., and Garain, U. Automatic detection of italic, bold and all-capital words in document images. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)* (1998), vol. 1, IEEE, pp. 610–612.

[7] Eltyeb, S., and Salim, N. Chemical named entities recognition: a review on approaches and applications. *Journal of cheminformatics 6*, 1 (2014), 1–12.

[8] Explosion. Spacy documentation, 2021.

[9] Finkel, J. R., Grenager, T., and Manning, C. D. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)* (2005), pp. 363–370.

[10] Jain, P., Taneja, K., and Taneja, H. Which ocr toolset is good and why: A comparative study. *Kuwait Journal of Science 48*, 2 (2021).

[11] Jiang, R., Banchs, R. E., and Li, H. Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop* (2016), pp. 21–27.

[12] Kise, K., Sato, A., and Iwata, M. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding 70*, 3 (1998), 370–382.

[13] Ma, H., and Doermann, D. Adaptive word style classification using a gaussian mixture model. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* (2004), vol. 2, IEEE, pp. 606–609.

[14] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* (2014), pp. 55–60.

[15] Mao, S., and Kanungo, T. Empirical performance evaluation methodology and its application to page segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 3 (2001), 242–256.

[16] Mori, S., Suen, C., and Yamamoto, K. Historical review of ocr research and development. *Proceedings of the IEEE 80*, 7 (1992), 1029–1058.

[17] Nagy, G., Seth, S., and Viswanathan, M. A prototype document image analysis system for technical journals. *Computer 25*, 7 (1992), 10–22.

[18] Namboodiri, A. M., and Jain, A. K. Document structure and layout analysis. In *Digital Document Processing*. Springer, 2007, pp. 29–48.

[19] O'Gorman, L. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 11 (1993), 1162–1173.

[20] Partalidou, E., Spyromitros-Xioufis, E., Doropoulos, S., Vologiannidis, S., and Diamantaras, K. I. Design and implementation of an open source greek pos tagger and entity recognizer using spacy. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (2019), IEEE, pp. 337–341.

[21] Saikrishna, P., and Ramakrishnan, A. Script independent detection of bold words in multi font-size documents. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)* (2013), IEEE, pp. 1–4.

[22] Shafait, F., Keysers, D., and Breuel, T. M. Performance comparison of six algorithms for page segmentation. In *International Workshop on Document Analysis Systems* (2006), Springer, pp. 368–379.

[23] Shinde, A. A., and Chougule, D. Text pre-processing and text segmentation for ocr. *International Journal of Computer Science Engineering and Technology 2*, 1 (2012), 810–812.

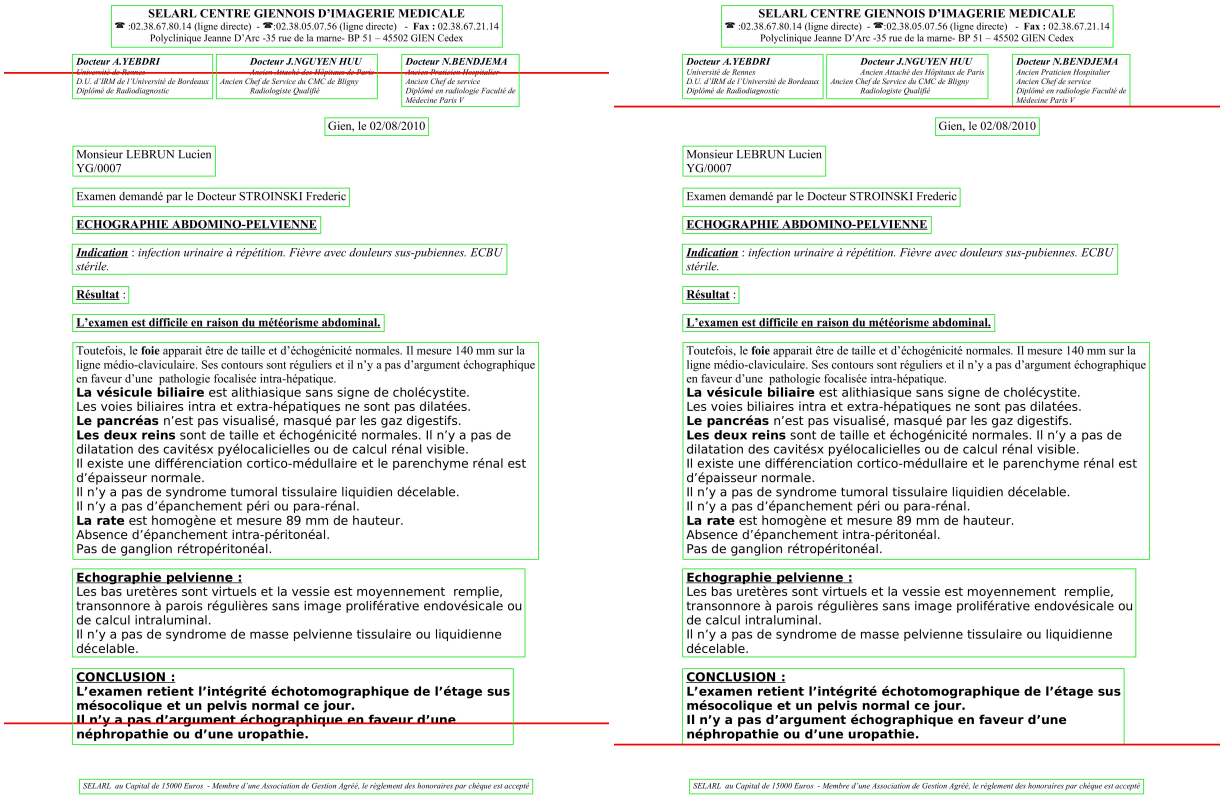
[24] Suzuki, S., et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing 30*, 1 (1985), 32–46.

[25] Wen, G., Chen, H., Li, H., Hu, Y., Li, Y., and Wang, C. Cross domains adversarial learning for chinese named entity recognition for online medical consultation. *Journal of Biomedical Informatics 112* (2020), 103608.

[26] Wong, K. Y., Casey, R. G., and Wahl, F. M. Document analysis system. *IBM journal of research and development 26*, 6 (1982), 647–656.

1 Appendix

1.1 Appendix 1



(a) ACABS threshold before adaptation

(b) ACABS threshold after adaptation

FIGURE 1 – ACABS threshold adaptation

Figure 1 shows an example of the adaptation of the ACABS algorithm's thresholds. These thresholds are used to remove the text blocks (in green) from the header and footer. The initial position of the thresholds (red lines) is defined with the crop_prercentage parameter. If these thresholds pass through a text block, as it is the case in the figure 1a, they are lowered to the bottom of it (figure 1b). ACABS can then decide whether or not to integrate this block of text to the OCR stage where the text is extracted from it.

1.2 Appendix 2

	DOCTOR	DATE	REFERING_DOCTOR	ADRESS	PATEINT	BIRTHDATE	DATE
train	390	301	236	299	429	203	396
test	144	99	78	90	122	59	113

TABLE 1 – NER Dataset entities distribution

1.3 Appendix 3

Algo + parameters	seg_errorFP	seg_error	STD (with FP)	STD (no FP)	Avg FP per doc	Time (s), avg on 20 rounds
K8	5.208262	0.812046	2.587197	0.591313	73.175676	0.131
K12	1.634895	0.206811	1.996838	0.254957	20.675676	0.132
K15	0.154637	0.087487	0.180071	0.090965	1.202703	0.134
K20	0.243085	0.174336	0.185886	0.133494	1.189189	0.136
Docstrum 15	0.573985	0.457685	0.793448	0.614237	1.648649	0.375
Docstrum 20	0.340999	0.309089	0.304709	0.274242	0.391892	0.424

TABLE 2 – Segmentation results

Table 2 shows the complete results of the segmentation evaluation performed by the ACABS algorithm and the Docstrum algorithm with different parameter values. ACABS with parameter K1 equal to 15 produces the lowest seg_error and seg_errorFP. ACABS with K1 equal to 15 or 20 is also more consistent on the errors produced based on the medical reports (lower standard deviation) whether with or without the false positives. The average execution time per medical report of Docstrum and OCR, whatever the parameter, is higher than the ACABS algorithm. This execution time is an average of the execution times on 74 medical reports. In order to decrease the variance of this execution time, the measure was performed 20 times. Nevertheless, ACABS has a significant flaw : it generates a lot of false positives when segmenting an image into text blocks. These false positives are mostly located between 2 spaced lines in the same block of text or near logos and images in medical reports. Since OCR represents the majority of the execution time of the ACABS algorithm, it is necessary to filter these false positives and thus reduce the execution time. Therefore, before extracting the text from a block, the algorithm checks the percentage of "white" pixels in the block. If this percentage is higher than 95% the block is considered as a false positive and the OCR is not applied. This percentage has not been set to 100% because some blocks may contain a small percentage of "black" pixels if they are next to an image or overlap with another text block.

(a) Example 1 (b) Example 2 (c) Example 3

FIGURE 2 – Example of ACABS robustness to font size change

In order to verify that the value chosen for the K1 parameter of the ACABS algorithm is the best, we performed a robustness test to the font size change. We created an image document with several very different font sizes, ranging from 11 to 80. We then used the ACABS algorithm to segment the image and generate the text blocks. The figure 2 shows examples of this test. We can see on the figures 2a and 2b that the ACABS algorithm correctly detects the text blocks, even with a font size of 55 for the title in figure 2b. Some false positives appear but, with the countermeasure described above, the false positives are less problematic. In figure 2c, ACABS correctly detects the two lower text blocks, but has difficulties

detecting the first block of text. In this extreme case (font size equal to 80) the algorithm will not be able to detect the text blocks correctly, and it will be split into smaller text blocks. However, there is very little chance that such a font size is used in medical reports.

1.4 Appendix 4

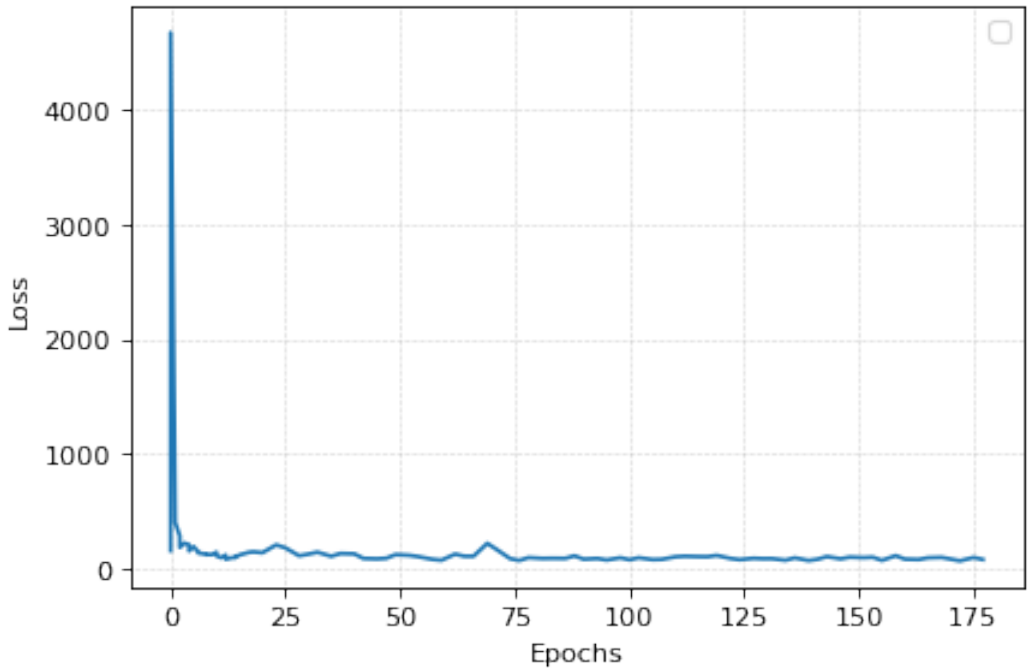


FIGURE 3 – Loss evolution over the epochs

1.5 Appendix 5

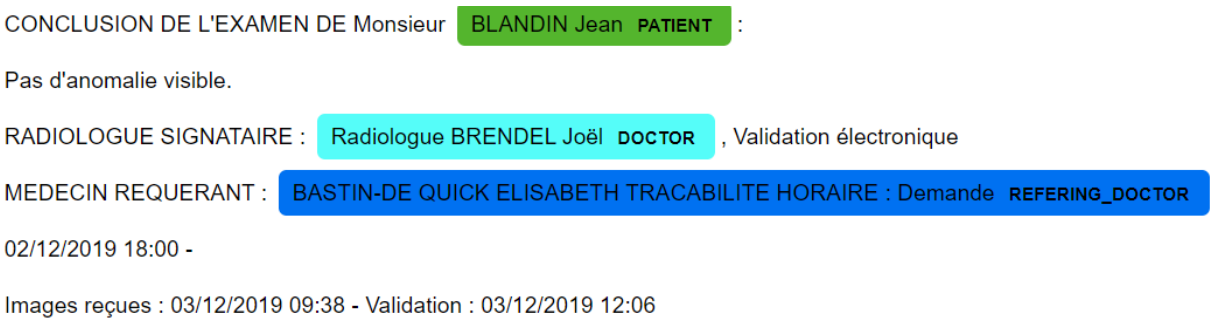


FIGURE 4 – Example bad entities detection

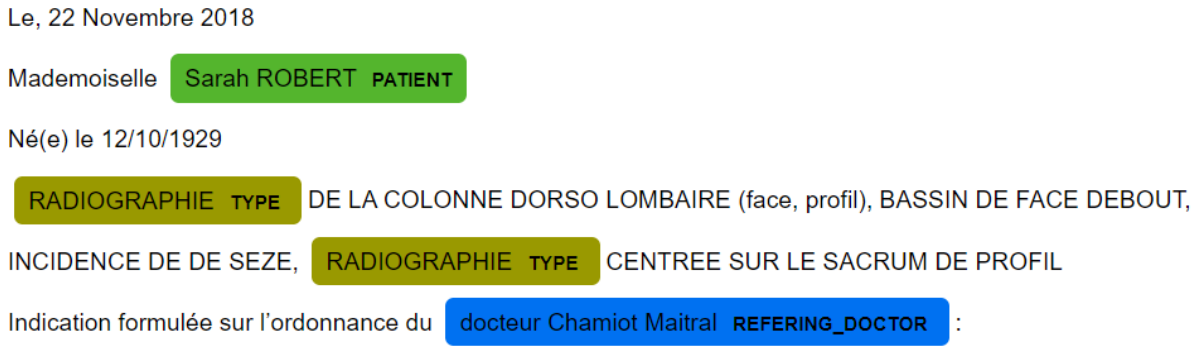


FIGURE 5 – Example of missed entities

Figures 4 and 5 present examples of prediction errors made by our model. In figure 4, the model detects the name of the patient and the name of the doctor, but has a problem with the REFERING_DOCTOR entity. The model detects the entity, but it is badly defined, since other words not belonging to the entity are included in REFERING_DOCTOR. In figure 5, the model has detected the name of the patient, the type of intervention and the referring doctor, but has not detected the patient's birthdate nor the date of the intervention.