OCR of Medical Records

Optical Character Recognition of Individual Health Records

BSc + MSc in Informatics and Computer Engineering
Programming Paradigms
PRODEI018

Gil Manuel Almeida Domingues - 201304646 (gil.domingues@fe.up.pt)
Pedro Martins Pontes - 201305367 (pedro.martins.pontes@fe.up.pt)
Rui Miguel Teixeira Vilares - 201207046 (rui.vilares@fe.up.pt)

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, 4200-465 Porto, Portugal

July 11, 2017

Abstract

This project aims to make it possible to convert to a digital format the growth chart found in child's individual health record, responding to the current effort to digitize medical records. Thus, from a technical point of view, the major domains concerning this project are image processing and optical character recognition. This tool's operation can be summarized as follows: the user selects a file corresponding to the image of the aforementioned table, from which each entry is extracted. Internally, this was achieved through the recognition of the table and later division into cells, whose individual analysis will allow the creation of a digital model of these entries, which is then displayed in the form of a table. Results proved to be better for high quality images, in which the text is at high contrast and properly spaced.

Keywords

Image Processing, Medical Records, Programming Paradigms, *OpenCV*, Optical Character Recognition, *Tesseract*.

Contents

1 Introduction

Developed within the scope of the *Programming Paradigms* course, this project envisioned the implementation of an *OCR* tool for digitizing medical records, using different programming languages and applying various programming paradigms.

More specifically, this tool should be able to perform *OCR* on the growth chart kept in a child's individual health record - with handwritten information about the child's height, weight and cephalic perimeter at a given date and age - and display its contents.

With this proof of concept, it is possible to obtain the information on a child's growth chart record in digital form, by providing a scanned image of said chart via the application's Web Interface and saving the results.

Given the trend towards the digitalization of records, this project has the potential to have a real world application, as it addresses the need for the efficient conversion from handwritten to digital format of medical records.

2 Related Work

There has been some work done in reading machine-printed documents in known predefined tabular-data layout styles (J.H. Shamilian, H.S. Baird and T.L. Wood). In these tables, textual data are presented in record lines made up of fixed-width fields.

These approaches often do not contemplate the handling of handwritten data, but most consist in applying image segmentation and use neural networks (J. Hansen).

Several programming languages have been used to accomplish this.

Early applications used both *MatLab* (J. Hansen) and and *C* (D. Andre), primarily for performance reasons.

More recently, the use of programming languages which fit an Object-Oriented paradigm has been increasingly observed, such as *C++* (*Tesseract*), *Java* (V. R. Nadal) and *C#* (H. E. Saadi).

This transition to Object-Oriented programming languages was made possible by the emergence of sophisticated and much more powerful computers (I. A. Albidewi), and impulsed by the need to achieve higher accuracy levels. Additionally, Object-Oriented principles and design patterns can be introduced as means to cope with design complexity, providing modularity and reusability, and therefore improving the efficiency of the development process.

3 System Description

3.1 Conceptual Description

3.1.1 Functionalities

The major domains concerning this project are image processing and optical character recognition. In fact, the application developed has a single functionality: to extract the data from the growth chart of a child's individual health record.

3.1.2 Architecture

The system was designed as a Web Service, and its core was split into four separate modules, each one with different responsibilities, as illustrated in **Figure 1**.

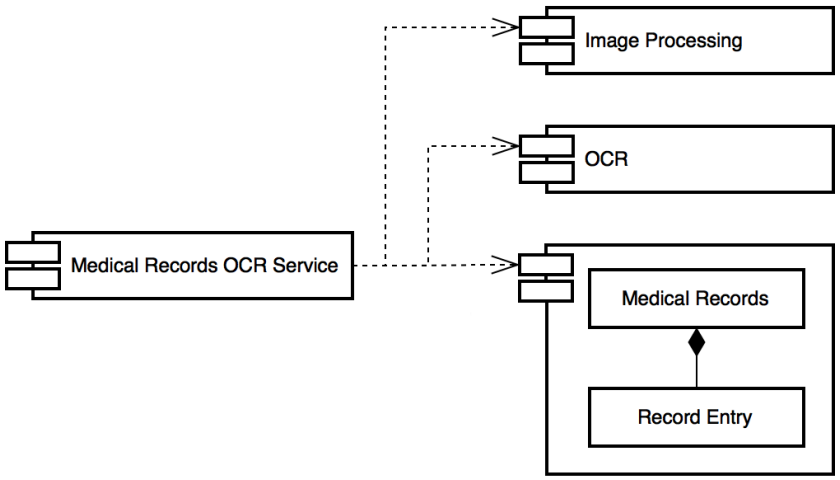

Figure 1: Module diagram, depicting the application's architecture.

Image Processing

Image processing consists in manipulating an inputted image so further analysis can be done either on its whole or on segments. The steps to follow will, of course, be dependent on the problem at hand. In this case, the goal is twofold.

Firstly, this module will need to detect the lines which compose the table in the original image and create a clone of this image without said lines. Lastly, in order for the OCR module to be able to recognize which characters belong to which cells in the table, the module is responsible for detecting the origin coordinates and size of each cell, storing this information in an array before feeding it to the OCR module.

OCR

Optical Character Recognition (*OCR*) pertains to the conversion of images with typed, handwritten or printed text into machine-encoded text. It has become an important and widely used technology for digitizing text so that it can be electronically edited, searched and stored more compactly.

In this module, each individual cell from the growth table will be processed, being subjected to optical character recognition to obtain the data it contains - the date, the child's age, height, weight or cephalic perimeter.

Medical Records

This module encompasses the data representation of the information retrieved via *OCR*, kept following a Object Oriented paradigm, as detailed in Figure 2.

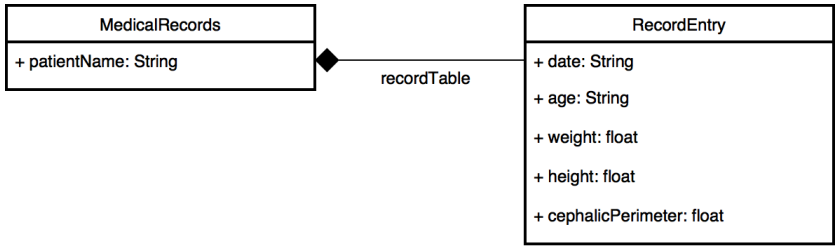

Figure 2: Class diagram, with the objects used for keeping the information extracted.

Medical Records Service

The system was implemented as a Web Service. This module depends on the other modules described to provide its service: upon receiving a request, the application proceeds to process the image selected, splitting the growth table on it into individual cells. Once this step is concluded, *OCR* is performed in each cell, with the data being kept as a set of record entries grouped as a child's medical records.

3.1.3 Programming Languages

Java is a general-purpose computer programming language that is Object-Oriented, Imperative, Reflective and Concurrent. Because it fits the Object-Oriented paradigm, it is well suited for keeping the information generated by the *OCR* process - height, weight and cephalic perimeter at a given date and age -, as this data is of a structured nature. Furthermore, the support for concurrent programming allows to implement the Web Service in such a way that it is able to respond to multiple requests in parallel. As an alternative

to *Java*, *C++* was also considered, as it also has Imperative and Object-Oriented programming features. Ultimately, *Java* was preferred to *C++* for multiple reasons. Firstly, it allows the programmer to deal with memory and file management in a simpler way, which is a major upside, particularly taking into account the need to frequently create clones of the original image for manipulation and analysis. Another preferable characteristic of *Java* is the greater degree of ease with which one can create a server application, when compared with *C++*.

As part of the implementation of the Web Service, several web languages were used: *HTML*, *CSS* and *JavaScript*. *HTML* is the standard markup language for creating web pages and web applications. Along with *CSS* and *JavaScript*, it is one of the three core technologies for World Wide Web content production. Both *HTML* and *CSS* follow a Declarative paradigm, as they define structure, but do not detail flow control. *JavaScript* is a high-level, dynamic, weakly typed, multi-paradigm, and interpreted programming language. As a multi-paradigm language, *JavaScript* supports Event-Driven, Functional, Imperative and Object-Oriented programming styles.

3.2 Implementation Description

3.2.1 Tools and Development Environment

Several tools were used in the implementation.

OpenCV stands for Open Source Computer Vision Library, and it puts together a large collection of algorithms for image processing. This library was implemented in optimized *C/C++*, supports multiple operative systems and provides interfaces for several programming languages, including *Java*.

Tesseract is an *OCR* engine for various operating systems implemented in *C/C++*. It provides interfaces for several programming languages - including *Java* - and it supports several idioms. Another important feature is the possibility to train the engine to recognize different font styles.

Apache Tomcat, often referred to as Tomcat Server, is an open-source *Java* servlet container. Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which *Java* code can run.

The development process itself was conducted in a specific environment. Windows 10 was the operating system used, and IntelliJ IDEA 2016 Ultimate was the IDE of choice, as it facilitated the management of the several

dependencies imposed by the selected image processing and *OCR* tools.

3.2.2 Implementation Details

This section details the interconnection between the system's several modules. Additionally, further relevant implementation details are presented.

The system was implemented as a Web Service. Upon receiving a request, the application proceeds to process the image selected, splitting the growth table on it into individual cells. Once this step is concluded, *OCR* is performed in each cell, with the data being kept as a set of record entries grouped as a child's medical records. The data generated is then presented in a table, displayed in a new web page. These steps are presented in further detail next.

Web Server

As mentioned, *Apache Tomcat* was used to support a *Java* servlet container. Specifically, Java Server Pages (JSPs) was the selected specification. JSPs allow the creation of dynamically generated web pages based on HTML, by allowing Java code to be interleaved with static web markup content - in this case, HTML -, with the resulting page being compiled and executed on the server before being delivered.

Two pages were devised: a first for submitting the scanned image of a growth chart to be processed, and a second for displaying the results.

Excerpt 1: Excerpt of code capturing the information flow between Java and Javascript code.

```
<script>
 const fileName = request.getParameter("file_name");
 const recordTable =
 '<%= (new MedicalRecordsOCR(fileName)).process() %>';
 recordTable = JSON.parse(recordTable);
</script>
```

As it can be seen in the excerpt above, JSP pages use several delimiters for scripting functions, which allow the bridge between the Java code and the Javascript script embedded in the HTML code.

Image Processing

The image processing step begins when the analysis of a document is requested. The goal of this step is twofold. First, it calculates the regions of interest for the table provided, that is, it determines the coordinates, width and height for each cell of the table. Secondly, it removes from the original image the borders of the table, so that no lines in the table are mistakenly

interpreted as characters in the character recognition step. The way this goals are attained will be discussed in detail.

The program starts by firstly converting the image to gray-scale and then using an adaptive threshold to convert the image to black and white. Afterwards, two clones of the resulting image are run through an erosion and dilation algorithm, one clone using a vertical matrix and the other an horizontal one. This process extracts the lines which compose the inner and outer borders of the table. The resulting image containing only the table lines is used for two purposes. It is used to extract the table borders from the original image, so that those lines will not interfere with the character recognition process.

The second use for the border image is to determine the regions of interest, one for each cell in the table. The intersections between the lines of the table are calculated in a separate mat object. As these resulting intersections may be rectangles, four kernels are run through the image to extract the four corners of the polygon. If, as a result of a low quality image, one of the four corners of an intersection is missing, an algorithm is run that generates the missing point from those already found.

Finally, using the corners of the intersections, one rectangle is generated for each cell in the table and the list of rectangles is sent to the character recognition module for analysis.

OCR

Each individual cell from the growth table, defined by its coordinates and dimensions - obtained after the step of image processing - is subjected to optical character recognition, via *Tesseract*.

To improve the results of *OCR* operations, dictionary capabilities and other *Tesseract* heuristics were disabled.

Additionally, *Tesseract*'s *char whitelist* - the list of characters recognized - was reset and limited to those expected to be found in this type of records.

Finally, *Tesseract*'s *page segmentation mode* was set to a mode in which each cell within the image was treated as being a single text line.

Excerpt 2: Excerpt of code capturing *Tesseract*'s configurations.

```
// disable dictionaries
tesseract.setTessVariable("load_freq_dawg", "false");
tesseract.setTessVariable("load_system_dawg", "false");

// reset the character set accepted
tesseract.setTessVariable("tessedit_char_whitelist",
                          "adm0123456789.,-/");
```

```
// change page segmentation mode
tesseract.setPageSegMode(SINGLE_LINE_SEGMENTATION_MODE);
```

4 Conclusions

The importance of keeping accurate medical record cannot be overstated. *OCR* of previous medical records provides invaluable benefits in terms of the efficiency associated with record keeping.

Our approach consisted in obtaining the information on a child's growth chart record, by processing a scanned image and performing *OCR* on each individual cell extracted from the growth chart.

As for the image processing segment of the application, the best results are obtained when the table lines are drawn in black and the background is white, with high level of contrast between the two. It is also important for the table to be as straight in the image as possible.

It should be noted that *Tesseract* was not conceived with the purpose of being used for optical character recognition of handwritten documents. Nonetheless, the configurations applied proved sufficient to obtain satisfactory results to a certain degree.

Image resolution	Error rate
High	0%
Medium	5.5%
Low	90%
Mean	31.8%

Table 1: Best error rates observed using images of different resolutions.

As seen in **Table 1**, optimal results were obtained when the image used had a high resolution, with an error rate as low as 0%, calculated for each as follows:

$$Errorrate = \frac{Numberofcharactersmisidentified}{Totalnumberofcharacters}$$

Further testing showed that the spacing between the characters also had a significant impact on the results obtained.

In conclusion, *OCR* results were better for high quality images, and where the text is at high contrast (sharp, dark font on a white background) and properly spaced.

5 Future Work

Despite being a proof of concept, the ultimate goal of this project is to implement a fully functional tool, ready to open, process and digitize a child's growth chart with the minimal error possible. To fulfill it, there are some aspects still to be improved.

First among them, image processing could be enhanced, for instance, by incorporating into the tool the capability of noise reduction or the possibility of rotating the image so as to ensure the table is horizontal.

Second, but perhaps even more impacting on the reduction of the error rate, comes the training of *Tesseract* to better deal with the different calligraphies expected to be found in such records. This could be achieved by training a neural network to recognize different font styles - in this case, different calligraphies.

Though it was not implemented at this stage, the possibility of providing the graph corresponding to these inputs - as a way of validating the results obtained - was also considered, as was allowing the edition of the table - so as to dynamically correct errors introduced in the data.

6 Resources

6.1 *Bibliography*

J.H. Shamilian, H.S. Baird and T.L. Wood: *Retargetable table reader*. Fourth International Conference on Document Analysis and Recognition. 1997.

D. Andre *Learning and upgrading rules for an OCR system using genetic programming*. First IEEE World Congress on Computational Intelligence Evolutionary Computation. 1994.

J. Hansen: *A Matlab Project in Optical Character Recognition*. University of Rhode Island.

V. R. Nadal: *Optical Character Recognition - A Java Project*. 2012.

H. E. Saadi: *A C# Project in Optical Character Recognition (OCR) Using Chain Code*. 2011.

I. A. Albidewi: *The Use of Object-Oriented Approach for Arabic Documents Recognition*. International Journal of Computer Science and Network Security. 2008.

Tesseract Development Team: *Improve Quality*. Available at `https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality`. March 21st, 2017.

T. Mota: *Identification and Quantification of Oncocytes in Microscopic Images*. FEUP. June 23rd, 2014.

6.2 *Software*

IntelliJ IDEA 2016 Ultimate, JET BRAINS, `https://www.jetbrains.com/idea/`.

Apache Tomcat, Apache Software Foundation, `https://tomcat.apache.org/`.

Appendices

A User Manual

A.1 Setup

Due to the dependence to several libraries, the application's setup can be somewhat complex, as there are several pre-requisites to consider.

- OpenCV;
- Tessaract;
- Tomcat.

The following are some useful setup instructions:

- https://medium.com/@aadimator/how-to-set-up-opencv-in-intellij-idea-6eb103c1d45c
- http://www.jbrandsma.com/news/2015/12/07/ocr-with-java-and-tesseract/
- http://tess4j.sourceforge.net/tutorial/
- https://www.jetbrains.com/help/idea/2017.1/creating-and-running-your-first-web-application.html

A.2 Usage

Once the setup is complete, the application is fairly simple to use.

The user must run the server, access the Medical Records *OCR* web page and select a file corresponding to a child's growth table previously scanned. This image must be at the location specified in MedicalRecordsOCR.FILES_LOCATION and be either a *.PNG*, *.JPG* or *PDF*.

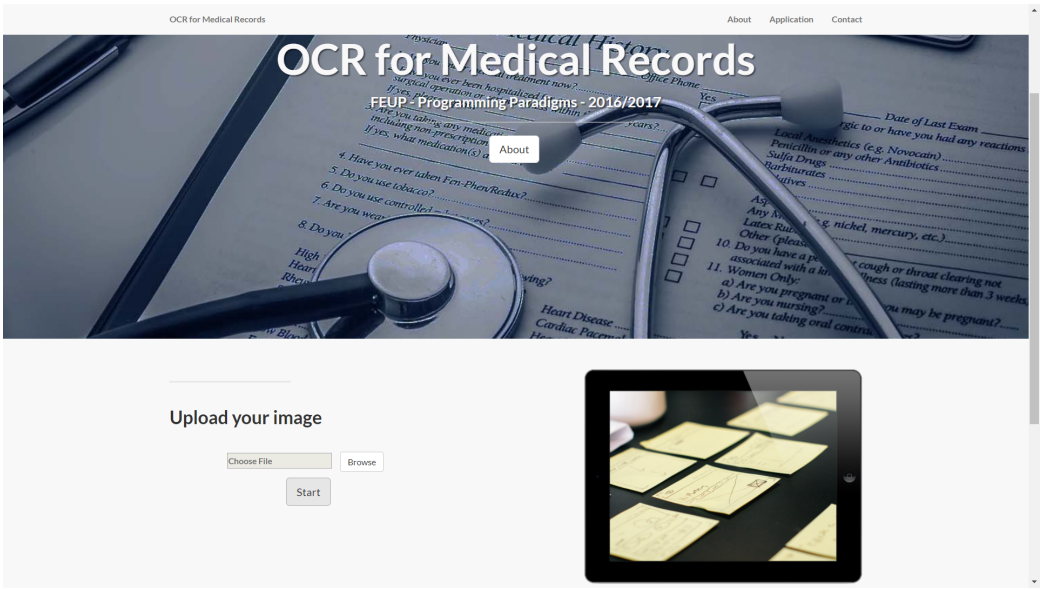

Figure 3: Service's homepage.

Upon receiving the request, the application proceeds to analyze the image selected. When this process finishes, the data generated will then presented in a table, displayed in a new web page.

Data	Idade	Peso	Estat.	P. Cef
19.1.96	18d	3950	52	36
31.1.96	31d	3600	54.5	37
7.3.96	2m	5500	59	39.5
9.5.96	4m	6500	66	41.5
11.7.96	6m	8500	70	43
24.10.96	10m	10100	73.5	45
13.2.97	13m	11400	80.5	46
16.4.97	15m	12000	83	47

Figure 4: Results page.

It should be noted that, for efficiency purposes, the analysis process halts when a row is signaled as being empty, and the following rows are not processed.