

Development of **U**niversal **M**edical **F**orm **R**eader

Project Report

Research team:

* Mufleh Al-Shatnawi (Principal investigator)
* Ralph Lisak (Research Head)
* Chirag Arora (Research Assistant)
* Manreet Kaur (Research Assistant)

**Table of Contents:**

* Executive Summary
* Methods/Approach
* Results
* Conclusions and Implications
* Limitations of system
* Directions for Future Research
* References

**Executive summary:**

Founded in 1994 and based in Markham ON, York Card Technology Inc. is a digital health service provider operating under the name YMS (ymsmd.com). The company supplies Ontario healthcare practitioners with the Electronic Medical Record (EMR) software applications required to bill, schedule, and manage patient information electronically.

YMS faces an automation challenge connected to creating and processing various forms used by medical professionals. The process for adding forms that are new to the YMS system is entirely manual and often takes days to complete, which negatively impacts the company’s profitability. In order to address this challenge, YMS has partnered with Seneca to develop a software solution to automate the process of new form digitization. The resulting proof-of-concept Universal Form Reader (UFR) analyzes new Portable Document Format (pdf) forms and extract attributes and keywords, such as patient name, age, etc. The UFR will have a significant impact on YMS’ bottom line and, equally important, it will pave the way for advanced research and development activities supporting cost reduction and proactive patient care opportunities.

In this project, the Seneca team developed an UFR system that leverages machine learning techniques using Python libraries to apply automated Optical Character Recognition (OCR) to read and scan the fillable pdf files. The objective was to use this system to process a new medical form to obtain a description of the interactive elements in pdf file in the form of html code. Such code can be then used to upload interactive elements of pdf file into the EMR (Electronic medical record) system, significantly minimizing the amount of manual intervention. We estimate that this new tool will reduce the time required to process new forms by 60% - 70%.

**Methods/Approach:**

The Universal Form Reader (UFR) system developed here uses on of the existing machine learning techniques, a decision tree, read/analyze pdf file and generate html script for interactive elements of that file.

**Requirements of system:**

1. Installation of integrated development environment for Python language such as Anaconda (Spyder).
2. All the interactive elements of PDF file should be fillable.
3. PDF file should not be protected, i.e., no permission should be required to open the file.

The universal pdf reader system is built up with following steps:

1. Reading Fillable PDF:

For reading a fillable PDF file, a Python library pdf query was used which applies Decision Tree Analysis (a rendition of older algorithm like C4.5). This library provides the tree structure of fillable PDF file which makes it easy to convert to xml format. This library not only gives the elements of pdf file in tree structure but also does Location (Spatial) Analysis and gives relative positional properties in terms of right, left, top and bottom.

1. Converting to xml:

In this step the tree structure is saved in xml format. This format was chosen because it is descriptive and detailed while at the same time easy to read/comprehend.

1. Parsing xml to get properties of the interactive elements:

The next step was to parse the xml and get the elements and their properties – their type (checkbox, text or text fill up), position (x0, y0, y1, x1) in form of data frame (csv file). This was done using ET library which drags down from root to the lowest layer in file and get the all the branches. Therefore, a csv was made containing all the elements and their properties.

1. Linking html script to database of EMR:

To make this html work on EMR system, it should be linked with its database. In order to achieve that a database dictionary, provided by company in form of csv file (linked history), was used to link the elements with database of EMR system. Note: The linked history can be manually modified.

1. Label finder:

The closest text element was selected through k-means nearest neighbor (a machine learning algorithm) in which Euclidian distance between the checkbox and text near the checkbox is calculated. Thus, text which is at minimum distance from element is considered label for it which is saved as another csv.

1. Generating html script:

An html script is created using the file html generator which uses three different csv files. First, the html generator uses the csv generated from parsing the xml file to generate the interactive elements in html, this generator uses the object type column of csv to generate the different types of interactive elements of pdf. Secondly, linked history csv and label finder csv are used to give as many as accurate labels to interactive elements of pdf in html script. Linked history is also used for linking the elements of pdf to database.

**Regular updates to be made in system:**

The UFR system is linked to the database of EMR system so a regular update must be made in csv file of linked history which contains the name of field in pdf file and its database name. For new pdfs which might not have their database name stored in linked history, these new names must be updated from time to time to ensure correct operation of the system. We think that over time this linked history will stop evolving as enough hospital forms are processed.

**Where to find the systems code and setup:**

The code for universal pdf reader system can be found on GitHub, where our team has created a private repository named Seneca Universal PDF Reader and given its access to the team members of YMS company. The code files in repository contain the readme file which describes the different releases of code files made in this repository, and it also contains five python files of the system and a csv file named linked history. Our code is structured in a way that its five python files are solution to five different challenges which combines to give solution to the final challenge.

For working with this system it is required to go through user guide and system setup presentations which can also be found in this repository folder. The user guide presentation guides in working with python files of system whereas system setup presentation describes the entire procedure to setup the environment in the machine which is required to work with the code files of system.

**Results:**

This system was tested on multiple (20) pdf files. In general, it worked well. However, we found out that occasionally there were cases when the input file was not completely fillable which resulted in errors, which emphasizes the importance of providing fillable pdf forms.

Based on the conducted tests, we estimate that the accuracy rate for labelling the checkboxes and textboxes, is typically within **60-70%** range.

Note: Accuracy, defined as a degree to which results of system conforms to standard = (correct no of elements / total number of elements) \* 100

* Forms that yield all the elements in a csv format without any errors with code:

**Accuracy = (18/ 20) \* 100 = 90%**

It is observed that 90% of fillable forms worked well with system without any errors with code whereas the rest 10% had the exceptions like winding font or images used in pdf due to which fields were not made fillable or not extracted.

* Average accuracy of labels of interactive fields of forms:

**Fibroscan form** = (34/47)\*100 = **72.3%**

**Sleep lab requisition** = (40/75)\*100 = **53.3%**

**Lithotripsy\_booking\_referral**= (25/39) \* 100 = **64.1%**

**NYGH Parental\_req** = (35/62) \* 100 = **58%**

We observed that the labelling accuracy changes form to form according to the organization of elements in the pdf forms. However, it is believed that on average the system will be able to achieve labelling accuracy **between 50-60%** (depending upon the fonts used in forms) for almost all kind of pdf forms.

**Conclusions and Implications**:

Generally, the developed code performed well for majority of tested forms. However, we encountered some cases where the results were not as good. We estimate that the manual effort required to bring a new form into the system will be reduced by **60-70%**.

**Limitations of system:**

Our UFR is enormously powerful and cognizant of the company requirements however it still can be improved and has some limitations. These limitations include, not able to convert non fillable pdf to fillable pdf as this work is done by adobe. Even adobe can convert forms to interactive only to certain extent at times missing some elements with winding font in pdf. Moreover, the accuracy of labelling of interactive elements is also low which decreases the efficiency of system. The location, with and structure of interactive elements in html developed by system as compared to original pdf is slightly different and may require some manual editing post creation of HTML. These limitations of this research system can be addressed in future.

**Directions for Future Research:**

In the future we can use REST API Framework or FLASK to make an API out of this using some very basic code or Computer Vision techniques, which is a method which helps us to gain higher understanding of the media items like images and videos which will allows us to extract the images or reconstruct them using the pixel data from which we can extract the border table size, subheading and possibly some more labels which will further improve the accuracy. This might be the way to go ahead for phase two. We can also include probabilistic analysis for identify labels utilizing bayes theorem. Secondly, the width and height for textboxes in html is not same as the width and height of them in pdfs even after unit conversion does not function - this can be also a part of future research.